



Facultad de Ingeniería  
Escuela de Ingeniería Civil Informática

# **MODELO DE CREDIBILIDAD EXTENDIDO SOBRE T-CREO**


Por


**Pamela Quinteros Henríquez**

Trabajo realizado para optar al Título de  
**INGENIERO CIVIL EN INFORMÁTICA**  
Prof. Guía: Ana Aguilera Faraco  
Prof. Co-Referente: Yudith Cardinale Villarreal  
Octubre 2022

# Resumen

Actualmente, se produce una cantidad importante de información en la web. Las redes sociales son un ejemplo de ello, donde millones de usuarios generan e intercambian información de todo tipo. Si bien las redes sociales se han transformado en un medio ideal para compartir información con facilidad, también son un entorno óptimo para la propagación de información falsa. Dada esta situación, el desarrollo de herramientas automáticas de análisis de credibilidad para estas fuentes de información se ha vuelto un tema esencial.

En este contexto, el trabajo presentado  continuación tiene como objetivo principal extender el modelo de credibilidad existente *World White Web*, incorporando detección de bots y análisis semántico de texto en posts de Twitter. Estas nuevas funcionalidades permiten detectar la confiabilidad de un usuario, en base a la gramática utilizada en los textos, las características sociales de su cuenta y la coherencia entre los conceptos utilizados en la publicación, aportando para una evaluación más certera a la herramienta de credibilidad.

Haciendo uso de algoritmos de aprendizaje, se logró entrenar un modelo de aprendizaje Random Forest, tanto para el idioma inglés y español, con una precisión sobre el 80 %  tras hacer las pruebas necesarias, y el desarrollo de un algoritmo para calcular la credibilidad semántica del texto de un tweet, utilizando la distancia semántica y las definiciones de palabras clave; obteniendo resultados coherentes al comparar los resultados obtenidos con la nueva funcionalidad con los obtenidos con el modelo original.

# Índice general

<b>Resumen</b>	<b>II</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Marco Conceptual y Estado del Arte</b>	<b>3</b>
2.1. Marco Conceptual . . . . .	3
2.1.1. Redes Sociales . . . . .	3
2.1.2. Credibilidad en redes sociales . . . . .	4
2.1.3. Bots . . . . .	4
2.1.4. Detección de bots en redes sociales . . . . .	4
2.1.5. Análisis de texto en redes sociales . . . . .	5
2.2. Estado del Arte . . . . .	7
2.2.1. Métodos de detección de bots en Twitter . . . . .	7
2.2.2. Métodos de análisis semántico aplicados a Twitter . . . . .	12
2.2.3. Comparación de métodos existentes . . . . .	15
<b>3. Definición del Problema y Análisis de Requerimientos</b>	<b>19</b>
3.1. Formulación del Problema . . . . .	19
3.1.1. Sistema actual . . . . .	19
3.2. Solución Propuesta . . . . .	22
3.2.1. Filtro de detección de bots . . . . .	22
3.2.2. Filtro de análisis semántico . . . . .	23
3.3. Objetivos . . . . .	24
3.3.1. Objetivo General . . . . .	24
3.3.2. Objetivos Específicos . . . . .	24
3.4. Metodología . . . . .	25
3.5. Especificación de Requerimientos . . . . .	26
3.5.1. Requerimientos Funcionales . . . . .	26
3.5.2. Requerimientos No Funcionales . . . . .	26

<b>4. Diseño</b>	<b>27</b>
4.1. Diseño Arquitectónico . . . . .	27
4.2. Diseño de funcionalidades . . . . .	28
4.2.1. Diseño filtro detección de bots . . . . .	28
4.2.2. Diseño filtro análisis semántico . . . . .	32
4.3. Diseño de Pruebas . . . . .	34
4.3.1. Pruebas de Funcionalidad: Detección de bots . . . . .	35
4.3.2. Pruebas de Funcionalidad: Filtro Análisis Semántico . . . . .	35
4.3.3. Integración . . . . .	36
<b>5. Implementación</b>	<b>37</b>
5.1. Herramientas utilizadas . . . . .	37
5.1.1. Software . . . . .	37
5.1.2. Lenguajes de programación . . . . .	38
5.2. Filtro Detección de Bots . . . . .	39
5.2.1. Clasificadores . . . . .	40
5.2.2. Script para detección de bots . . . . .	44
5.3. Filtro Análisis Semántico . . . . .	47
5.3.1. Fase Léxica-Sintáctica . . . . .	47
5.3.2. Fase Semántica . . . . .	50
<b>6. Pruebas</b>	<b>52</b>
6.1. Detección de Bots . . . . .	52
6.1.1. Funcionamiento de Implementación . . . . .	52
6.1.2. Llamada desde Node JS . . . . .	53
6.1.3. Comprobación de resultados con herramienta externa . . . . .	54
6.2. Filtro de Análisis Semántico . . . . .	56
6.2.1. Análisis Léxico-Sintáctico . . . . .	56
6.2.2. Análisis semántico . . . . .	59
6.2.3. Llamada desde Node JS . . . . .	62
6.3. Integración . . . . .	63
6.4. Tiempos de ejecución . . . . .	69
6.4.1. Detección de bots . . . . .	69
6.4.2. Análisis semántico . . . . .	70
<b>7. Implantación</b>	<b>71</b>
7.1. Implantación . . . . .	71
7.1.1. Requerimientos . . . . .	71
7.2. Documentación . . . . .	71
7.2.1. Manual de Usuario . . . . .	71

<b>8. Conclusiones</b>	<b>78</b>
<b>Bibliografía</b>	<b>80</b>
<b>A. Matrices de Confusión</b>	<b>87</b>

# Índice de figuras

2.1.	Diagrama de sistema de clasificación [1] . . . . .	7
2.2.	Diagrama de componentes de sistema DeBot [2] . . . . .	8
2.3.	Diagrama de arquitectura de método LSTM nivel de cuenta de usuario [?] .	10
2.4.	Diagrama de arquitectura de método LSTM nivel de tweet [?] . . . . .	10
2.5.	Diagrama de flujo de plug-in [3] . . . . .	11
2.6.	Diagrama de arquitectura de solución general y un ejemplo particular [4] . .	12
2.7.	Rendimiento de clasificación de BTM, LDA y mezcla de unigramas en dataset de Q&A. K representa proporción de data [5] . . . . .	13
2.8.	Diseño de sistema para extracción de intereses [6] . . . . .	14
2.9.	Diseño de arquitectura OBIE (Ontology-Based Information Extraction) [7]	15
3.1.	Diagrama de Modelo de Credibilidad de T-CREo . . . . .	20
3.2.	Diagrama de credibilidad incluyendo ambos filtros propuestos (en morado).	24
3.3.	Diagrama de metodología . . . . .	25
4.1.	Diagrama de flujo de solución propuesta . . . . .	28
4.2.	Diagrama de filtro detección de bots . . . . .	29
4.3.	Diagrama de proceso de evaluación y selección del modelo de clasificación	32
4.4.	Diagrama de filtro de análisis semántico . . . . .	34
5.1.	Extracto de script de detección de bots . . . . .	44
5.2.	Extracto de script de detección de bots . . . . .	45
5.3.	Extracto de script de detección de bots . . . . .	46
5.4.	Integración de script Python a Node JS . . . . .	46
5.5.	Inclusión de función que llama a script de Python . . . . .	47
5.6.	Código respectivo a la fase léxica-sintáctica . . . . .	48
5.7.	Método para consultar a DBpedia por Python . . . . .	49
5.8.	Estructura para extraer entidades . . . . .	49
5.9.	Código respectivo a la fase semántica . . . . .	50
5.10.	Integración de script Python a Node JS . . . . .	51
5.11.	Inclusión de función que llama a script de Python . . . . .	51

6.1.	Resultado de predicción en consola de Visual Studio Code . . . . .	52
6.2.	Resultado al invocar script Python usando Node JS . . . . .	53
6.3.	Parser tree y etiquetas POS de texto en inglés . . . . .	56
6.4.	Parser tree y etiquetas POS de CoreNLP en inglés . . . . .	57
6.5.	Parser tree y etiquetas POS de texto en español . . . . .	57
6.6.	Parser tree y etiquetas POS de CoreNLP en español . . . . .	57
6.7.	Reconocimiento de entidades en tweet en inglés . . . . .	58
6.8.	Reconocimiento de entidades en CoreNLP en inglés . . . . .	58
6.9.	Reconocimiento de entidades en tweet en español . . . . .	59
6.10.	Reconocimiento de entidades en CoreNLP en español . . . . .	59
6.11.	Consulta de prueba a DBpedia en inglés . . . . .	60
6.12.	Consulta de prueba a DBpedia en español . . . . .	60
6.13.	Resultado de consulta a DBpedia en idioma inglés . . . . .	60
6.14.	Resultado de consulta a DBpedia en idioma español . . . . .	60
6.15.	Extracto de consulta Wordnet del verbo “drink” . . . . .	61
6.16.	Extracción de sinónimos para diferentes sentidos de la palabra “drink” . . .	61
6.17.	Extracto de consulta Wordnet del verbo “tomar” . . . . .	61
6.18.	Extracción de sinónimos para diferentes sentidos de la palabra “tomar” . . .	62
6.19.	Función para invocar script de Python . . . . .	63
6.20.	Resultado obtenido de ejecución en segundo plano . . . . .	63
6.21.	Servidor local desplegado . . . . .	64
6.22.	Formato de solicitud en Postman . . . . .	64
6.23.	Resultados obtenidos tras analizar solicitud de Postman . . . . .	65
6.24.	Puntaje obtenido por servidor de extensión . . . . .	65
7.1.	Activar modo desarrollador . . . . .	72
7.2.	Cargar paquete de extensión . . . . .	72
7.3.	Extensión WWW agregada satisfactoriamente . . . . .	72
7.4.	Ventana extensión T-CREo . . . . .	73
7.5.	Resultados de cálculo de credibilidad de tweets . . . . .	74
7.6.	Administración de extensiones . . . . .	75
7.7.	Vista de proyecto en la ventana de extensiones . . . . .	75
7.8.	Opciones de extensión . . . . .	76
7.9.	Configuración de parámetros . . . . .	77
A.1.	Matrices de confusión de entrenamiento dataset en inglés . . . . .	87
A.2.	Matrices de confusión de validación dataset en inglés . . . . .	87
A.3.	Matrices de confusión de entrenamiento dataset en español . . . . .	88
A.4.	Matrices de confusión de validación dataset en español . . . . .	88

# Índice de tablas

2.1.	Tabla de comparación de métodos de detección de bots . . . . .	16
2.2.	Tabla de comparación de métodos de análisis semántico . . . . .	18
5.1.	Descripción de dataset [8] obtenido a través de MIB. . . . .	40
5.2.	Datasets de entrenamiento y validación en español e inglés . . . . .	41
5.3.	Mejores resultados de clasificador: dataset en inglés . . . . .	43
5.4.	Mejores resultados de clasificador: dataset en español . . . . .	43
6.1.	Resultados pruebas modelo español . . . . .	54
6.2.	Resultados pruebas modelo inglés . . . . .	55
6.3.	Validación del análisis de credibilidad incluyendo detección de bots (inglés) . . . . .	66
6.4.	Validación del análisis de credibilidad incluyendo detección de bots (español) . . . . .	67
6.5.	Análisis de filtro semántico con cuentas en inglés y español . . . . .	68
6.6.	Tiempo de ejecución de análisis de credibilidad de las Tablas 6.3 y 6.4 . . . . .	69
6.7.	Tiempo de ejecución de filtro de análisis semántico de Tabla 6.5 . . . . .	70



# Capítulo 1

## Introducción

Las redes sociales son plataformas en línea que permiten el libre intercambio de información entre los usuarios que componen su comunidad, siendo ejemplos claros Twitter, Facebook, Instagram, etc. Tras su creación, se han vuelto un nuevo canal para la comunicación entre personas que utilizan la web, constituyendo un mecanismo esencial en la vida de muchos usuarios a la hora de intercambio de información, comunicación y socialización.

Debido a que en estos espacios se genera un volumen de data considerable diariamente, es una tarea ardua el poder validar la información que es compartida. A medida que las redes sociales siguen creciendo, aumentando exponencialmente el número de personas que se vuelven usuarios dentro de éstas [9] [10], se vuelve más difícil mantener el control de las actividades o acciones que realiza la comunidad. Añadiendo esto a la gran influencia que tienen las redes sociales en la formación de opiniones y para informarse de eventos de la actualidad, el mal uso de estas plataformas para actividades dañinas y la diseminación de desinformación era inminente [11]. Un gran ejemplo de esta situación fue estudiado en [12], donde se analiza la presencia de bots en Twitter en las elecciones presidenciales chilenas del año 2017.

Esto ha creado la necesidad, tanto para investigaciones como para uso cotidiano, de poder analizar o calcular la credibilidad de la información contenida en una publicación en una red social. Considerando el alto volumen de datos generado en estas plataformas, se requiere desarrollar herramientas que permitan automatizar este análisis. En la actualidad, nuevos factores en estas plataformas incluyen nuevos desafíos en esta área, siendo uno de los principales el uso de bots para la manipulación de otras cuentas y realización de fraudes [13] [14] [15], cuentas utilizadas para la propagación de información o noticias falsas [16] [17], posts sobre información incoherente o sin sentido, entre otros.

Este trabajo se centra en el proyecto *T-CREo*, una herramienta automática de análisis de credibilidad enfocada en Twitter [18][19][20] y extender su modelo de credibilidad considerando la detección de bots y el reforzamiento del análisis de credibilidad de texto haciendo uso de análisis semántico.

El documento está estructurado de la siguiente manera:

- **Capítulo 2:** Marco Conceptual y Estado del Arte. En este capítulo se expondrán los conceptos clave para el entendimiento del contexto del proyecto y trabajos relacionados.
- **Capítulo 3:** Definición del Problema y Análisis de Requerimientos. Esta sección define la solución propuesta y los objetivos requeridos para su realización.
- **Capítulo 4:** Diseño. En este capítulo se expondrán las herramientas utilizadas y el diseño de implantación del trabajo.
- **Capítulo 5:** Implementación. En esta sección se mostrará en detalle la implementación realizada en el trabajo.
- **Capítulo 6:** Pruebas. En este capítulo se expondrán los resultados obtenidos al realizar las pruebas diseñadas para probar el correcto funcionamiento de las funcionalidades desarrolladas.
- **Capítulo 7:** Implantación. En este capítulo se detallará dónde será implementado el producto del trabajo.
- **Capítulo 8:** Conclusión. En esta última sección, se presentará la conclusión del trabajo en su totalidad en perspectiva del autor.

## Capítulo 2

# Marco Conceptual y Estado del Arte

En este capítulo en la sección 2.1 se presenta el Marco Conceptual, donde se expondrán los conceptos clave para el entendimiento del contexto en el que se desenvuelve y desarrolla tanto el proyecto inicial como las nuevas funcionalidades, como también en la sección 2.2 se expone el estado del arte, donde se analizan diferentes trabajos que presentan soluciones y/o propuestas similares que se utilizaron como referencia para la solución propuesta de este trabajo.

### 2.1. Marco Conceptual

#### 2.1.1. Redes Sociales

Las redes sociales pueden ser definidas como sectores en la Internet (web) donde los usuarios publican y comparten todo tipo de información (personal o profesional), con terceras personas que pueden ser conocidos o absolutos desconocidos[21], así también como plataformas virtuales donde los usuarios que la componen interactúan entre ellos mediante el intercambio de información de todo tipo.

Independiente de cuál sea la definición que se utilice para referirse a una red social, es cierto que éstas son espacios sociales que han sido creadas virtualmente para facilitar la interacción entre personas o usuarios, donde las interacciones poseen características particulares como el anonimato (total o parcial) y la facilitación de contacto sincrónico y asincrónico. [22]. Estas plataformas tienen la característica de tener una gran cantidad de usuarios en su comunidad. Por ejemplo, Twitter en el año 2020 alcanzó los 339,6 millones de usuarios, Facebook en el mismo año llegó a los 2.449 millones de usuarios, seguidos también por Youtube, Instagram, TikTok y LinkedIn [23].

### 2.1.2. Credibilidad en redes sociales

El concepto de credibilidad puede definirse como el nivel de confianza o veracidad percibido de un objeto, evento o persona [18] y establece una medida de evaluación a la información analizada respecto a su cualidad de poder ser “creíble” [24]. Dependiendo de la fuente de información de la que se extraiga, los niveles de credibilidad de ésta pueden deducirse.

En el caso de las redes sociales, donde no existe una clasificación de información definida, deducir los niveles de credibilidad se vuelven un desafío no menor, principalmente por la cantidad de información que se genera diariamente en estas plataformas.

Al ser toda esta información creada por personas, generalmente sin una referencia para poder verificar la información compartida, generan una incertidumbre respecto a la confianza y calidad de la información que es presentada, fomentando la necesidad de desarrollar mecanismos o herramientas para validar y clasificar toda aquella información [18].

### 2.1.3. Bots

Los bots pueden definirse como una aplicación de software que es automatizada para realizar trabajos o tareas específicas que se le han asignado a través de scripts o código[25]. Estos son conocidos como Robots Web o, como son conocidos mundialmente, bots. Estas aplicaciones pueden realizar tareas que son simples y/o repetitivas a una velocidad mayor de lo que una persona podría hacer por su cuenta.

Los bots pueden categorizarse como “inofensivos” (chatbots, crawlers, de información, de entretenimiento, etc.) y “dañosos” (spambots, hacking bots, scrapers, imitadores)[25].

### 2.1.4. Detección de bots en redes sociales

La detección de bots es una prioridad clave de seguridad para cualquier negocio o empresa con presencia online [26]. En redes sociales, detección de bots engloba a todos los métodos que permiten su detección, siendo el concepto de bot social definido como cuentas automatizadas de usuario que generan contenido o interactúan con otros usuarios dentro de la red social [27].

A medida que pasan los años y crecen las comunidades en las redes sociales, los bots se han vuelto mucho más sofisticados, obligando a los investigadores y desarrolladores de esta área a tener que buscar nuevas técnicas que permitan detectarlos.

La influencia de los bots en redes sociales es considerable, al componer entre el 9 % a 15 % de los usuarios y siendo los responsables de generar un gran porcentaje de contenido dentro de éstas. Como se puede observar en la gráfica de [28], Twitter poseía 436 millones de usuarios activos a la fecha de enero del 2022; considerando que en años anteriores se calculaba que cerca del 35 % de los contenidos publicados en Twitter correspondía a

usuarios bot [29], se puede dimensionar la influencia que éstos pueden llegar a tener en la plataforma. Generalmente son utilizados para aumentar artificialmente la popularidad de una persona o movimiento [30], influenciar o manipular elecciones [12], manipular los mercados financieros [31] [32], diseminar spam y/o información falsa [16][17] y realizar fraudes o robar información de otros usuarios [13] [14] [15].

En esta misma área, se pueden encontrar los siguientes conceptos:

- **Sybil:** Cuentas que representan identidades falsas que generalmente son controladas por un número pequeño de usuarios reales, utilizando estas identidades para coordinar campañas para propagar spam y malware [33].
- **Cyborg o usuario híbrido:** Cuentas que son gestionadas por humanos y bots de manera alterna [34].

### 2.1.5. Análisis de texto en redes sociales



El análisis de texto o “text analysis” es una técnica de máquina de aprendizaje que permite comprender de manera automática información de texto, como pueden ser correos, reseñas de productos, tweets, etc. extrayendo información específica de esta misma, como palabras clave, entidades, sentimientos analizados en el texto, entre otros [35].


En el contexto particular de las redes sociales, la información de texto que se enfoca en analizar con esta categoría de técnicas es la extracción de información cuantitativa de los comentarios o publicaciones que se generan en estas plataformas. En este trabajo en particular, se definen y se analizan 3 niveles de análisis de texto que se describen a continuación.

#### 2.1.5.1. Análisis léxico

El análisis de texto a nivel léxico corresponde al análisis de las palabras contenidas en el texto por sí mismas, identificando la categoría gramatical de cada palabra (sustantivo, adjetivo, verbo, etc.) y colocando la etiqueta gramatical correspondiente[36] para poder trabajar de manera correcta las palabras que componen el texto al momento de analizar el texto a un nivel superior. Esta fase es muy similar al proceso de análisis léxico realizado por los intérpretes o compiladores de lenguajes de programación, principalmente debido a que el resultado obtenido tras este proceso es un grupo de tokens.

#### 2.1.5.2. Análisis sintáctico

 Respecto al análisis sintáctico, se puede definir como el proceso de análisis y revisión ~~de que~~ la oración de interés, compuesta por tokens o componentes léxicos, se encuentre organizada o construida de manera correcta siguiendo un grupo de reglas en particular[37]. 


Debido a que esto se aplica a lenguaje natural, se refiere principalmente a que lo escrito en un lenguaje específico siga las reglas definidas por aquel lenguaje mismo. Un ejemplo de esto es el orden en que el inglés ordena los componentes gramaticales dentro de sus oraciones, los adjetivos anteponiéndose a los sustantivos. Haciendo un análisis sintáctico del texto se puede corroborar que la forma en que se encuentra redactada la información es la correcta por el lenguaje en que fue escrita  por lo que se podría deducir la formalidad o seriedad del contenido de su información.

### 2.1.5.3. Análisis semántico

El análisis semántico de texto representa la información basada en las relaciones significativas de un texto escrito, estructurado como una red de palabras asociadas unas a las otras de manera cognitiva [38]. Este análisis permite la extracción de opiniones significativas, definiendo grupos de conceptos emergentes en vez de analizar la frecuencia de palabras aisladas. A pesar de que las redes sociales son un foco de interés en el área de análisis semántico, éste ha sido poco explorado, principalmente debido a la dificultad de encontrar las herramientas adecuadas o la complejidad de la data, siendo más utilizadas técnicas orientadas a minería de datos como análisis de texto [24][39][40][38][41].

En la actualidad, se han descrito varias técnicas para lograr realizar un análisis semántico a textos dentro de redes sociales (principalmente en Twitter), entrando en las siguientes categorías: modelado de temas [42], extracción de información [43], sistema de recomendación [44], análisis de sentimientos [45], similitud semántica [46] y resumen de texto [47]. Se requiere un paso previo, definido como la etapa de preprocesamiento, donde se limpia el texto.

### 2.1.5.4. Ontología

Puede ser definida como una base de conocimiento con un lenguaje especializado, conocimiento teórico y fáctico. Posee una estructura y establece un marco para acomodar, asociar y relacionar conceptos y proporciona contexto para aquellas relaciones: [48]  Se compone de:

- **Conjunto de conceptos:** Palabras que representan un objeto, procedimiento, etc.
- **Vocabulario controlado:** Un conjunto de palabras acordadas con definiciones comunes a la comunidad.
- **Taxonomía:** Una lista de palabras especializada en una jerarquía generalizada.
- **Tesauro:** Muestra como los conceptos están asociados y son equivalentes.
- **Esquema:** Una especificación para la organización y definición de data, sus propiedades y relaciones.

- **Teorías:** Afirmaciones del concepto.

## 2.2. Estado del Arte

A continuación, se presentarán métodos y/o proyectos que abordaron temas relacionados con este trabajo, mostrando sus ideas principales, sus contribuciones y su forma de operar. Se dividirán en dos secciones: Métodos de detección de bots en Twitter y Métodos de análisis semántico aplicados a Twitter. Esta revisión es necesaria debido a que el objetivo de este trabajo es desarrollar dos nuevos filtros que aporten mayor precisión al actual modelo, enfocándose en estos dos aspectos particularmente.

### 2.2.1. Métodos de detección de bots en Twitter

#### 2.2.1.1. Detectando automatización de cuentas de Twitter [1]:

Este trabajo se enfoca en la clasificación de personas, bots y cyborgs en Twitter. Luego de una investigación y medición previa con una colección sobre 500.000 cuentas de Twitter, se logró observar las diferencias entre estas tres clasificaciones de cuentas de usuario a través de su forma de interactuar, el contenido de sus tweets y sus propiedades. El método propuesto en este trabajo se compone de cuatro partes importantes: (1) un componente basado en entropía, (2) un componente de detección de spam, (3) un componente de propiedades de cuenta y (4) un tomador de decisiones.

Los datos utilizados corresponden a las características extraídas de un usuario desconocido para determinar la probabilidad de que corresponde a una persona, cyborg o bot.

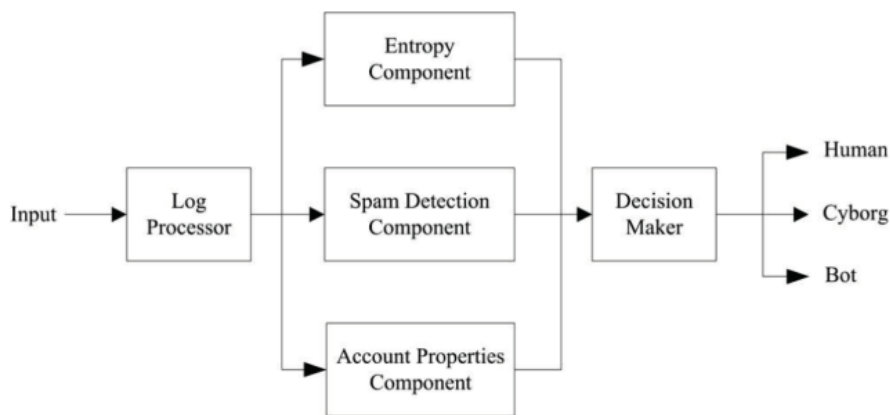


Figura 2.1: Diagrama de sistema de clasificación [1]

El componente de entropía se encarga de detectar tiempo periódico o regular entre tweets, que es una característica de automatización. El detector de spam utiliza una variante de clasificación bayesiana para detectar patrones de texto reconocidas como spam. El componente de propiedades de cuenta permite detectar desviaciones bot de la distribución normal de humanos. Finalmente, el tomador de decisiones, basado en un algoritmo Random Forest, analiza las características indicadas en los componentes anteriores y clasifica la entrada como bot, persona o cyborg.

A pesar que el sistema puede diferenciar a un humano de un bot con alta precisión, existen dificultades al poder clasificar correctamente a una cuenta correspondiente a la clase cyborg. Sin embargo, la precisión global del sistema resulta ser de un 96 %.

#### 2.2.1.2. Debot: Detección de bots a través de correlación [2]:

Debot es un sistema de detección de bots en Twitter desarrollado en la Universidad de Nuevo México. El sistema realiza una búsqueda de cuentas correlacionadas haciendo uso de la correlación deformada (correlación extendida por deformación dinámica del tiempo) al analizar su actividad, dada la idea de que los seres humanos no pueden ser altamente sincrónicos en períodos largos de tiempo.

Los problemas que pretende abordar este método son: (1) la dependencia de datasets etiquetados para aprendizaje supervisado y (2) que no se considera las características de usuarios cruzados. Debot trabaja en base a la correlación de actividades sin la necesidad de datos etiquetados, es decir, un enfoque no supervisado. La precisión del método DeBot es de un 94 %, siendo más rápido y eficiente que el sistema de suspensión de Twitter. La arquitectura de DeBot se compone de cuatro bloques mostrados en la Figura 2.2:

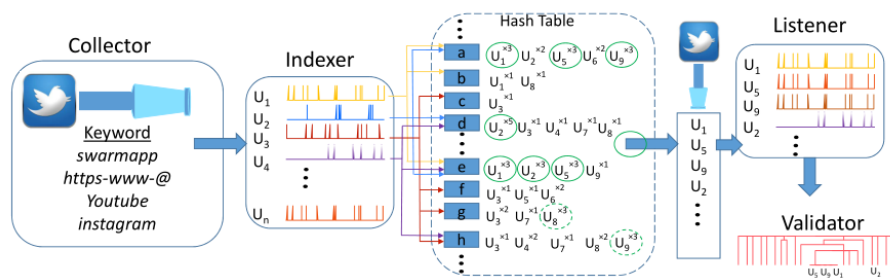


Figura 2.2: Diagrama de componentes de sistema DeBot [2]

1. **Collector:** Se encarga de la recolección de tweets en base a palabras clave, utilizando el filtro de Twitter API. También forma la serie de tiempo del número de actividades de cada usuario, filtrando aquellas cuentas que tengan sólo una actividad, para luego entregar los datos al siguiente bloque.



2. **Indexor:** Toma como entrada las series de tiempo de las actividades de usuario, dividiendo cada una de ellas en múltiples cubos hash. Los usuarios que colisionan en un mismo cubo son informados como usuarios sospechosos.
3. **Listener:** Analiza de manera exclusiva a los usuarios sospechosos. A diferencia del primer bloque (Collector), recibe todas las actividades realizadas por los usuarios sospechosos de un período de T horas. Filtra todos los usuarios que tengan menos de 40 actividades registradas en el período de tiempo establecido.
4. **Validator:** Valida las series de tiempo resultantes del bloque anterior (Listener) y verifica su validez, calculando una matriz de correlación deformada en pares sobre un conjunto de usuarios para agruparlos jerárquicamente hasta un límite de distancia restrictivo. Los usuarios individuales se consideran falsos positivos y los grupos estrechos se identifican como bots.

Las contribuciones de este trabajo corresponden a: (1) el desarrollo de un sistema casi en tiempo real de detección de bots en redes sociales no supervisado; (2) el desarrollo de un método hashing sensible al lag para el agrupamiento de usuarios basados en sus correlaciones y (3) una precisión de detección del 94 % y con la capacidad de detectar bots que otros métodos no son capaces de identificar.

#### 2.2.1.3. Detección de bots utilizando Deep Neural Networks [?]

El método presentado por Sneha Kudugunta y Emilio Ferrara consiste en una red neuronal profunda basado en memoria contextual a corto plazo (LSTM) que aprovecha tanto contenido como metadata para la detección de bots a nivel de tweet. En las pruebas hechas con esta arquitectura, se demostró que, utilizando un solo tweet, ésta puede alcanzar una alta precisión de clasificación (sobre 96 %) al identificar bots de humanos. Al probar la misma arquitectura para probar detección de bots a nivel de usuario, se logró alcanzar una precisión de clasificación muy cercana al 100 %. Este sistema utiliza un conjunto de características relativamente pequeño e interpretable y requiere una cantidad mínima de data de entrenamiento.

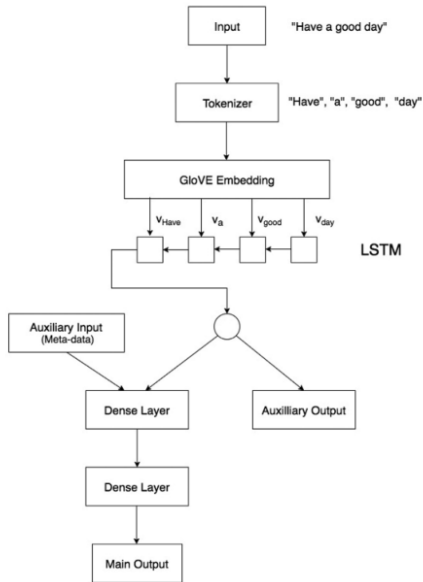



Figura 2.3: Diagrama de arquitectura de método LSTM nivel de cuenta de usuario [?] 

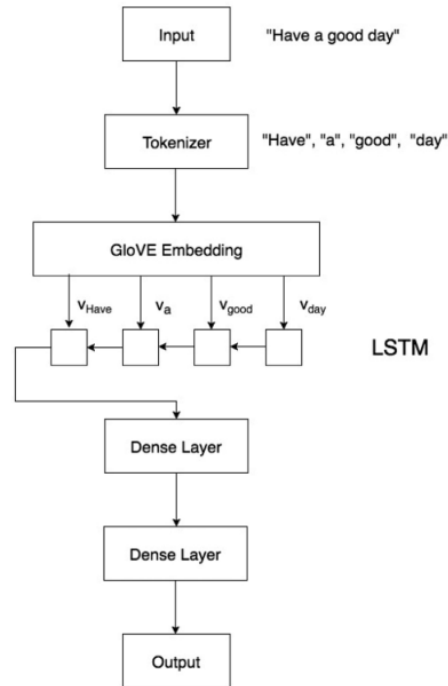



Figura 2.4: Diagrama de arquitectura de método LSTM nivel de tweet [?] 

#### 2.2.1.4. Detección de bots a través de crowd-sourcing y clasificación [3]

Debido al desarrollo más sofisticado de los bots en la actualidad, este método presenta la creación de un dataset GTR (Ground Truth Reference) de cuentas de Twitter utilizando crowd-sourcing, incluyendo en estas cuentas reales, híbridas y Sybils (bots de redes sociales) para el entrenamiento de un clasificador supervisado. Adicionalmente se desarrolló un plug-in para la identificación de bots en Twitter antes que el usuario interactúe con la cuenta posiblemente automatizada. Las actividades realizadas en este trabajo son las siguientes:

- **Adquisición de data:** Se recolectó una muestra aleatoria grande de tweets publicados recientemente utilizando Twitter Streaming API y luego se utilizó Twitter API REST para sacar información de las cuentas, siendo cerca de 1.8 millones de cuenta. A continuación, se etiquetó y clasificó manualmente las cuentas de Twitter, seleccionando 2.000 cuentas del total extraídas.
- **Entrenamiento de calificadores y etiquetado:** Este proceso se define en 3 fases: (1)

selección de los clasificadores; (2) entrenamiento de los clasificadores y (3) etiquetado de la data. Los clasificadores seleccionados debían ser familiares con la interfaz de Twitter, además de ser expertos en TI. Tras el entrenamiento, se pudo ver una gran mejora en el criterio de etiquetado, teniendo como resultado un 96 % de precisión y un coeficiente Kappa de 0,61, indicando que el equipo de clasificadores tenía un gran porcentaje de acuerdo.

- **Clasificación y evaluación:** Se entrenaron los modelos de clasificación con 4 algoritmos de aprendizaje diferentes: árbol de decisiones, red Bayesiana, Support Vector Machine (SVM) y red neuronal artificial multicapa. Al momento de entrenar, se consideraron dos situaciones: (1) que sólo se debían clasificar dos categorías de cuentas de usuario de Twitter (Sybil y humano) y (2) que existían tres posibles categorías para las cuentas de Twitter (Sybil, humano e híbrido). De los resultados obtenidos, RFT (Random Forest) y BayesNet (red Bayesiana) fueron los dos mejores clasificadores en ambas situaciones, resultando con una precisión de 91 % .

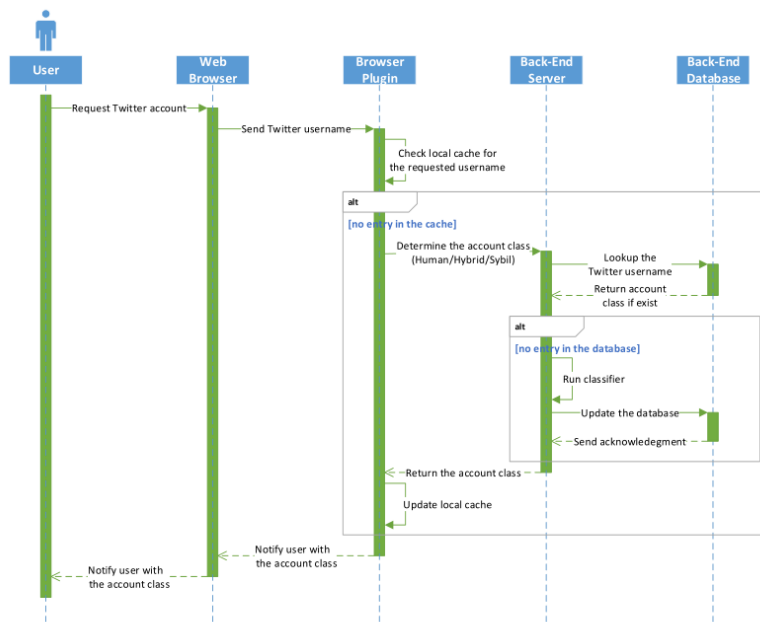


Fig. 3. Sequence diagram showing the relationships between the processes in our system.

Figura 2.5: Diagrama de flujo de plug-in [3]

## 2.2.2. Métodos de análisis semántico aplicados a Twitter

### 2.2.2.1. Enriquecimiento semántico de tweets para construcción de perfiles de usuario [4]

El trabajo presentado en esta instancia tiene como objetivo principal poder resolver hasta cierto punto la dificultad actual respecto a la extracción relevante de información en Twitter dado los grandes volúmenes de datos que se generan día a día. La solución propuesta consiste en representar las actividades en Twitter de los usuarios y modelar sus intereses para permitir su personalización y contrarrestar la sobrecarga de información.

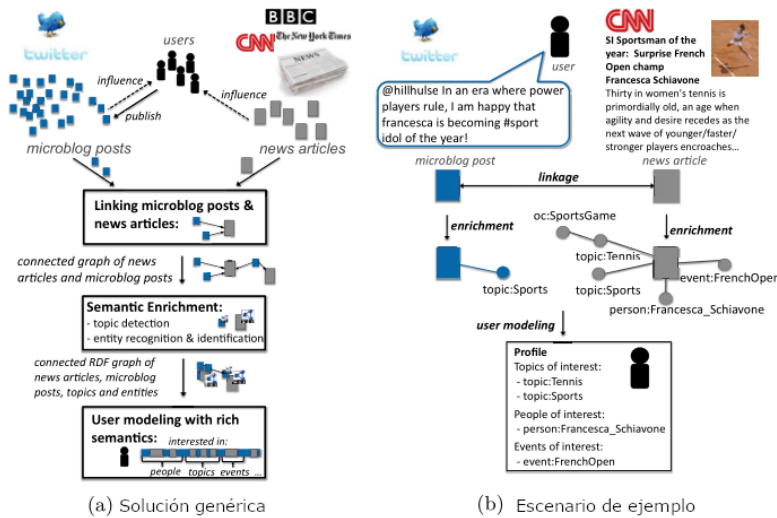


Figura 2.6: Diagrama de arquitectura de solución general y un ejemplo particular [4]

La Figura 2.6 presenta el proceso de este método. Dada una observación de un trabajo relacionado que afirma que el 85 % de los tweets están relacionados a noticias, se pensó en relacionar estos posts con artículos de noticias para enriquecer la semántica y generar un contexto, permitiendo así encontrar también los temas en que los usuarios se encuentran interesados. Para esto, se utilizan las siguientes estrategias:

- Estrategias basadas en URL: La presencia de URL suele ser indicadora de que un tweet está relacionado a noticias y que ésta redirecciona a la fuente. Se consideran las URLs de la persona que hace el tweet y las URLs que puedan aparecer como resultado de interacción con el primer post (respuestas a tweet).
- Estrategias basadas en contenido: Dado que no todos los tweets poseen una URL, se buscan formas para lograr relacionar texto del tweet con noticias. Se comparan las palabras con títulos de noticias y se relaciona con el más similar (Bag of Words).

Pruebas al método indicaron que las estrategias basadas en URL presentaban mayor precisión que las basadas en contenido, resultando en 80 % y 43 % respectivamente.

#### 2.2.2.2. Extracción de temas en textos cortos con Biterm Topic Model (BTM) [5]

En este artículo se presenta un método para el modelado de temas en textos cortos utilizando Biterm Topic Model. Lo que se busca resolver con este trabajo es el problema de escasez de patrones de coincidencia de palabras en textos cortos al utilizar métodos convencionales. BTM modela los temas en un texto utilizando pares de palabras distintas (biterm), ignorando las “*stop words*” y considerando el mensaje o texto completo como una unidad de contexto.

Se utilizó un dataset de tweets para la probar el rendimiento de este método y compararlo con LDA (Latent Dirichlet Allocation) y mezcla de unigramas convencional. Los resultados mostraron que BTM logra aprender temas de mayor calidad y además es capaz de capturar con mayor precisión los temas en un documento que los métodos tradicionales.

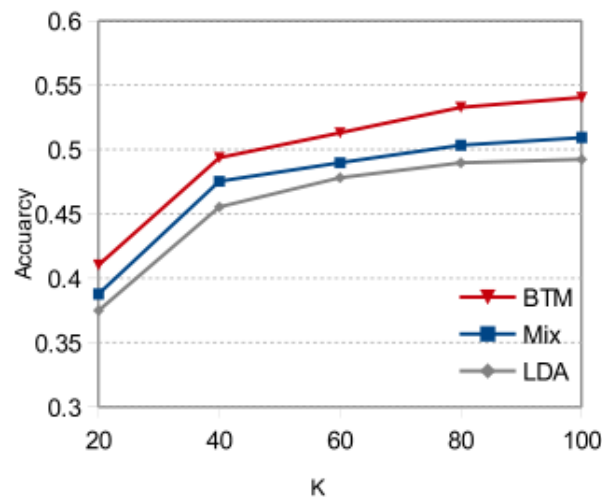


Figura 2.7: Rendimiento de clasificación de BTM, LDA y mezcla de unigramas en dataset de Q&A. K representa proporción de data [5]

En la Figura 2.7 se muestra la comparación entre los tres métodos utilizando un dataset diferente para demostrar que BTM es un método que funciona independiente del dominio.

### 2.2.2.3. (3) Extracción de preferencias utilizando grafos semánticos [6]

El trabajo presentado por M.Jose y K.Rahamathulla propone un método para la extracción de intereses y preferencias de los usuarios haciendo uso de un método basado en grafos semánticos para el análisis de feeds de Twitter, exclusivamente utilizando los textos generados por el usuario dentro de una red social (Twitter). El enfoque dado a este trabajo corresponde a uno orientado a Marketing, al ser uno de los objetivos principales de este trabajo el unir este método con un sistema de recomendación o parecidos. El proceso general de este sistema, descrito por Figura 2.8, es el siguiente:

1. Se recolectan tweets utilizando API Twitter.
2. Se preprocesa la información eliminando “stop words”, extrayendo una lista de palabras clave usando RAKE (Rapid Automate Key Extraction) y aplicando un formato especial a la lista.
3. Se filtran palabras clave que no aportan o no poseen una entidad archivada en DBpedia. El resto de las palabras son extendidas con sinónimos o pequeñas descripciones (de DBpedia) para generar un contexto.
4. Se genera un grafo semántico, las palabras clave siendo nodos y las aristas las relaciones semánticas entre ellos. Un nodo está conectado a otro si, en DBpedia, el primero se encuentra como una entidad relacionada al segundo.

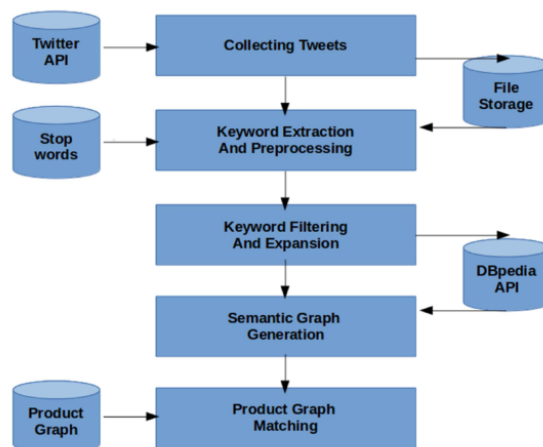


Figura 2.8: Diseño de sistema para extracción de intereses [6]

Dentro del grafo, los nodos que contengan una mayor cantidad de conexiones se consideran como el subgrafo de interés, que es comparado con un “grafo de producto”, un

grafo semántico hecho a base de palabras clave para un producto en específico. Si tienen una alta similitud, entonces el usuario tiene como preferencia aquel producto.

Los resultados obtenidos de las pruebas realizadas confirman que el uso de grafos semánticos con expansión de palabras clave tiene un buen rendimiento (90 %), sobrepasando en precisión a otros métodos tradicionales, como RAKE exclusivo.

#### 2.2.2.4. Extracción de información basada en ontología [7]

El sistema presentado por *Kamel Nebhi* está basado en reglas para el reconocimiento y desambiguación semántica de entidades (lugares, personas, organizaciones, etc.) en tweets. El sistema planteado encuentra el tipo de entidad extraída y la relaciona con una descripción semántica en la ontología formal, recibiendo como datos de entrada documentos semi estructurados o no estructurados (tweets).

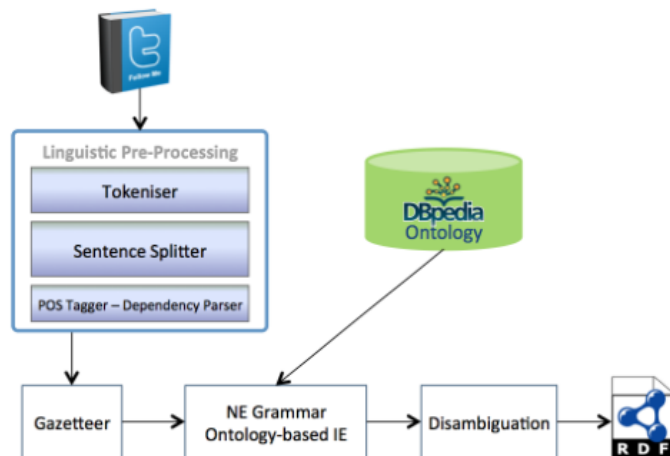


Figura 2.9: Diseño de arquitectura OBIE (Ontology-Based Information Extraction) [7]

Como se muestra en la Figura 2.9, el sistema primero genera un preprocesamiento lingüístico de los datos de entrada, para luego utilizar un diccionario geográfico (gazetteer) para la identificación de entidades. Luego se hace una anotación semántica basada en reglas para pasar luego a la desambiguación de las entidades encontradas dependiendo del contexto y, finalmente, obtener el documento final.

#### 2.2.3. Comparación de métodos existentes

A continuación, se presentarán en dos secciones diferentes, los beneficios o ideas importantes rescatadas de los métodos presentados, mostrando una tabla de comparación

para cada ítem, mostrando los criterios de evaluación importantes a considerar en cada técnica.

### 2.2.3.1. Detección de bots en Twitter

Como se puede mostrar en la Tabla 2.1, se hizo una comparación con criterios de evaluación importantes a considerar respecto a desarrollar un módulo que se encargue de la detección de bots dentro de la herramienta de análisis automático de credibilidad.

- **Detección en tiempo real:** Determina si la técnica requiere recolección de información para servir como datos de entrada o si permite el procesamiento de stream de datos.
- **Depende del idioma:** Si la detección de bots se encuentra restringida por el idioma de los tweets, el alcance de la técnica utilizada se ve limitada.
- **Enfoque del modelo:** Dependiendo del modelo utilizado para la detección de bots, este necesitará un dataset etiquetado para su entrenamiento o no.
- **Posee API:** La presencia de una API para utilizar funcionalidades o utilizar información recolectada por estas técnicas podría ser una ayuda al desarrollo de otros sistemas.
- **Atributos para detección:** Debido a la limitación que existe con la API de Twitter, es importante considerar la cantidad de atributos que requiere la técnica para su funcionamiento.

Métodos	Detección en tiempo real	Depende del idioma	Enfoque de modelo	Posee API	Atributos para detección
(1) Detección de cuentas automatizadas	-	X	Supervisado	-	8
(2) DeBot	X	-	No Supervisado	X	7
(3) Deep Neural Network	-	X	Supervisado	-	16
(4) Crowd-Sourcing	X	X	Supervisado	-	8

Tabla 2.1: Tabla de comparación de métodos de detección de bots

Como se puede observar, la mayoría de las técnicas presentadas dependen del idioma de los tweets. La técnica (1) depende del idioma exclusivamente por tener un componente de spam que se encarga de identificar patrones en el texto, a diferencia de la técnica (3) que toma como entrada el texto dentro del tweet. Con los trabajos expuestos, se llega a la



conclusión que, para incluir el análisis del texto de un tweet como característica para la predicción, se debe entrenar un modelo específico para el lenguaje a analizar.

La técnica (4), si bien contribuye con la elaboración de un dataset y un plug-in para la detección de bots, ninguno de los productos resultantes del trabajo realizado se encuentra disponible para su uso, no existiendo un canal para solicitarlos como tal. Caso similar ocurre con el trabajo (2), donde el resultado final culminó en una API y un sitio web para descargar información sobre las cuentas identificadas como bots por DeBot. Sin embargo, tras haber contactado al equipo para solicitar una clave para su utilización, notificaron que ya no se otorgaba más acceso a la API.

Para el objetivo de este trabajo, es importante que el proceso de detección de bots se pueda realizar en tiempo real, considerando que la extensión funciona actualmente de aquella manera. Dos de estos trabajos cumplen con este objetivo, (2) y (4); sin embargo, ninguno de sus recursos se encuentra disponibles para su uso y, por ende, sus resultados no son posibles de reproducir.

Si bien no se pudo obtener permisos para la utilización de la API de uno de los trabajos revisados o algún extracto de los datasets ocupados para el entrenamiento de modelos o recolección de datos, al contrastar cuáles trabajos se acercan más al objetivo y a las herramientas disponibles para el desarrollo de éste, se pudo explorar con mayor precisión qué métodos y características serían posibles seleccionar para poder obtener resultados positivos al momento de realizar el entrenamiento de los modelos de predicción.

#### 2.2.3.2. Análisis semántico en Twitter

Como se puede mostrar en la Tabla 2.2, se hizo una comparación con criterios de evaluación importantes a considerar respecto a desarrollar un módulo que se encargue del análisis semántico dentro de la herramienta de análisis automático de credibilidad.

- **Usa fuentes externas para contextualizar:** Determina si la técnica utiliza algún tipo de fuente de información externa para comparar o buscar palabras similares para ampliar el contexto del texto de entrada.
- **Reconoce entidades en texto:** Si logra reconocer entidades dentro del texto, ya sea personas, lugares, organizaciones, etc.
- **Reconoce temas en texto:** Si la técnica es capaz de extraer el tema que trata el texto de entrada. Este criterio es utilizado más en ámbito exploratorio en esta fase, no siendo un requisito fundamental para la comparación.
- **Presenta proceso de desambiguación:** Si presenta un proceso dedicado a establecer el significado de una palabra dado el contexto en el que se encuentra.

Métodos	Usa fuentes externas para contextualizar	Reconoce entidades en texto	Reconoce temas en texto	Presenta proceso de desambiguación
(1) Construcción semántica de perfiles de usuario	X	X	X	-
(2) BTM	-	-	X	-
(3) Grafos semánticos	X	X	X	-
(4) Basado en Ontología	X	X	-	X

Tabla 2.2: Tabla de comparación de métodos de análisis semántico

Como puede observarse, a excepción de (2), los trabajos revisados tienen en común tres criterios: el uso de fuentes externas para contextualizar, reconocimiento de entidades y reconocimiento de temas. Si bien el reconocimiento de temas no es un objetivo que se quiera realizar dentro de este trabajo en particular, se exploró este criterio principalmente para tener mayor claridad respecto a las técnicas utilizadas dentro de este enfoque e investigar algún posible procedimiento que fuera beneficioso para el desarrollo de esta funcionalidad de análisis semántico. Tras las investigaciones realizadas, se optó por no incluir este aspecto para no dispersar el trabajo a realizar.

Si bien el trabajo (3) presenta un método interesante para la contextualización y reconocimiento de entidades utilizando un grafo semántico, se optó por sólo considerar el uso de DBpedia para la contextualización. Esto se debe principalmente a que el procedimiento de análisis semántico se realizará para dos idiomas diferentes que se estructuran de otra manera, pudiendo complicar el cálculo dependiendo del tamaño del texto.

Debido a que existen una gran variedad de palabras que cambian su significado dependiendo del contexto, se requiere hacer una desambiguación de las palabras importantes del texto (siendo considerados sustantivos, verbos, entidades, etc.), por lo que un método para la desambiguación basado en similitud de sintaxis o similitud semántica sería un buen procedimiento para realizar este proceso.

## Capítulo 3

# Definición del Problema y Análisis de Requerimientos

### 3.1. Formulación del Problema

La extensión de Google Chrome *T-CREo*, inicialmente desarrollada como una prueba de concepto de un proyecto de cálculo de credibilidad para fuentes en internet instanciado para Twitter [49], realiza un análisis automático de credibilidad utilizando técnicas de *web scraping* y los componentes API REST y API Twitter para lograr realizar este análisis en tiempo real [50],[19]

Los filtros mencionados funcionan de la forma esperable, dada la revisión de su funcionalidades destacadas en [18] y [20]. Sin embargo, el modelo de credibilidad de texto, hasta el momento, sólo se enfoca en un análisis léxico del texto de un post.

Debido a que existe una gran cantidad de bots en Twitter (entre 9 % a 15 % de las cuentas corresponden a bots [51]) y que la mayoría de los usos que se les dan a estos programas suelen ser dañinos para la comunidad de la red social, se debe considerar la naturaleza del usuario como un factor importante para el cálculo de credibilidad de sus publicaciones.

En este trabajo se pretende mejorar el modelo existente de la herramienta web y extender su funcionalidad, considerando factores adicionales para el cálculo de porcentaje de credibilidad y modificando las fórmulas que componen el modelo, si se considera necesario.

#### 3.1.1. Sistema actual

Actualmente, T-CREo se corresponde a una extensión de Chrome que realiza un análisis automático de credibilidad de publicaciones haciendo uso de APIs (API Twitter, API REST) y web scraping[50],[19].

El objetivo de esta herramienta es poder analizar credibilidad de fuentes de información, tomando como factores el texto, el usuario y su influencia social [49]. Dado que esta herramienta pretende extenderse a un futuro a otras redes sociales, los parámetros utilizados para los cálculos son configurables para poder considerar aspectos únicos de cada plataforma (límite de caracteres, lenguaje formal o informal, etc.). Las pruebas operativas del modelo actualmente se han instanciado para ser realizadas en la plataforma Twitter. El modelo de credibilidad utilizado por T-CREo se detalla en la Figura 3.1.

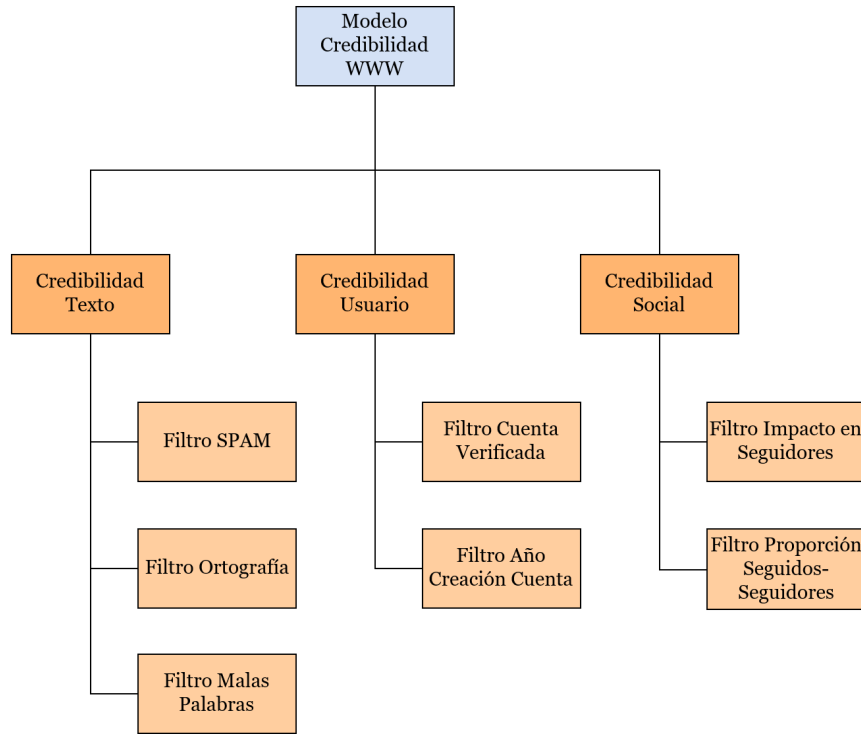


Figura 3.1: Diagrama de Modelo de Credibilidad de T-CREo

A continuación, se mostrarán las fórmulas utilizadas para el cálculo de credibilidad de un post. Cabe mencionar que este enfoque es general para diversas fuentes, pero la mencionada a continuación es la instancia particular de Twitter, mencionada en el artículo [49]:

#### 1. Credibilidad de Texto:

$$TextCred(t.text) = w_{SPAM} \times isSpam(t.text) + w_{BW} \times bad\_words(t.text) + w_{MW} \times misspelling(t.text) \quad (3.1)$$

Donde

- **t** corresponde al tweet a analizar, siendo **t.text** el texto correspondiente a dicho tweet.
- $w_{SPAM}$ ,  $w_{BW}$ ,  $w_{MW}$  corresponden a los pesos o importancia que se le da a la detección de spam, malas palabras y faltas de ortografía, respectivamente. Se debe cumplir que  $w_{SPAM} + w_{BW} + w_{MW} = 1$
- **isSpam()** corresponde a la función que se encarga de evaluar el texto de entrada (t.text) y devolver la probabilidad de que dicho texto sea spam. Devuelve un valor perteneciente al intervalo  $[0, 100]$
- **bad\_words()** es la función que busca malas palabras (groserías) dentro del texto. Devuelve valores pertenecientes al intervalo  $[0, 100]$ .
- **misspelling()** es el filtro encargado de encontrar palabras mal escritas en el texto del tweet. Retorna valores pertenecientes al intervalo  $[0, 100]$

## 2. Credibilidad de Usuario:

$$UserCred(t.user) = Verif\_Weight(t.user) + Creation\_Weight(t.user) \quad (3.2)$$

Donde:

- **t.user** corresponde al usuario del tweet a analizar
- **Verif\_Weight()** revisa si es que la cuenta del usuario está verificada o no. En el caso de que lo esté, devuelve un valor de 50, si no, devuelve un valor 0.
- **Creation\_Weight()** verifica el año en que la cuenta se unió a Twitter, el año de creación de Twitter y el año actual. Si el año en que se unió la cuenta es cercano a la que se creó Twitter (2006), la credibilidad de este factor aumenta considerablemente. Retorna un valor entre  $[0, 50]$ .

## 3. Credibilidad Social:

$$SocialCred(t.social) = Followers\_Impact(t.social_{user}) + FFProportion(t.social_{user}) \quad (3.3)$$

Donde:

- **t.social** corresponde a la metadata social, por lo que  $t.social_{user}$  representa a la metadata social de un usuario en particular.
- **Followers\_Impact()** calcula la influencia que posee una cuenta en específico. Retorna un valor perteneciente al intervalo  $[0, 50]$
- **FFProportion()** calcula la proporción que tiene la cuenta respecto a su cantidad de seguidos y seguidores. Retorna un valor perteneciente al intervalo  $[0, 50]$

Finalmente, los tres valores calculados para los tres aspectos de tweet son considerados en la fórmula global del modelo, correspondiente a:

$$TCred(t) = weight_{text} \times TextCred(t.text) + weight_{user} \times UserCred(t.user) + weight_{social} \times SocialCred(t.social) \quad (3.4)$$

- $weight_{text}$ ,  $weight_{user}$ ,  $weight_{social}$  corresponden a parámetros configurables que determinan el peso o importancia de los factores de texto, usuario y social del post o tweet. Esto permite manipular hasta cierto grado el modelo y dar importancia a los factores que se consideren más relevantes.

## 3.2. Solución Propuesta

A continuación, se presentarán las soluciones propuestas a la problemática presentada.


### 3.2.1. Filtro de detección de bots

Una parte de la solución propuesta corresponde al desarrollo de una nueva funcionalidad para la extensión que permita detectar bots, dando mayor precisión a la sección de credibilidad de usuario que se presenta en la Figura 3.1, permitiendo así analizar al usuario del tweet de interés y devolver su predicción correspondiente de si se trata de un bot o un usuario genuino.

Si bien se ha hecho una revisión de diferentes trabajos que hacen uso de APIs para este proceso [52] y otros que disponen su trabajo para ser utilizados por la comunidad [2], se ha optado por crear y entrenar máquinas de aprendizaje para poder realizar esta funcionalidad.

Similar a algunos trabajos presentados en el Capítulo 2 que restringen su funcionamiento a un lenguaje en específico, se ha decidido crear dos máquinas para poder realizar predicciones en español y en inglés, respectivamente. Esta decisión fue tomada luego de

analizar las diferencias de estructura de cada idioma, considerando el mismo texto del tweet como una característica importante del análisis, complementado también con los factores sociales y estadísticos de este mismo (likes, seguidos, amigos, etc.), por lo que se consideró que hacer un análisis de cada de lenguaje por separado daría mejores resultados que generar una máquina que pudiera trabajar ambos lenguajes de manera simultánea.

Para integrar este nuevo factor, se ha decidido “penalizar” al usuario dependiendo si este es bot o no, considerando el puntaje obtenido por éste mismo en los filtros de Verificación de Cuenta y  Año de creación. La nueva fórmula de credibilidad de usuario siendo la siguiente:

$$UserCred'(p.user) = \begin{cases} UserCred(p.user) & \text{Si } p.user \text{ no es bot,} \\ 0.85 \times UserCred(p.user) & \text{Si } p.user \text{ es bot y } UserCred(p.user) \text{ mayor que 50,} \\ 0.75 \times UserCred(p.user) & \text{Si } p.user \text{ es bot y } UserCred(p.user) \text{ entre 35 y 50,} \\ 0 & \text{De lo contrario,} \end{cases}$$

Siendo el valor de  $UserCred$  calculado con la fórmula original detallada en la Ecuación 3.2, y  $UserCred'$  la nueva fórmula de credibilidad integrando la funcionalidad bot.

### 3.2.2. Filtro de análisis semántico

La otra parte de la solución propuesta es desarrollar una segunda funcionalidad a la extensión que permita realizar un análisis semántico del contenido del tweet que será un apoyo para la sección de credibilidad de texto que por el momento sólo analiza el tweet a nivel léxico. El resultado obtenido de este nuevo filtro se presentará como un factor más al que se deberá indicar su peso antes del cálculo de confiabilidad del tweet, al igual que los otros componentes que la conforman y que fueron presentados en la Ecuación 3.1.

Cabe mencionar que, si bien el sistema actual opera con 3 idiomas (francés, español e inglés), se tomó la decisión de priorizar la realización de esta funcionalidad (y la anterior) considerando los idiomas español e inglés.

$$TextCred'(t.text) = w_{SPAM} \times isSpam(t.text) + w_{BW} \times bad\_words(t.text) + w_{MW} \times misspelling(t.text) + w_{SEM} \times semantic(t.text) \quad (3.5)$$

Donde  $w_{SEM}$  corresponde al peso otorgado al cálculo semántico y  $semantic(t.text)$  la función que calcula el porcentaje obtenido del texto del tweet respecto a la evaluación semántica, devolviendo un valor entre [0,100].

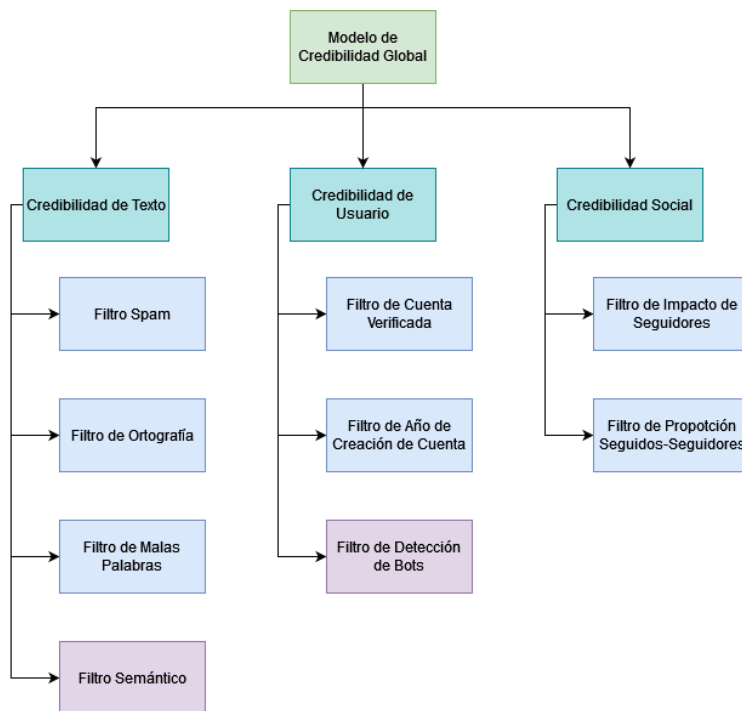


Figura 3.2: Diagrama de credibilidad incluyendo ambos filtros propuestos (en morado).

## 3.3. Objetivos

### 3.3.1. Objetivo General

Extender el modelo de análisis de credibilidad de la herramienta existente, incorporando filtros de detección de bots y análisis semántico de texto.

### 3.3.2. Objetivos Específicos

- Desarrollar un filtro de detección de bots para aumentar la precisión de cálculo de credibilidad.
- Desarrollar un filtro de análisis semántico de texto para aumentar precisión de cálculo de credibilidad.
- Actualizar fórmulas de cálculo de credibilidad de texto y usuario.





### 3.4. Metodología

Para lograr la completitud de los objetivos planteados, se distribuyó el trabajo en etapas, presentadas en la Figura 3.3, que serán explicadas a continuación:

1. **Etapa inicial:** Esta instancia se enfocó principalmente para el estudio y entendimiento de la aplicación existente y la recolección de información de herramientas útiles para el desarrollo de las funcionalidades propuestas, además de la recolección y estudio de trabajos similares para utilizar como punto de referencia.
2. **Etapa desarrollo 1:** Se enfocó en el desarrollo e implementación del filtro de detección de bots. En la medida que se fue desarrollando esta funcionalidad, se realizaron pruebas locales para poder verificar su funcionamiento y detectar algún tipo de detalle que pudiera dificultar su integración al proyecto.
3. **Etapa desarrollo 2:** Aquí el objetivo fue el desarrollo e implementación del filtro de análisis semántico. A medida que se desarrolló esta funcionalidad, se realizaron pruebas locales para verificar el funcionamiento del filtro y detectar algún tipo de detalle que pudiera dificultar su integración al proyecto.
4. **Etapa testing:** Finalmente, la etapa para la realización de pruebas a la aplicación de forma global. El objetivo principal de esta etapa es poder hacer pruebas al modelo y obtener resultados experimentales que puedan comprobar que los filtros desarrollados efectivamente aportan al modelo de credibilidad, mejorando su precisión.

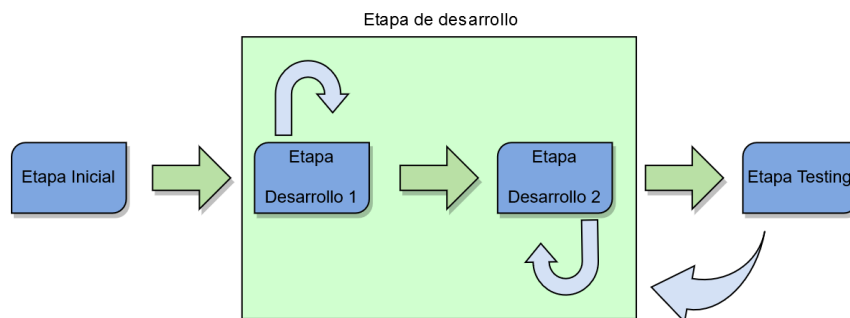


Figura 3.3: Diagrama de metodología

Se consideró que la forma de trabajo debería ser similar a la de una metodología ágil, dado que las funcionalidades a desarrollar no dependen de los otros filtros para su correcto

funcionamiento, además de tener en consideración posibles cambios de tecnologías durante el desarrollo. A sí mismo, se realizaron pruebas locales en las etapas para corroborar de que los filtros desarrollados para el modelo estén en correcto funcionamiento antes de ser implementados en la herramienta y corregir con anticipación cualquier mal funcionamiento que pudo haber sido acarreado hasta la última etapa de trabajo.

## **3.5. Especificación de Requerimientos**

### **3.5.1. Requerimientos Funcionales**

- Se debe detectar bots en tiempo real.
- Se debe realizar un análisis semántico en tiempo real.
- El análisis semántico debe considerar el idioma español y el inglés.

### **3.5.2. Requerimientos No Funcionales**

- Las técnicas utilizadas para ambos filtros deben respetar el límite de características del API de Twitter a utilizar.
- El desarrollo de los filtros debe realizarse con la misma tecnología del proyecto existente (Typescript) y/o con herramientas que sean posibles de integrarse.




# Capítulo 4

## Diseño

En este capítulo se presentará el diseño de la solución propuesta anteriormente, tomando en consideración la herramienta ya existente para explicar los datos que se recibirán y se procesarán para ser incluidos dentro de la fórmula de credibilidad de la herramienta.

### 4.1. Diseño Arquitectónico

A continuación, se presentará el sistema completo de T-CREo desde una vista de alto nivel, considerando la solución propuesta.

Como se puede observar en la Figura 4.1, el flujo comienza cuando el usuario (haciendo uso del browser)  genera una solicitud a la extensión para el cálculo de tweets, generando el envío de los parámetros previamente instanciados por el usuario hacia el modelo de credibilidad. Éste envía una solicitud al API de Twitter para la información del usuario y el tweet correspondiente necesarios para el cálculo de credibilidad.

La información del usuario y su tweet es segmentada acorde a la data necesaria como entrada para el cálculo de credibilidad de los factores que componen el modelo: la credibilidad de texto requiere del mensaje del tweet, la credibilidad social recibe los metadatos sociales del usuario dueño del tweet (cantidad de seguidores, por ejemplo) y la credibilidad de usuario requiere de los atributos de éste.

Finalmente, al haber realizado los cálculos correspondientes, los resultados de cada sección son unidos en la fórmula de credibilidad global para generar el porcentaje de credibilidad del tweet analizado. Este valor es devuelto a la extensión para poder comunicar al browser del resultado y que sea visto por el usuario.

Los módulos resaltados en verde claro corresponden a los dos filtros de la solución propuesta, posicionándose en la sección del modelo de credibilidad que le corresponde. A continuación, se explicará con más detalle el diseño interno de estos filtros y los módulos que lo componen.

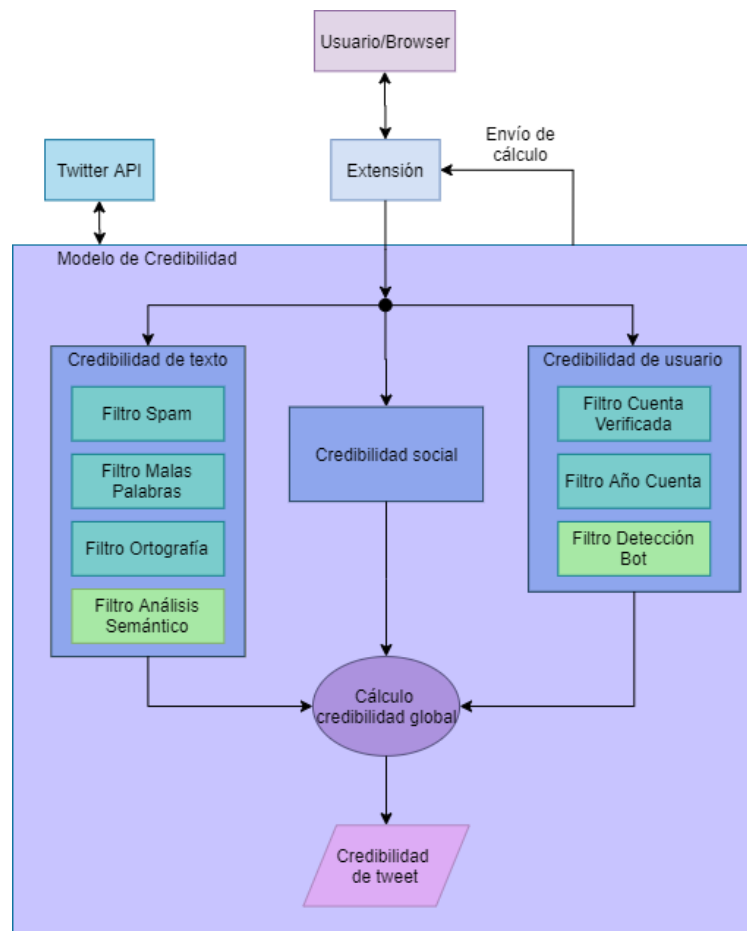


Figura 4.1: Diagrama de flujo de solución propuesta

## 4.2. Diseño de funcionalidades

### 4.2.1. Diseño filtro detección de bots

El objetivo de este filtro es poder identificar con cierta mínima seguridad si el usuario del tweet que se está analizando corresponde a un bot o a un usuario genuino (humano), haciendo uso de características sociales del usuario y características descriptivas del tweet.

A continuación, en la Figura 4.2 se presenta el diseño del filtro de detección de bots. Una vez se haya solicitado el analizar la credibilidad de tweets, los atributos del usuario del tweet son ingresados a la parte del modelo de credibilidad que se encarga de hacer el análisis de usuario. En el caso del filtro de detección de bots, la información necesaria para poder realizar el análisis correspondiente es el nombre de usuario o “screen\_name” que publicó el tweet.

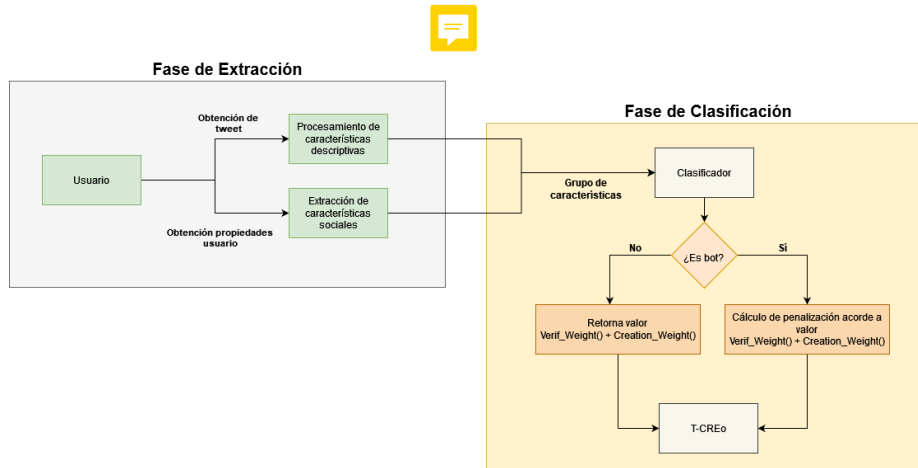


Figura 4.2: Diagrama de filtro detección de bots

Una vez que el nombre de usuario es obtenido por el filtro, éste se encarga de extraer las características sociales de este usuario, a la vez de extraer también un tweet reciente de este mismo usuario, que es procesado para poder limpiar las impurezas o ruido que este contenga (problemas de encoding, emojis, etc.), para posteriormente realizar la extracción de características del texto necesarias (características descriptivas) para poder pasar a la fase de clasificación en conjunto a las extraídas inicialmente al usuario.

Una vez obtenidas las características correspondientes, éstas son utilizadas como entrada para el modelo de clasificación (español o inglés) para ser evaluadas. Una vez es obtenida la predicción del modelo, éste se evalúa. En el caso de que se trate de un usuario legítimo (humano), se retorna a la extensión el valor completo máximo posible para la fórmula, indicando que el usuario efectivamente es humano. Como se ha mostrado en la Figura 4.1, el componente de Credibilidad de usuario considera tres subelementos que aportan al “puntaje” o cálculo de esta métrica.

Dado a que la nueva fórmula de credibilidad de usuario 3.2.1 penaliza al usuario que ha sido clasificado como bot dependiendo de la credibilidad obtenida por los otros dos filtros que la componen (verificación de cuenta y año de creación), dependerá del valor de la suma del resultado de ambas para obtener el valor de credibilidad de usuario; que luego es utilizado para el cálculo de la credibilidad global del tweet y devuelto a la extensión para mostrar.

Como se vio en el Capítulo 2, existen varios trabajos con distintas técnicas para la detección de bots, estando entre ellas a través de “crowd-sourcing”, observación de comportamiento anormal, modos de comunicación y aprendizaje automático, entre otros [53] [54]. Con esto dicho, en este trabajo se optó por utilizar una técnica basada en aprendizaje automático, ya que se ha demostrado su eficiencia en este contexto [54] [55].

Si bien existen una variedad de técnicas que abordan la temática de detección de bots en redes sociales, tales como la detección de actividad sincronizada por períodos largos de tiempo [2], análisis de la relación de los seguidores de una cuenta a través del uso de grafos

[56] y métodos más cercanos al lado investigativo de las actividades de cuentas respecto a un evento en específico, analizando contenido, actividad en el tiempo, entre otros factores [57]; se ha optado por realizar esta funcionalidad haciendo uso de algoritmos de aprendizaje, principalmente por sus notables resultados [58, 59, 60] y su relativa simplicidad de implementación.

Para poder realizar la clasificación de usuarios planteada anteriormente en la Figura 4.2, se requiere utilizar un algoritmo de aprendizaje automático que permita la clasificación a partir de características dentro del tweet y del usuario, representado como el componente Clasificación en dicho diagrama. A partir de la primera experiencia al entrenar la máquina de aprendizaje, más la revisión de más trabajos que realizaron este mismo procedimiento [61], se definieron las siguientes características como entrada a la máquina de clasificación, siendo éstos representados como “*Grupo características*” en la Figura 4.2:



### 1. Características Sociales:

- **Número de seguidores:** Se extrae la cantidad de seguidores que tiene el usuario.
- **Número de status:** Corresponde a la cantidad total de tweets y retweets del usuario.
- **Número de favoritos:** Se obtiene la cantidad de tweets que el usuario marcó como favoritos.
- **Número de listas:** Corresponde a la cantidad total de listas públicas en las que está como miembro el usuario.
- **Número de amigos:** Corresponde a la cantidad total de usuarios que tiene agregados como amigos el usuario en cuestión.
- **Número de retweets:** Corresponde a la cantidad total retweets del usuario.
- **Número de estados:** Corresponde a la cantidad total tweets que tiene/ha hecho el usuario en su cuenta.

### 2. Características estructurales de texto:

- **Número de URLs:** Se extrae el número de URLs que contiene el tweet.
- **Número de Hashtag:** Se extrae el número de hashtags que contiene el tweet
- **Número de Menciones:** Se extrae el número de menciones que contiene el tweet.
- **Promedio wavelet:** Se vectoriza el texto del tweet utilizando un método similar al descrito en [62]. Se vectoriza el texto del tweet haciendo uso de la librería spaCy, haciendo una diferenciación si es un texto en español o inglés, para luego calcular la Transformada Discreta Wavelet del vector. Finalmente, se realiza un

promedio del vector completo y este resultado es utilizado como característica para la clasificación.

- **Razón URLs:** Corresponde a la razón entre la cantidad de URLs presentes en el texto y el total de palabras del tweet.
- **Razón Mención:** Corresponde a la razón entre la cantidad de menciones presentes en el texto y el total de palabras del tweet.
- **Razón Hashtag:** Corresponde a la razón entre la cantidad de hashtags presentes en el texto y el total de palabras del tweet.

Las características presentadas fueron seleccionadas de los trabajos anteriormente revisados en el Capítulo 2, a excepción del Promedio Wavelet, que fue incorporado tras la revisión del trabajo [62].

El proceso utilizado para calcular la característica Promedio Wavelet se explica brevemente en el punto 11). Si bien el trabajo utiliza la Transformada Wavelet Discreta para obtener información espectral de acuerdo al contenido del texto del tweet, esta clasificación se realiza en base a un tópico o tema específico (los autores también incluyeron una prueba con un tema misceláneo). Para la propuesta de este trabajo en particular, se utilizará una técnica similar, en el sentido de que el contenido del tweet se transformará en un vector al que se le calculará la Transformada Wavelet Discreta y luego se calculará su centroide, proceso que no se considera en el procedimiento original.

El resto de características corresponden a valores extraíbles de la propia cuenta de Twitter, correspondientes a características sociales, o a información cuantitativa del propio texto. En concreto, los ítems 3), 6), 8), 9) y 10) de las características enumeradas se extrajeron de [59], mientras que las características 1), 2), 4) y 5) se extrajeron de [59]. Los ítems 12), 13) y 14) son características estructurales del texto que se consideraron interesantes de estudiar, ya que [59] tiene una característica que considera el ratio entre seguidores y seguidos de la cuenta. La idea es poder ver si la razón entre el número de palabras que componen un tuit y el número de menciones/URLs/hashtags puede ser una característica importante que ayude a definir un patrón de comportamiento de los bots sociales en Twitter.

#### 4.2.1.1. Entrenamiento de algoritmo de Clasificación

Si bien hasta el momento se ha explicado el cómo se incorporará el modelo de aprendizaje seleccionado al proceso de detección, ésta misma tiene que ser inicialmente entrenada y evaluada. En la Figura 4.3 se muestra el proceso a utilizar para la definición del algoritmo para clasificar los bots detectados por la extensión.

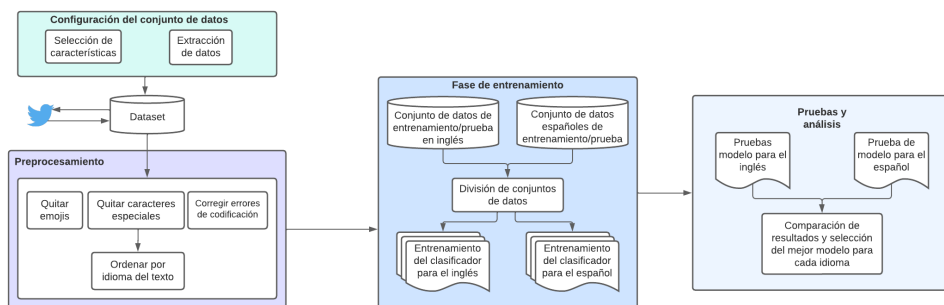


Figura 4.3: Diagrama de proceso de evaluación y selección del modelo de clasificación

Para el entrenamiento de estos algoritmos, se requiere la elaboración o búsqueda de un dataset que contenga tweets etiquetados con su clase correspondientes (de usuarios reales y bots), además de las características anteriormente mencionadas anteriormente. La obtención de este dataset, su compilación y preprocesamiento se detallará más adelante en el Capítulo 5. En este proceso, se evaluarán un conjunto de algoritmos para poder identificar cuál de ellos presenta mejores resultados a través de la evaluación de métricas establecidas (precisión, recall, etc.). Finalmente, cuando se haya identificado el algoritmo que mejor responde a lo requerido, se extraerá el modelo de clasificación para ser utilizado en el actual proceso del filtro.

#### 4.2.1.2. Dataset de entrenamiento

Como se mencionó anteriormente en la sección 3.2.1, se optó por generar dos modelos de aprendizaje, una para el idioma español y otra para inglés, por lo que se requiere la creación u obtención de dos datasets que tengan las características mencionadas en la sección anterior tanto para tweets en español e inglés.

Debido a que existen una gran cantidad de trabajos e investigaciones respecto a la detección de bots en redes sociales (especialmente en Twitter y Facebook, por dar unos ejemplos) [63] [64] [65] [60], se ha tomado la decisión de buscar repositorios públicos u organizaciones que posean la información ya etiquetada y empezar posteriormente de manera directa con el procesamiento de las características ya definidas para el entrenamiento de las máquinas de aprendizaje.

#### 4.2.2. Diseño filtro análisis semántico

El objetivo de este filtro, como establece su nombre, es realizar un análisis semántico del texto del tweet. Para poder cumplir con este objetivo, se requiere hacer una revisión del nivel más bajo de NLP para poder llegar hasta el nivel de análisis semántico. En base a



esto, se puede dividir esta funcionalidad en dos secciones: el módulo Léxico-Sintáctico y el módulo Semántico.

Como se muestra en la Figura 4.4, el primer módulo se compone de tres procesos. En el bloque de Tokenización, se segmenta el texto en las palabras que lo componen, siendo la lista de tokens resultante de este proceso las palabras y la puntuación contenidas en éste. En este proceso, a diferencia del análisis léxico que realiza ya la extensión T-CREo, se realiza una “agregación” de palabras acortadas o abreviadas, que son naturalmente encontradas en las publicaciones debido al límite de caracteres que tiene Twitter.

Luego, en la Revisión Léxica y Sintáctica, se toma la lista de tokens y se realizan dos tareas importantes:

- Se incorpora a las palabras una etiqueta de Part-Of-Speech (POS) para identificar a qué componente del habla éstas corresponden. Las etiquetas posibles pueden ser: sustantivos, adjetivos, verbos, adverbios, etc.
- Se efectúa un análisis de la oración del texto, identificando la estructura de ésta y generando un árbol de análisis (parser tree) con ella.

Cabe mencionar, si bien ya existen filtros que realizan una tarea de análisis léxico dentro del modelo de credibilidad (ortografía y detección de malas palabras), el análisis léxico realizado en esta primera fase es para poder hacer principalmente un etiquetado del texto en base a sus componentes (sustantivos, verbos, etc.) y poder generar un posterior análisis sintáctico y semántico de manera más sencilla. Se considera un paso necesario para poder llegar de manera satisfactoria al objetivo de este proceso.

Tras realizar estas dos tareas, se pasa el texto a una revisión estructural de la oración, para verificar que ésta se encuentre escrita como corresponde respondiendo al lenguaje en el que están escritos, revisando que estén cumpliendo con las reglas gramaticales correspondientes (chequeo sintáctico).

Una vez realizada esta revisión estructural, se realiza la extracción de entidades en el texto (lugares, personas, etc.) para su posterior uso; otros componentes importantes para la siguiente fase son los sustantivos y verbos que puedan encontrarse dentro del texto. Se pasa luego al siguiente módulo, correspondiente al análisis semántico. Dependiendo si en la tarea de extracción realizada se han encontrado entidades o no, el módulo semántico puede realizar dos acciones:

- **Desambiguación y Contextualización:** Si en el texto se pudieron extraer entidades (lugares, organizaciones, personas, etc.), se prepara una consulta respecto a estas entidades para realizarla al repositorio de DBpedia y poder obtener una definición breve de la entidad y poder hacer una comparación de las palabras clave que contiene esta definición con las palabras que componen el texto del tweet. Adicionalmente, se hace una revisión de los verbos utilizados en el tweet para ver si su uso dentro del

contexto de las entidades es coherente, haciendo uso del repositorio Wordnet para obtener su definición y hacer la comparación respectiva.

- **Revisión Contextual:** En el caso de que el script no sea capaz de identificar entidades, se recurre a buscar palabras clave dentro del texto (como verbos, sustantivos, etc.) y buscar su definición dentro del repositorio Wordnet, haciendo el mismo proceso de comparación entre los resultados obtenidos del repositorio con el texto contenido en el tweet.

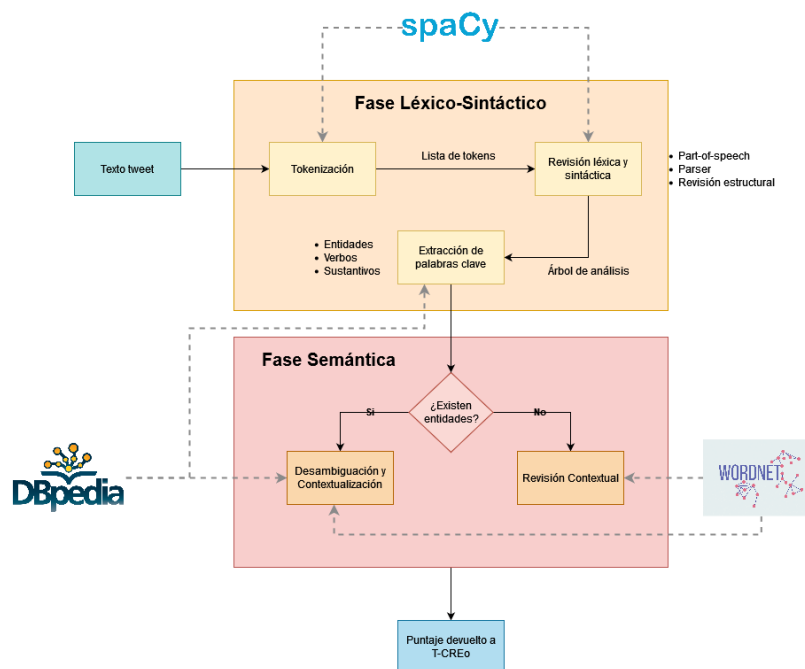


Figura 4.4: Diagrama de filtro de análisis semántico

Una vez es realizado uno de estos procesos, se hace el cálculo total de lo obtenido en la primera fase y la segunda, enviando el puntaje obtenido a la aplicación principal para ser incluida a la fórmula de credibilidad.

### 4.3. Diseño de Pruebas

En esta sección se presentarán las pruebas a realizar en este trabajo. Se hace una diferenciación en tres subsecciones: las pruebas en relación al funcionamiento del filtro de detección de bots, las pruebas en relación al funcionamiento de los componentes del filtro de análisis semántico y las pruebas para la revisión de la integración de las funcionalidades a la herramienta principal.

### 4.3.1. Pruebas de Funcionalidad: Detección de bots

- **Correcto funcionamiento de implementación:** Una sencilla prueba para verificar que los resultados obtenidos están dentro de los esperados. Aquí, los valores esperados que devuelva son las etiquetas que serán utilizadas para entrenar la máquina de aprendizaje (“human” y “bot”).
- **Correcta llamada desde Node JS:** Como la herramienta principal se encuentra implementada en Node JS, se requiere realizar una función en Typescript que permita invocar a la función desarrollada para realizar la detección de bots. Con esto se prueba que la funcionalidad desarrollada se puede integrar con la herramienta principal.
- **Comprobación de resultados:** Haciendo uso de Botometer<sup>1</sup>, se pretende hacer una selección de usuarios, un grupo que publique inglés y otro en español, para comparar resultados respecto a los clasificadores entrenados. Si bien Botometer no define si un usuario es o no un bot, da un porcentaje de posibilidad de que un usuario sea un bot. Con esto se busca evidenciar qué tan efectivo es la detección de bots desarrollada al compararlo con una herramienta ya existente.

### 4.3.2. Pruebas de Funcionalidad: Filtro Análisis Semántico

Debido a que la mayoría de las funcionalidades que se requieren para realizar este filtro requiere el uso de librerías de terceros, se realizarán pruebas para verificar que las herramientas elegidas funcionen de la manera requerida para realizar el adecuado funcionamiento del filtro.

- **Análisis de POS Tag:** El objetivo de esta prueba es poder visualizar y analizar si es que la librería seleccionada para realizar la tokenización y el etiquetado de las palabras lo hace de la manera esperada para ambos lenguajes. Se hará una revisión con otra herramienta para corroborar los resultados obtenidos, la librería CoreNLP de Java<sup>2</sup>.
- **Análisis de reconocimiento de entidades:** Al igual que la prueba anteriormente expuesta, esta prueba busca el poder analizar el alcance que tiene la librería para identificar entidades en el texto, tanto en el idioma inglés como el español. De la misma manera, los resultados serán comparados con otra herramienta que realiza la misma operación para comparar resultados, la librería CoreNLP de Java.
- **Consultas a DBpedia:** Debido a que se necesita realizar consultas al repositorio DBpedia para realizar el proceso de Contextualización y Desambiguación, se debe

---

<sup>1</sup><https://botometer.osome.iu.edu/>


<sup>2</sup><https://stanfordnlp.github.io/CoreNLP/>

verificar que las consultas realizadas a éste mismo sean válidas y la respuesta obtenida se encuentre en un formato que pueda ser trabajado fácilmente para el resto del proceso.

- **Consultas a Wordnet:** Al igual que el punto anterior, se debe validar que las consultas realizadas a Wordnet sean válidas y que las respuestas obtenidas de este repositorio sean en un formato que se pueda procesar posteriormente o, en su defecto, que se posible transformar para su manipulación.
- **Correcta llamada desde Node JS:** Una vez desarrollada esta funcionalidad, ésta debe ser utilizada por la herramienta principal, que se encuentra implementada en Node JS, por lo que es crucial que el filtro de análisis semántico pueda ser llamado a través de Node JS para poder aportar a la fórmula global.

### 4.3.3. Integración

Estas pruebas están diseñadas para emplearse una vez la fase de integración de las funcionalidades haya sido realizado, para comprobar que ambos filtros se encuentran operando de la manera esperada en la extensión.

- **Captación de datos:** El objetivo de esta prueba es verificar que los datos capturados como parámetros a ambos filtros (funciones en NodeJS) sean los correspondientes, al igual que el tipo de dato.
- **Procesamiento:** Esta prueba busca, a través de impresión en pantalla a través del servidor, verificar que el resultado del procesamiento de los datos sea el correcto y esté dentro de los resultados esperados.
- **Integración a fórmula global:** Esta prueba tiene como objetivo verificar y validar que los resultados obtenidos en ambos filtros sí están siendo considerados dentro de la fórmula global de la extensión/herramienta, corroborando el funcionamiento total de ambas funcionalidades dentro de ésta misma.
- **Tiempos de ejecución:** Esta última prueba consiste en verificar que el tiempo de ejecución de ambos filtros estén dentro del rango considerado "en tiempo real" (entre 1 a 10 segundos). Los resultados de las pruebas se mostrarán en milisegundos. 

# Capítulo 5

## Implementación

A continuación, se presentará la implementación de ambos filtros anteriormente descritos, además de las herramientas utilizadas para el completo desarrollo de éstas.

### 5.1. Herramientas utilizadas

#### 5.1.1. Software

El software por utilizar para el desarrollo de ambas funcionalidades es en su totalidad open source, aunque se requirió obtener permisos para algunos de ellos para poder ser utilizados. Se presentan a continuación todo el software a utilizar para el desarrollo:

- **Visual Studio Code v1.47.2:** IDE con soporte para un gran número de lenguajes de programación (Python, Typescript, Node.js, C++, Java, etc.). Utilizado para desarrollar el filtro de detección de bots y el de análisis semántico.
- **Google Drive:** Servicio de almacenamiento y sincronización de archivos. Se utiliza para respaldar el desarrollo hecho hasta el momento en caso de emergencia, como también de los materiales de investigación analizados.
- **Twitter API:** Permite la lectura y escritura de data de Twitter. Permite la composición de tweets, leer perfiles y acceder a un gran volumen de tweets de un tema particular en lugares específicos. Las claves recibidas para utilizar esta API se utilizan para el desarrollo de ambos filtros y la prueba de ellos.
- **DBpedia:** Es un proyecto de esfuerzo comunitario para extraer contenido estructurado de la información creada en varios proyectos Wikimedia. Esta información estructurada asemeja a un gráfico de conocimiento. Se utilizará la API en Python de

DBpedia para poder realizar la segunda fase del análisis semántico, para la desambiguación y contextualización del sujeto, objeto y entidades, además de la revisión contextual.

- **WordNet:** Es una base de datos léxica en inglés. Sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets), cada uno de ellos expresando un concepto distinto. Se utilizará la API en Python de WordNet para poder realizar la segunda fase del análisis semántico, para la desambiguación y contextualización del predicado.
- **Microsoft Excel:** Herramienta de hoja de cálculo. Este software es utilizado principalmente en la edición, creación y almacenamiento de datasets requeridos para la fase de entrenamiento del filtro de detección de bots.
- **Postman:** Cliente API que facilita el proceso de crear, probar y documentar API, permitiendo crear y guardar solicitudes HTTP, tanto simples como complejas. Este software en particular se utilizará para la recopilación de resultados de las pruebas diseñadas y presentadas del capítulo 4.

### 5.1.2. Lenguajes de programación

El lenguaje elegido para el desarrollo de ambos filtros es Python, la versión 3.9.1 más específicamente. Python es un lenguaje de programación interpretado de alto nivel y de propósito general, además de ser multiparadigma. La principal razón para elegir este lenguaje fue por la gran cantidad de librerías que se encuentran disponibles para su uso, sobre todo enfocadas en el procesamiento de lenguaje natural y algoritmos de aprendizaje. Como segunda razón de su elección, se tiene la simpleza con la que se pueden manejar los datos, especialmente en el caso de strings y listas o arreglos. A continuación, se presentan las librerías utilizadas hasta el momento en el desarrollo:

- **spaCy:** Es una librería gratuita open source para el procesamiento avanzado de lenguaje natural (NLP) en Python. Se utiliza principalmente en el análisis semántico para la tokenización, etiquetado de palabras (part-of-speech) y la estructura sintáctica de las oraciones. Se consideró como la principal herramienta para el filtro de análisis semántico debido a que tiene modelos tanto para el idioma inglés como para español.
- **PyWavelets:** PyWavelets es una librería open source de software para el cálculo de transformadas wavelet en Python. Se utiliza para el cálculo de la transformada discreta wavelet requerida para el cálculo de las características necesarias para el entrenamiento del modelo de clasificación.
- **langid:** Librería para un procesamiento fácil para la detección de lenguajes en el texto.

- **nltk:** Librería para trabajar con lenguaje natural en Python. Proporciona interfaces simples para la utilización de más de 50 recursos de corpus y léxicos como WordNet. Se utiliza exclusivamente para limpiar el tweet, removiendo las palabras que no aportan al contenido de éste.
- **collections:** Librería o módulo de Python que provee estructuras de datos adicionales a las ya integradas en Python. Utilizada de manera exclusiva para el cálculo de las características del tweet necesarias para el entrenamiento del modelo de clasificación.
- **urllib.parse:** Librería para el manejo de URLs en Python, específicamente encargándose del procesamiento de la URL. Se utiliza para obtener el nombre de la fuente de la URL contenida en el texto, una de las características utilizadas para entrenar el modelo de clasificación.
- **sklearn:** Librería para máquinas de aprendizaje y para modelamiento estadístico. En este trabajo, se utiliza exclusivamente para la creación, entrenamiento y análisis de métrica de un grupo de máquinas de aprendizaje para posteriormente elegir entre éstas la de mejor rendimiento para utilizar en el filtro de detección de bots.
- **tweepy:** Librería para ingresar de manera fácil a la API de Twitter.
- **pandas:** Librería especializada para la manipulación y análisis de estructuras de datos. En este trabajo se utiliza principalmente para el cálculo y preparación de información requerida en el filtro de detección de bots.
- **SPARQLWrapper:** Librería de Python que trabaja como un contenedor alrededor de un servicio SPARQL para ejecutar consultas de forma remota, ayudando a crear consultas y convertir el resultado obtenido a un formato más simple de manipular (JSON).
- **dbpedia.spotlight:** Librería para anotar automáticamente las menciones de los recursos de DBpedia en un texto. Esta herramienta se utiliza para identificar de manera rápida las entidades en un texto del tweet que existe en la colección de datos de DBpedia.

## 5.2. Filtro Detección de Bots

En esta sección se presentará con detalle el trabajo realizado en el desarrollo del filtro de detección de bots, desde la obtención y preparación de los datasets de entrenamiento hasta el script en Python que permite realizar el proceso de detección de bots respecto al usuario del tweet analizado.

### 5.2.1. Clasificadores

En los inicios de este trabajo, la intención inicial había sido entrenar máquinas de aprendizaje que fueran capaces de detectar si un usuario es bot o no y, en el caso que lo fuera, si este bot se trataba de uno inofensivo (bots de utilidades, de entretenimiento, etc.) o peligroso/dañino (fraudes, robo de identidad, phishing, etc.). Si bien se pudo encontrar un trabajo que particularmente trabajó con esta misma idea [66], el dataset que se pudo obtener tras pedir permiso de trabajar con el dataset utilizado, el tamaño de datos etiquetados que este tenía era muy pequeño (no más de 200 etiquetados por cada categoría de bot presentados en el trabajo), además que la mayoría de los usuarios registrados en este dataset ya no se encontraban habilitados en Twitter y que se presentaban solamente tweets en idioma inglés.

Debido a esto, se decidió restringir el funcionamiento de este filtro a una clasificación binaria (bot o humano) y buscar un dataset con una gran cantidad de data etiquetada para el entrenamiento de las máquinas aprendizaje. Finalmente, se logró obtener permiso de ocupar el dataset del trabajo de *“The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race”* [8], a través de la página MIB <sup>1</sup>, obteniendo así el dataset con la siguiente información:

Dataset	Descripción	Cuentas	Tweets	Year
Cuentas genuinas	cuentas verificadas que son operadas por humanos	3.474	8.377.522	2011
Social spambots #1	retweeters de un candidato político italiano	991	1.048.575	2012
Social spambots #2	spammers de aplicaciones de pago para dispositivos móviles	3.457	428.542	2014
Social spambots #3	spammers de productos en venta en Amazon.com	464	1.048.575	2011
Traditional spambots #1	conjunto de entrenamiento de spammers	1.000	145.094	2009
Traditional spambots #2	spammers de URLs fraudulentas	100	74.957	2014
Traditional spambots #3	cuentas automatizadas que envían spam a las ofertas de empleo	433	5.794.931	2013
Traditional spambots #4	otro grupo de cuentas automatizadas que hacen spam de ofertas de trabajo	1.128	133.311	2009
Fake followers	simples cuentas que inflan el número de seguidores de otra cuenta	3.351	196.027	2012

Tabla 5.1: Descripción de dataset [8] obtenido a través de MIB.

#### 5.2.1.1. Datasets de entrenamiento

Posteriormente se realizó un análisis de los datos obtenidos, teniendo que hacer un preprocesamiento de la información, al haber varios tweets duplicados, con errores de Unicode, además de querer limpiar el texto de menciones, hashtags y de enlaces URL. Todos los tweets del dataset fueron limpiados y procesados de tal forma de quedar con el texto limpio, además de asociar a cada tweet con su usuario y las características correspondientes a cada uno (cantidades de estados, número de seguidores, número de amigos, número de favoritos, número de listas). Los grupos de datos que no fueron utilizados dentro de este procedimiento fueron *“fake followers”*, *“Traditional spambots #4”*, *“Traditional spambots*

<sup>1</sup><http://mib.projects.iit.cnr.it/dataset.html>



#3”, “*Traditional spambots #2*” que se presentan en la Tabla 5.2.1, al no contar con los tweets asociados a los usuarios entregados en esos conjuntos.

Luego de hacer el trabajo de preprocesamiento, se necesitó realizar un script en python con la librería “*langid*” para poder identificar de manera masiva el lenguaje que tenía cada tweet, para luego separar los tweets en idioma español e inglés del resto de los idiomas incorporados en el dataset (italiano, sueco, tagalog, etc.).

Tras hacer las separaciones correspondientes haciendo uso del script hecho en Python y de Microsoft Excel, se obtuvieron los siguientes datasets, considerando también las características que fueron procesadas para cada dato, que fueron ya explicadas en detalle en el Capítulo 4, en la sección 4.2.1:

	Entrenamiento		Validación	
	Español	Inglés	Español	Inglés
Cantidad de datos	14650	26000	1992	2000
Características	14	14	14	14
Datos etiquetados como “bot”	7324	13000	1000	1000
Datos etiquetados como “human”	7326	13000	992	1000

Tabla 5.2: Datasets de entrenamiento y validación en español e inglés

Como puede observarse en la descripción de ambos datasets, el creado como dataset de entrenamiento en inglés tiene un tamaño considerablemente mayor que el de español. Esto es principalmente porque el dataset en español fue creado en base a la cantidad de tweets hechos por bots que pudieron ser rescatados luego de la fase de preprocesamiento. Si bien existía una mayor cantidad de tweets en español de usuarios genuinos (humanos), se optó por tomar aproximadamente la misma cantidad de tweets de bots para poder mantener el dataset de entrenamiento lo más equilibrado posible. Adicionalmente, otros trabajos revisados para el proceso de entrenamiento presentan buenos resultados incluso con una menor cantidad de tweets para su entrenamiento [59]

Además, a partir de los datos no utilizados para la creación de los datasets de entrenamiento, se utilizaron para conformar dos datasets de prueba para cada lenguaje, que son detallados de igual manera en la Tabla 5.2

#### 5.2.1.2. Fase entrenamiento y análisis

Una vez se obtuvieron ambos datasets de entrenamiento, se procedió a entrenar máquinas de aprendizaje para analizar cuál de ellas sería la más indicada para utilizar como clasificador. Para esto, se utilizó la librería de Python “*sklearn*” para crear un script para entrenar, probar y visualizar los resultados obtenidos por cada máquina. Se utilizaron los siguientes algoritmos de aprendizaje para probar su rendimiento, tanto con el dataset de entrenamiento en inglés y español:

- **AdaBoost classifier:** Se trata de un meta-estimador que comienza ajustando un clasificador al conjunto de datos inicial y, a continuación, ajusta copias adicionales del clasificador en el mismo conjunto de datos, modificando las ponderaciones de las instancias que fueron clasificadas erróneamente, de manera que los futuros clasificadores se concentren más en las situaciones difíciles [67, 68].
- **Bagging classifier:** Se trata de un metaestimador de conjunto que aplica clasificadores básicos a diferentes subconjuntos aleatorios del conjunto de datos inicial y, a continuación, combina cada predicción (ya sea por votación o por promedio) para obtener la predicción final. Al añadir la aleatoriedad al proceso de construcción de un estimador de caja negra (como un árbol de decisión), un metaestimador de este tipo puede utilizarse a menudo para reducir la varianza del estimador [69].
- **Decision Tree classifier:** Son una técnica de aprendizaje supervisado no paramétrico que se utiliza para la clasificación. El objetivo es aprender reglas de decisión directas derivadas de las características de los datos para construir un modelo que prediga el valor de una variable objetivo. Es una clase capaz de realizar una clasificación multiclase sobre un conjunto de datos [70].
- **Logistic Regression:** En lugar de un modelo de regresión, éste es de clasificación. En la literatura, la regresión logit, la clasificación de máxima entropía (MaxEnt) y el clasificador log-lineal también se utilizan para referirse a la regresión logística. En este modelo, se utiliza una función logística para simular las probabilidades que describen los posibles resultados de un único experimento [71].
- **Random Forest classifier:** Se trata de un metaestimador que emplea el promedio para aumentar la precisión de la predicción y reducir el exceso de ajuste después de ajustar numerosos clasificadores de árboles de decisión a diferentes submuestras del conjunto de datos [72].

Este conjunto de algoritmos fue seleccionado por ser, en su mayoría, utilizados en una gran variedad de trabajos con buenos resultados [58] [73] (Decision Tree, Random Forest, Adaboost) y por la decisión de considerar el problema de detección de bots como una clasificación binaria. El procedimiento realizado para entrenar y testear cada uno de los algoritmos presentados fue el siguiente:

1. Cargar el dataset de entrenamiento y se divide en un 80 % para el entrenamiento como tal, quedando el restante 20 % como testeo posterior.
2. Posteriormente, cargar el dataset de prueba que corresponde al mismo idioma que el de entrenamiento previamente cargado.

3. Elegir e instanciar el modelo que se quiere entrenar, ajustando los parámetros que corresponden a cada algoritmo.
4. Realizar el entrenamiento de la máquina y la prueba con el dataset de entrenamiento, dando el resultado teórico (con las métricas de recall, F-1 y precisión).
5. Con la librería de Python “*Pickle*”, guardar el algoritmo entrenado en un archivo extensión “.sav”.
6. Cargar nuevamente el algoritmo de aprendizaje entrando, haciendo uso de la librería “*Pickle*”, para hacer que haga una predicción del dataset de prueba cargado anteriormente, dando los resultados de prueba (con las métricas de recall, F-1 y precisión).

Este procedimiento se realizó varias veces por algoritmo (un mínimo de 8 veces por cada uno, para tener un estándar), explorando diferentes valores en los parámetros de cada uno de ellos para encontrar el ajuste que lograra los mejores resultados. Al finalizar todo el proceso, se hizo una recopilación de todos los datos obtenidos tanto en inglés como en español.

Clasificador	Entrenamiento				Validación			
	Precision	Recall	F1-score	Accuracy	Precisión	Recall	F1-score	Accuracy
Decision tree	1.00000	0.99651	0.99825	0.99827	0.66221	0.99000	0.79359	0.74250
Random Forest	1.00000	0.99884	0.99942	0.99942	0.80178	0.99100	0.88640	0.87300
Bagging	1.00000	1.00000	1.00000	1.00000	0.70035	0.99800	0.82309	0.78550
Adaboost	0.99768	0.99845	0.99806	0.99808	0.52243	0.99000	0.68394	0.54250
Logistic Regression	0.95807	0.86783	0.91072	0.91558	0.76761	0.62100	0.68657	0.71650

Tabla 5.3: Mejores resultados de clasificador: dataset en inglés

Classifier	Entrenamiento				Validación			
	Precision	Recall	F1-score	Accuracy	Precisión	Recall	F1-score	Accuracy
Decision tree	0.99119	0.99389	0.99254	0.99249	0.98897	0.99395	0.99145	0.99147
Random Forest	0.95958	1.00000	0.97937	0.97884	0.99799	1.00000	0.99899	0.99900
Bagging	1.00000	1.00000	1.00000	1.00000	0.90100	1.00000	0.94792	0.94528
Adaboost	0.99796	0.99864	0.99830	0.99829	1.00000	0.99899	0.99950	0.99950
Logistic Regression	0.95042	0.91168	0.93065	0.93174	1.00000	0.87802	0.93505	0.93926

Tabla 5.4: Mejores resultados de clasificador: dataset en español

Como se puede ver en las Tablas 5.3 y 5.4, la mayoría de los resultados obtenidos en la sección “Validación” están por encima del 70 % en cuanto a precisión. La precisión en ambas tablas es también considerablemente buena, con valores superiores al 70 % en la mayoría de los casos. Para ambos casos, inglés y español, el mejor algoritmo es el RandomForest por su capacidad de generalización, es decir, la precisión en la predicción de nuevos casos que es del 87 % para el inglés y del 99 % para el español.

Aunque el entrenamiento con Bagging y Adaboost resultó con métricas altas, RandomForest es más adecuado que el bagging porque cada árbol sólo aprende de un pequeño

subconjunto de características, la selección aleatoria de características hace que los árboles sean más independientes unos de otros que el bagging regular, lo que a menudo produce un mejor rendimiento predictivo (debido a una mejor compensación de varianza y sesgo). Esto hace que la selección aleatoria de características sea el proceso más rápido que el bagging. Adaboost también es más sensible al sobreajuste que Random Forest [74]. Los métodos de "bagging", como el bagging y los bosques aleatorios, están diseñados para simplificar los modelos que se ajustan en exceso a los datos de entrenamiento. El "boosting", por otro lado, es una estrategia para aumentar la complejidad de los modelos que tienen un fuerte sesgo, o modelos que se ajustan mal a los datos de entrenamiento.

### 5.2.2. Script para detección de bots

Una vez realizada la selección de la máquina de aprendizaje, se pasa al desarrollo del script que utilizará el archivo ".sav" donde se encuentra la máquina para poder realizar la detección de bots. Por comodidad, se hizo una petición a Twitter para poder utilizar el API de Twitter y poder obtener una clave de uso personal<sup>2</sup> para hacer las consultas necesarias y las posteriores pruebas una vez terminado el desarrollo.

A continuación, se presenta el script desarrollado para la detección de bots:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

user = api.get_user(screen_name='@'+sys.argv[1])
if not user:
    print("user not found")

tweet_id = []
status = api.user_timeline(screen_name=user.screen_name, count=5, include_rts = False)
```

Figura 5.1: Extracto de script de detección de bots

Para poder ingresar de manera sencilla al API de Twitter, se utilizó la librería *tweepy* de Python, colocando las credenciales entregadas por Twitter para ingresar a la API.

Se captura el "screen\_name" del usuario del tweet a analizar, información que llegará como un parámetro externo al script. En el caso de que no se encuentre el usuario que se ingresó como parámetro, el script imprimirá un mensaje indicando lo ocurrido.

---

<sup>2</sup><https://developer.twitter.com/en>

```

for text in status:
    tweet_id.append(text.id_str)

tweet = []

for id_text in tweet_id:
    temp = api.get_status(id_text, tweet_mode= "extended")
    #print(temp)
    tags = detect_hashtag(temp.full_text)
    url = detect_url(temp.full_text)
    mention = detect_mention(temp.full_text)
    total = tags + url + mention
    if len(temp.full_text.split()) > total:
        tweet.append(temp.retweet_count)
        tweet.append(temp.favorite_count)
        tweet.append(temp.full_text)
        break

```

Figura 5.2: Extracto de script de detección de bots

En la siguiente línea, se obtienen los cinco tweets más recientes del usuario en cuestión, colocando el modo “*extended*” para poder capturar el texto completo del tweet e indicando al API que no se consideren los retweets del usuario en esta captura de información. Cabe mencionar que este paso se realiza principalmente para poder obtener un tweet con texto que pueda ser procesado para obtener el resto de las características que se describieron anteriormente. En caso de que el primer tweet obtenido para procesar contenga sólo URLs, hashtags o menciones, esta función permite hacer un reemplazo de este tweet para poder proseguir con la trayectoria normal del proceso.

Luego se hace obtienen las características de cada uno de los tweets (cantidad de hashtags, URLs y menciones) que tienen dentro del texto y ver si las cantidades de estos son las mismas cantidades de palabras que hay dentro del texto. Esto es para corroborar que el tweet que se esté utilizando para poder detectar si el usuario es o no bot (no es necesariamente el mismo que se está analizando) contenga texto además de las características mencionadas, para poder luego procesar el resto de las características que lo requieren.

```

tweet_lang = lang_detection(tweet[2])
data = get_features(user, tweet, tweet_lang)

result = prediction(data, tweet_lang)

print(str(result[0]))

```

Figura 5.3: Extracto de script de detección de bots

Una vez se encuentra un tweet que cumple con estas características, se identifica el idioma en el que está escrito, luego se le calculan las características requeridas por la máquina de aprendizaje, que es ordenado y estructurado utilizando la librería *pandas*, y luego se hace la predicción. El resultado de la predicción es impreso por el script, que luego será captado por Node JS en la extensión.

Cabe mencionar que tanto la función de calcular las características como la función para la predicción están sujetas al idioma en el que está escrito el tweet, por lo que es con ese valor que es obtenido en la línea 157 donde se toma la decisión en estas funciones de cómo manipular el contenido del tweet.

Para poder integrar esta funcionalidad a la extensión, se requiere de una función en Node JS que pueda invocar el script de Python para poder capturar el resultado que éste mismo entrega. La función en cuestión es presentada a continuación en la Figura 5.4

```

const road = require('path')
const proc = require('child_process')
const namefile = road.join(__dirname, 'predictUser', 'bot_detection_module.py')
function predictUser(user: String) : String{
  var test = proc.spawnSync('python', [namefile, user])

  return test.stdout.toString().trim()
}

```

Figura 5.4: Integración de script Python a Node JS

Para poder ejecutar el script de Python en segundo plano, se requirió utilizar la función “*spawnSync*”, principalmente por necesidad de mantener los tipos de las funciones que se encargan de hacer el cálculo de credibilidad, ya que estos no trabajan en base a promesas. En resumen, las promesas (o “promises”, en inglés) corresponden a un objeto que representa la finalización exitosa o fallida de una operación asíncrona [75]. Debido a que la función encargada de hacer el cálculo de credibilidad del usuario es una función síncrona (al utilizar datos de entrada que se extraen casi instantáneamente del usuario), le función para llamar a la detección de bots requiere mantener esta función con ese tipo.

Una vez realizada esta función para unificar el filtro a la extensión, se debe hacer la conexión con esta misma para poder integrar el filtro como aporte a la fórmula global, haciéndose de la siguiente forma:

```
function calculateUserCredibility(user: TwitterUser) : number {
  const weightedScore = getVerifWeight(user.verified) + getCreationWeight(user.yearJoined)
  if (predictUser(user.username) == 'bot') {
    if (weightedScore > 50) {
      return weightedScore * 0.85
    } else if (weightedScore > 35) {
      return weightedScore * 0.75
    } else {
      return weightedScore * 0
    }
  } else {
    return weightedScore
  }
}
```

Figura 5.5: Inclusión de función que llama a script de Python

Como se puede observar en la Figura 5.5, se modificó la función de credibilidad de usuario en Node JS para incorpora el script presentado en 5.4. El flujo es el siguiente: una vez se hace la predicción, si el resultado corresponde a un bot se hace una evaluación del puntaje obtenido por las otras métricas que componen la credibilidad de usuario. Si la suma de ambas variables es mayor que 50, se penaliza la credibilidad del usuario en un 15 %, en el caso de que sea menor que 35 se penaliza la credibilidad en un 25 %. Si la credibilidad es menor que eso, entonces el resultado de la credibilidad de usuario corresponde a 0. En el caso de que el usuario sea clasificado como “humano”, entonces directamente se retornará el valor de la suma de los resultados obtenidos por los otros dos filtros.

## 5.3. Filtro Análisis Semántico

En esta sección se presentará con detalle el trabajo realizado en el desarrollo del filtro de análisis semántico, explicando con detalle las funcionalidades utilizadas en cada fase que compone a este filtro.

### 5.3.1. Fase Léxica-Sintáctica

En esta primera fase, como se comentó en el capítulo 4 en la sección 4.2.2, se debe primero hacer un análisis del nivel más bajo de análisis de texto, siendo éste el análisis léxico y, posteriormente, el sintáctico.

Como es común en los trabajos de NLP, se requiere primero hacer un preprocesamiento al texto de llegada, referenciado en la línea 14 de la Figura 5.6, donde la función “*clean\_text*” hace una limpieza del texto, quitando URLs, hashtags, menciones, emoticones o errores de Unicode que puedan haberse producido en el texto del tweet.

Posterior a esto, se agregó un proceso antes de pasar a la tokenización del texto, tomando la información para primero "agregar" o "extender" las palabras acortadas utilizadas en el tweet, que se ve representado en la Figura 5.6 en la línea 17. Para lograr esto, se buscó las abreviaciones más comunes tanto en español ("mñn", "pq") como inglés ("idk", "lol", "tbh"). Haciendo un análisis de alrededor unos 30 tweets por cada idioma, se logró recopilar y conformar dos archivos que permiten extender las palabras abreviadas dentro del texto. Si bien no es una base de datos extensa, permite mejorar la posibilidad de realizar un análisis con menos "ruido" para procesar en la fase de contextualización o de revisión contextual.

```

14 text = clean_text(sys.argv[1])
15 score = []
16 if get_lang(text) == 'en':
17     text = dct.add_text(text)
18     score.append(check_eng(text))
19     ents = get_entities_eng(text)

```

Figura 5.6: Código respectivo a la fase léxica-sintáctica

Luego, en la función "*check\_eng*" de la línea 18, se hace una revisión del texto haciendo uso de un diccionario, tanto para revisar gramática como ortografía. Cabe mencionar que, para cada lenguaje, se definió un par de reglas de gramática que deben ser ignoradas, principalmente por la naturaleza informal de la plataforma Twitter, evitando generar "penalizaciones" por reglas gramaticales muy estrictas (por ejemplo, no escribir ambos signos de exclamación o de interrogación en el caso de español; ocupar "wanna" en vez de "want" o no capitalizar el pronombre en primera persona "I" en inglés).

Una vez se calcule el puntaje total del texto respecto a los errores válidos encontrados, dado por la siguiente fórmula:

$$1 - \frac{\text{errores}}{\text{tamano\_texto}} \quad (5.1)$$

éstos se guardan en el arreglo "**score**", donde se guardan todos los puntajes calculados del filtro semántico.

Una vez acabado este análisis, se procede a hacer la extracción de entidades del texto, como puede ser observado en la línea 19 de la Figura 5.6. La función mostrada en esta línea de código realiza 2 tareas: (1) realiza la tokenización del texto y (2) extrae las entidades encontradas en ella, en el caso de encontrarlas.

Para realizar la extracción de entidades, se utilizan ambas librerías *SPARQLWrapper* y *dbpedia\_spotlight*, la primera para hacer las consultas a DBpedia y la segunda para poder relacionar con mayor facilidad una entidad con una ya existente en el repositorio, obteniendo inmediatamente la URL a la que se debe realizar la consulta.



A continuación, en la Figura 5.7, se muestra la función encargada de hacer la consulta a DBpedia para poder extraer la información respectiva de la entidad identificada al utilizar *dbpedia\_spotlight*. Se puede ver que la estructura de consulta se mantiene a excepción por la variable “query” que se encuentra en azul, que es donde se hace el cambio a la URL identificada. Luego la respuesta recibida se convierte a JSON para poder hacer más fácil la extracción de la información y se extrae sólo el campo “comment”, donde se obtiene la definición de la entidad y sus características más relevantes.

```
def extract_eng(query):
    sparql = SPARQLWrapper("http://dbpedia.org/sparql")
    sparql.setQuery("""
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        SELECT ?comment
        WHERE { """+query+"" rdfs:comment ?comment
        FILTER (LANG(?comment)='en')
        }
        """)
    sparql.setReturnFormat(JSON)
    result = sparql.query().convert()

    for hit in result["results"]["bindings"]:
        return(hit["comment"]["value"])
```

Figura 5.7: Método para consultar a DBpedia por Python

Como se puede ver a continuación, en la Figura 5.8, la función “*get\_entities*” está estructurada de tal forma para poder identificar todas las entidades de un texto y poder hacer la consulta a través de la función presentada en la Figura 5.7, retornando un arreglo con toda la información obtenida, que será utilizada para la siguiente fase del filtro.

```
def get_entities_eng(text):
    uris = spot_dbpedia_eng(text)
    info = []
    for item in uris:
        info.append(extract_eng(item).encode('unicode_escape').decode('unicode_escape'))
    return info
```

Figura 5.8: Estructura para extraer entidades

Cabe mencionar que, antes que cualquiera de estas funciones de agregar o revisar texto sean efectuadas, se requiere saber de antemano cuál es el idioma en el que está escrito el texto a analizar. Éste parámetro es recibido de la misma extensión, a través de la función *getTweetInfo()*, para no volver a ser calculada en el script. Si bien en la Figura 5.6 se presenta el caso respectivo del idioma inglés, el proceso es exactamente idéntico al español.

### 5.3.2. Fase Semántica

Como se planteó en la Figura 4.4, existen dos posibilidades al momento de extraer las entidades: (1) que no se identifique alguna entidad dentro del tweet o (2) que se logre extraer una entidad o más. Dependiendo de la existencia o no de estas entidades dentro del texto del tweet, los scripts utilizados en la fase semántica cambian para poder adaptarse a la situación.

```
if ents != []:
    from semantic_similarity_en import calculate_simil_eng, eval_verb_EN
    score.append(calculate_simil_eng(text, ents))
    score.append(eval_verb_EN(text, ents))
else:
    from seman_eng import get_score_EN
    score.append(get_score_EN(text))

result = sum(score)

punt = result/len(score)
print(punt)
```

Figura 5.9: Código respectivo a la fase semántica

En caso de que se logren identificar por lo menos una entidad, se cargan los scripts correspondientes al proceso de Desambiguación y Contextualización, donde la información consultada sobre la entidad a DBpedia es procesada, eliminando las “stop words”, y es contrastada con las palabras importantes del tweet (sustantivos, adjetivos) haciendo uso de la función de similitud semántica proporcionada por la librería spaCy, la que devuelve un valor en el rango [0,1], siendo el valor más cercano a uno si las palabras son similares entre ellas, basándose en los vectores a los que la librería los transforma para hacer la comparación, pudiendo ver así evaluar qué tan “concordante” es la información entregada en el texto con las entidades que menciona. Una vez se hacen todas las comparaciones de similitud semántica, los resultados se suman y se saca un promedio que es guardado en el arreglo “score”. Se hace este mismo procedimiento luego, pero exclusivamente entre las entidades y los verbos encontrados dentro del texto, que luego son guardados en el arreglo anteriormente mencionado.

En el posible escenario en que el filtro no logre extraer entidades, se carga el script correspondiente al proceso de Revisión Contextual, donde se realiza un proceso parecido al descrito, esta vez tomando en consideración los verbos, adjetivos y otros componentes que no sean considerados como “stop words”, para ser analizados con la función de similitud semántica. Al utilizarse Wordnet como repositorio para extraer las definiciones de los componentes en esta fase, se emplea una función que elige la definición que más se acomoda al contexto, basándose también en la similitud semántica para hacer la elección. Al igual que el proceso anterior, se suman los valores obtenidos y se saca el promedio de la similitud

semántica obtenida, para ser guardada en el arreglo.

Una vez se ha pasado por el análisis léxico-sintáctico y el semántico, se suman todos los puntajes obtenidos en el arreglo y se saca el promedio de éste, resultando en el puntaje total obtenido del filtro por el tweet analizado. Para integrar esta funcionalidad a la extensión, se hizo lo siguiente:

```
const path = require('path')
const scriptFilename = path.join(__dirname, 'semantic', 'analisis_seman.py')
import { spawn } from 'child_process'

async function semanticScore(tweet_text: String): Promise<number>{
  return new Promise((resolve, reject) => {
    const test = spawn('python', [scriptFilename, tweet_text])
    try {
      test.on("close", (data:any) => {
        return resolve(data)
      })
      test.on("error", (err:any) => {
        return reject(err)
      })
    } catch(error){
    }
  })
}
```

Figura 5.10: Integración de script Python a Node JS

En la Figura 5.10 se muestra una función asincrónica de Node para poder invocar al script de Python anteriormente mencionado. Para poder hacer la llamada de manera asincrónica y poder colocar los parámetros necesarios para el procesamiento del filtro, se utiliza el proceso secundario “*spawn*”. Con esto se logra captar los resultados obtenidos desde el script de Python y poder utilizarlos en Node JS para el procesamiento final en la extensión.

A continuación, se muestra el lugar exacto donde la función de la Figura 5.10 es incorporado para que el resultado del filtro sea parte de la fórmula global de credibilidad:

```
async function calculateTextCredibility(text: Text, params: TextCredibilityWeights) : Promise<Credibility> {
  const badWordsCalculation = params.weightBadWords * badWordsCriteria(text.text)
  const spamCalculation = params.weightSpam * spamCriteria(text)
  const missSpellingCalculation = params.weightMisspelling * (await missSpellingCriteria(text))
  const semanticCalculation = params.weightSemantic * [(await semanticScore((text.text))*100)]
  return {
    credibility: badWordsCalculation + spamCalculation + missSpellingCalculation + semanticCalculation
  }
}
```

Figura 5.11: Inclusión de función que llama a script de Python

# Capítulo 6

## Pruebas

En este capítulo se mostrarán los resultados obtenidos tras realizar las pruebas descritas en el Capítulo 4, respecto al funcionamiento por separado de ambos filtros (y sus componentes), y luego una vez están integrados en la extensión.

### 6.1. Detección de Bots

#### 6.1.1. Funcionamiento de Implementación

Primero que nada, se debe tener la certeza que el clasificador efectivamente devuelve las predicciones que se consideraron al inicio de su diseño, siendo éstas “human”, en caso de clasificar al usuario como una persona genuina, y “bot” en caso de tratarse de una cuenta automatizada.

Para verificar que efectivamente se obtienen estas dos predicciones en el script de Python, se ejecutó el script imprimiendo en pantalla los resultados obtenidos al ingresar el nombre de usuario de dos cuentas arbitrarias:

```
PS E:\Tesis\BackEnd-Integration> e:/Tesis/BackEnd-Integration/src/calculator/predictUser/bot_detection_module.py  
human  
PS E:\Tesis\BackEnd-Integration> e:/Tesis/BackEnd-Integration/src/calculator/predictUser/bot_detection_module.py  
bot
```

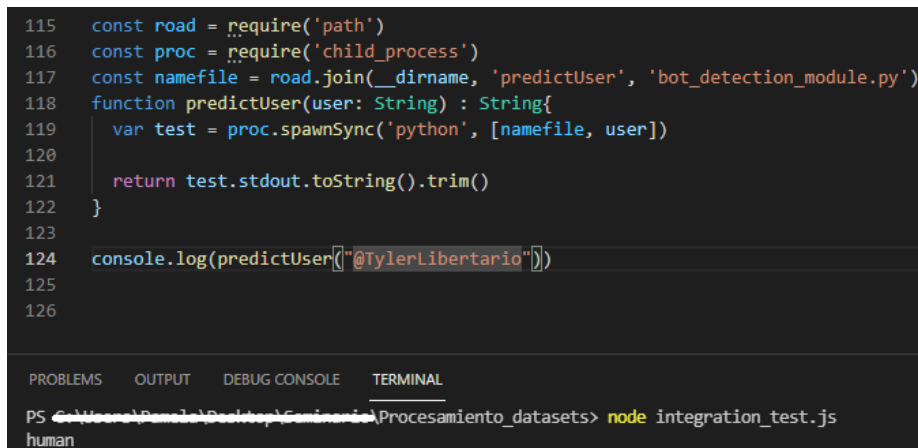
Figura 6.1: Resultado de predicción en consola de Visual Studio Code

En la figura 6.1 se puede observar, el primer resultado del script, reconociendo el primer usuario que fue analizado como “human”, un usuario genuino. De la misma forma, el siguiente fue analizado y el clasificador obtuvo como resultado que el usuario correspondía a “bot”.

Con esta simple prueba ejecutada en la misma consola, podemos asegurar que efectivamente el script, al realizar las predicciones, devuelve efectivamente los resultados esperados respecto a su diseño. Este proceso se repitió varias veces para validar que el comportamiento del clasificador fuera el esperado y que siempre devolviera la misma predicción al ingresar los mismos datos.

### 6.1.2. Llamada desde Node JS

En el capítulo anterior, en la Figura 5.4, se mostró la función en Node JS que se encarga de invocar en segundo plano el script de Python para permitir la captura del resultado de éste y ser utilizado en la extensión posteriormente. Ya que este proceso es imprescindible que funcione correctamente, se realizó la prueba para poder verificar que efectivamente se logra capturar de manera correcta el resultado del script de Python, haciendo efectiva la integración con la extensión.



```
115 const road = require('path')
116 const proc = require('child_process')
117 const namefile = road.join(__dirname, 'predictUser', 'bot_detection_module.py')
118 function predictUser(user: String) : String{
119     var test = proc.spawnSync('python', [namefile, user])
120
121     return test.stdout.toString().trim()
122 }
123
124 console.log(predictUser("@TylerLibertario"))
125
126
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Borja\Desktop\Procesamiento\_datosets> node integration\_test.js  
human

Figura 6.2: Resultado al invocar script Python usando Node JS

Como se puede ver en la Figura 6.2, se hace una llamada a la función “*predictUser*” y se indica que imprima en consola para verificar qué es lo que capta la función de Node. Abajo, en la pestaña del terminal, se puede observar que momentos después de ejecutar el código de predicción de usuario, se muestra el resultado “human”, corroborando así que la llamada al script de Python a través de la función de Node JS funciona efectivamente de manera correcta, permitiendo así la integración del filtro. Similar al caso anterior, se realizó este proceso con varias cuentas de Twitter aleatorias para corroborar que el comportamiento del script era el esperado.

### 6.1.3. Comprobación de resultados con herramienta externa

Para poder hacer un análisis de qué tan efectivo son los clasificadores utilizados en la detección de bots, se hizo la siguiente simple prueba: se eligió un grupo de 10 cuentas de manera arbitraria, tanto en inglés como español, para poder calcular la probabilidad de que fueran bots usando el programa Botometer y también utilizarlos en los clasificadores creados para ver su predicción. A continuación, se presenta una tabla con los resultados:

Usuario	Botometer	Predicción
@TylerLibertario	3.8	human
@cata_1933	4.1	bot
@chile_soberano	3.4	human
@Ciudadanolevi1	1.4	bot
@Rosario77488957	2	human
@1_chilemejor	1.8	human
@CNNChile	3.2	bot
@Mr_JhonnR	1.2	human
@gastonoyarzun2	1.6	human
@ChileUnidoAvanz	1.6	human

Tabla 6.1: Resultados pruebas modelo español

Como se puede observar en la Tabla 6.1, la columna de los resultados obtenidos por Botometer están coloreados de tal forma que mientras más cercano al 5 esté, más oscuro es el color; esto para indicar que es más probable que se pueda tratar de una cuenta automatizada; por ende, mientras más cercano al 0 esté, más probable que se trate de una persona genuina. Botometer utiliza una cantidad de características<sup>1</sup> para verificar la probabilidad, como el volumen de tweets políticos que retweetea, cuántas publicaciones hace en el tiempo, etc.

En la tabla mostrada, la columna Predicción corresponde al resultado obtenido por el clasificador entrenado anteriormente, el modelo Random Forest. Si bien no se puede asegurar que los valores más cercanos al 5 en la columna Botometer corresponden realmente a un bot, podemos ver si la tendencia a clasificarlo como bot es similar en ambos métodos. Aquí se puede observar que, de los 4 valores que existen en la tabla cercanos al 5, el clasificador sólo identificó a 2 de ellos como bot.

Analizando de manera más específica las cuentas, @TylerLibertario es una cuenta de Twitter que se dedica a postear y compartir tweets que tienen contenido político, mayoritariamente bromas o sátiras, indicando la alta probabilidad que la cuenta en cuestión se trate de una cuenta híbrida, es decir, que es manejada por una persona genuina y a la vez tiene

<sup>1</sup><https://botometer.osome.iu.edu/faq#which-score>

cierto grado de automatización, por ende, la clasificación como “human” realizada por el clasificador.

En el caso del usuario @chile\_soberano, revisando más de cerca el contenido de la cuenta, pareciera ser que efectivamente corresponde a un usuario genuino. Sin embargo, al tener como su principal punto de tópico la política, es posible que Botometer lo haya considerado con más probabilidad de ser un bot que comparte tweets políticos a gran volumen.

El resto de los resultados obtenidos en esta prueba de comparación de resultados parece alinearse con los resultados obtenidos con Botometer, por lo que se considera que el clasificador en español obtiene buenos resultados.

Usuario	Botometer	Predicción
@RicFlairNatrBoy	1.2	human
@tommyinnit	1.3	bot
@TheDougRush	0.1	human
@Brother_ZERO	1.4	bot
@ElectrisVibe	1.8	human
@trulykaykay	1.2	human
@NoContextBrits	3.5	bot
@SportsCenter	1	human
@ayooitsalvarez	0.4	human
@BlancoBraxx	1.4	human

Tabla 6.2: Resultados pruebas modelo inglés

Luego, en la Figura 6.2, se muestra la tabla comparativa de los resultados obtenidos en Botometer y el clasificador en inglés. Como se puede observar, la cuenta @NoContext-Brits es la que más probabilidades tiene de ser un bot en el análisis hecho por Botometer, el clasificador indicando que se trata de un bot. Haciendo un análisis al contenido obtenido de ese usuario en particular, la cuenta tiene características de ser un medio para publicar bromas o sátira respecto a un tópico en específico, sus tweets por lo general siendo texto con una imagen adjuntada, dando indicadores de una posible cuenta híbrida.

Como podemos ver, el resto de las predicciones realizadas al resto de las cuentas tiene un buen encuadre con las probabilidades calculadas por botometer, por lo que se considera que el clasificador en inglés también obtiene buenos resultados al trabajar con datos duros de Twitter.

## 6.2. Filtro de Análisis Semántico

En esta sección se presentarán las principales pruebas de las herramientas/software recolectadas para el funcionamiento descrito en el Capítulo 5. Cabe mencionar que todas las pruebas realizadas en esta sección son para comparar el funcionamiento del script realizado para este trabajo usando la librería spaCy y otra librería reconocida por sus buenos resultados en este mismo ámbito, CoreNLP.

### 6.2.1. Análisis Léxico-Sintáctico

#### 6.2.1.1. Análisis POS Tag

El primer proceso realizado para en el análisis léxico y sintáctico del texto corresponde a asegurar que el programa reconozca de la manera más precisa posible las palabras que componen al texto y etiquetar los tokens de manera coherente.

La librería *spaCy*, el componente principal del filtro desarrollado en este trabajo, posee modelos para un conjunto de lenguajes, entre ellos inglés y español. Para cerciorar que los modelos de esta librería logran predecir de manera correcta las palabras dentro de una oración en ambos idiomas, se muestra a continuación el mapa de dependencias y las etiquetas POS de dos tweets, haciendo uso de *displacy*, un visualizador de la librería spaCy, comparándolos con la salida de la librería CoreNLP de Java:

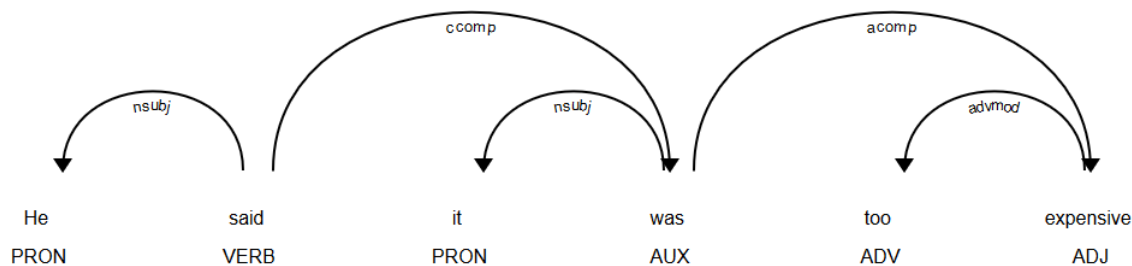


Figura 6.3: Parser tree y etiquetas POS de texto en inglés



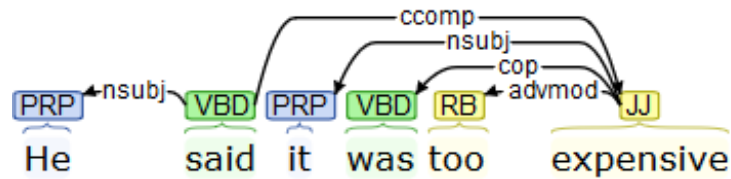


Figura 6.4: Parser tree y etiquetas POS de CoreNLP en inglés

Como se puede ver en ambas Figuras 6.3 y 6.4, las etiquetas POS y las dependencias sintácticas de ambas librerías son idénticas y se encuentran asignadas correctamente.

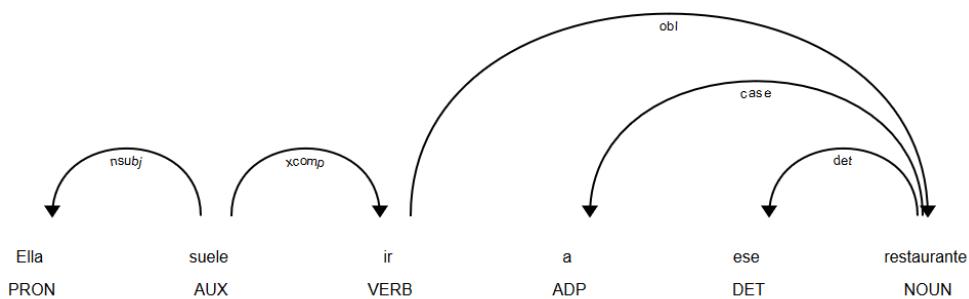


Figura 6.5: Parser tree y etiquetas POS de texto en español

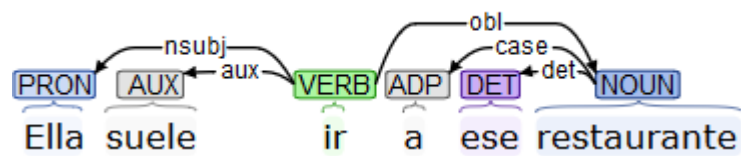


Figura 6.6: Parser tree y etiquetas POS de CoreNLP en español

Las etiquetas asignadas en las Figuras 6.5 y 6.6 son casi idénticas a excepción de la dependencia sintáctica que conecta la palabra "suele" con el verbo "ir", siendo en CoreNLP considerada como una dependencia auxiliar y en spaCy se considera un complemento clausal abierto. En este caso, el modelo en español de spaCy predijo correctamente la dependencia sintáctica.

Como se puede observar en ambas Figuras 6.3 y 6.5, spaCy es capaz no sólo de predecir correctamente pronombres, verbos, adjetivos, etc. en ambos idiomas, sino también la dependencia sintáctica entre las palabras, pudiendo reconocer el sujeto, el adverbio modificador, etc.

Hay que considerar que el modelo de inglés tiene mejor precisión que el modelo en español, pero de las pruebas que se han hecho, no se han encontrado problemas con las etiquetas asignadas por este.

### 6.2.1.2. Análisis de reconocimiento de entidades

Esta sección evalúa una fase importante antes de pasar al análisis semántico: la identificación de entidades. El reconocimiento de entidades está disponible para ambos modelos de español e inglés en spaCy, por lo que se quiere corroborar que las predicciones de estos modelos para el reconocimiento de personas, organizaciones, países, localizaciones, etc. sea el más preciso posible.

A continuación, se presenta el reconocimiento de entidades en inglés y español haciendo uso de *displacy*. Se hará una comparación de los resultados de spaCy con los obtenidos con CoreNLP.

Instead of sending 25,000 **CARDINAL** Iraqi **NORP** troops, just send the equivalent in american **NORP** troops

Watching Andrew Cuomo **PERSON** on CNN **ORG** this morning **TIME** . This man is a total fraud

Figura 6.7: Reconocimiento de entidades en tweet en inglés

Instead of sending <sup>NUMBER</sup>25000.0 <sup>NATIONALITY</sup>Iraqi troops , just send the equivalent in <sup>NATIONALITY</sup>american troops

Watching <sup>PERSON</sup>Andrew Cuomo on <sup>ORGANIZATION</sup>CNN <sup>TIME</sup>this morning .

This man is a total <sup>CRIMINAL\_CHARGE</sup>fraud

Figura 6.8: Reconocimiento de entidades en CoreNLP en inglés

Como se puede observar en las Figuras 6.7 y 6.8, ambas librerías son capaces de reconocer nacionalidad, números y frases que indican tiempo en la oración ingresada. En lo único que difieren es que CoreNLP puede reconocer entidades que tienen que ver con

cargos criminales, una categoría que spaCy no posee, por lo que no resulta un punto de comparación.

Amazon **ORG** abre hoy en EE.UU. **LOC** su primer minimercado del futuro sin colas ni cajas

Puerto Valparaíso **LOC** firma acuerdo con Senda **ORG** para fomentar prevención de drogas en su sistema portuario

Figura 6.9: Reconocimiento de entidades en tweet en español

**ORGANIZATION** Amazon abre **THIS P1D DATE** hoy en EE.UU. su primer minimercado del futuro sin colas ni cajas

**CITY** Puerto **LOCATION** Valparaíso firma acuerdo con **ORGANIZATION** Senda para fomentar prevención de **CRIMINAL\_CHARGE** drogas en su sistema portuario

Figura 6.10: Reconocimiento de entidades en CoreNLP en español

En las Figuras 6.9 y 6.10, se reconoce correctamente a Amazon y a Senda como organizaciones dentro del texto. Sin embargo, la palabra “EE.UU.” no es reconocida como un país o nación, sino como una localización en spaCy, mientras que en el caso de CoreNLP, esta no la reconoce como una entidad, lo que indica que el modelo en español de spaCy sí reconoce la abreviatura “EE.UU.” como una entidad y sólo habría que entrenarla para que la reconociera como país. Igualmente, spaCy reconoce “Puerto Valparaíso” como un lugar, mientras que CoreNLP reconoce a “Puerto” como ciudad y “Valparaíso” como una localización. En este contexto, “Puerto Valparaíso” hace referencia a una empresa, por lo que la etiqueta correcta sería organización.

Con esto, se corrobora que los modelos en español e inglés del filtro desarrollado en base a spaCy tienen una buena precisión en el etiquetado y reconocimiento de entidades.

## 6.2.2. Análisis semántico

### 6.2.2.1. Consultas a DBpedia

Para la extracción de información de DBpedia se requiere hacer uso del lenguaje “SPARQL”. Para poder facilitar la extracción dentro de las funciones, se decidió hacer uso

de la librería de Python “SparqlWrapper” en la tarea de extraer una síntesis del concepto o entidad reconocido por la fase anterior. Se hicieron pruebas tanto en el idioma inglés como español considerando solamente un enlace arbitrario para poder corroborar que se realiza la conexión de manera correcta y que es posible extraer la información solicitada.

```
extract_eng("<http://dbpedia.org/resource/Dog>")
```

Figura 6.11: Consulta de prueba a DBpedia en inglés

```
extract_esp("<http://es.dbpedia.org/resource/Francia>")
```

Figura 6.12: Consulta de prueba a DBpedia en español

A continuación, se puede observar que la información requerida (síntesis del concepto, persona, lugar, etc.) pudo ser extraída de manera satisfactoria.

```
The domestic dog (Canis lupus familiaris or Canis familiaris) is a domesticated canine which has been selectively bred over millennia for various behaviours, sensory capabilities, and physical attributes. Dogs perform many roles for people, such as hunting, herding, pulling loads, protection, assisting police and military, companionship and, more recently, aiding handicapped individuals. This influence on human society has given them the sobriquet, "man's best friend".
```

Figura 6.13: Resultado de consulta a DBpedia en idioma inglés

```
Francia (en francés, France, pronunciado /fʁɑ̃s/ ( )), oficialmente República Francesa (en francés, République française pronunciado /ʁepyblik fʁɑ̃sɛz/ ( )), es uno de los veintisiete estados soberanos que forman la Unión Europea. Su forma de gobierno es la república semipresidencialista. Territorialmente comprende la Francia metropolitana y la Francia de ultramar, siendo a su vez el país más grande de la Unión Europea. Su territorio, que incluye regiones de ultramar o Territorios dependientes, se extiende sobre una superficie total de 675 417 km². En 2017 el país contaba con 67,1 millones de habitantes (65 millones en los departamentos metropolitanos y 2,1 millones en los departamentos de ultramar).
```

Figura 6.14: Resultado de consulta a DBpedia en idioma español

#### 6.2.2.2. Consultas WordNet

Para el caso de WordNet, se pudo facilitar su uso al utilizar la interfaz contenida en la librería NLTK. Al utilizar la interfaz proporcionada por esta librería, es sencillo poder extraer los diferentes sinónimos o sentidos que puede tener una palabra (además de la definición correspondiente a cada sinónimo).

```

syms = wn.synsets("drink")
for i in range(len(syms)):
    print(syms[i].lemma_names())

```

Figura 6.15: Extracto de consulta Wordnet del verbo “drink”

En la Figura 6.2.2.2 se muestra un pequeño extracto de ejemplo para demostrar cómo se realiza una consulta a la librería de wordnet, en este caso con una palabra en inglés. Se presentan los resultados de esta en la Figura 6.16.

```

['drink']
['drink', 'drinking', 'boozing', 'drunkenness', 'crapulence']
['beverage', 'drink', 'drinkable', 'potable']
['drink']
['swallow', 'drink', 'deglutition']
['drink', 'imbibe']
['drink', 'booze', 'fuddle']
['toast', 'drink', 'pledge', 'salute', 'wassail']
['drink_in', 'drink']
['drink', 'tope']

```

Figura 6.16: Extracción de sinónimos para diferentes sentidos de la palabra “drink”

Se observa que hay varios resultados para el verbo “drink”, relacionando el verbo a algunos sustantivos o adjetivos que puedan estar relacionados con la temática de beber, confirmando el uso correcto de la librería para el idioma inglés.

A continuación, se presenta la misma consulta, esta vez en el idioma español.

```

syms = wn.synsets("tomar", lang='spa')
for i in range(len(syms)):
    print(syms[i].lemma_names('spa'))

```

Figura 6.17: Extracto de consulta Wordnet del verbo “tomar”

```
['consumir', 'ingerir', 'tomar']  
['beber', 'tomar']  
['beber', 'emborracharse', 'soplar', 'tomar', 'tomar_unas_copas']  
['consumir', 'tocar', 'tomar']  
['prender', 'tomar']  
['coger', 'tomar']  
['pillar', 'subirse', 'tomar']  
['tomar']  
['quitar', 'tomar']  
['apropiarse', 'coger', 'confiscar', 'quitar', 'tomar']  
['adquirir', 'conseguir', 'obtener', 'sacar', 'tomar']  
['tomar']  
['adoptar', 'seguir', 'tomar']  
['asumir', 'tomar']  
['asumir', 'ocupar', 'tomar']  
['tomar']
```

Figura 6.18: Extracción de sinónimos para diferentes sentidos de la palabra “tomar”

Como se puede observar en la Figura 6.18, en el idioma español se encuentra bastantes sinónimos de la misma palabra al comprarlas con su homólogo en inglés. Cabe mencionar, si bien Wordnet puede devolver los sinónimos en el idioma español como se presentan en la imagen, las definiciones de cada palabra no están disponibles en el mismo idioma, teniendo que tomar las definiciones en inglés y luego traduciéndolas al español para poder realizar el análisis semántico respecto a las entidades encontradas en el texto del tweet.

Un procedimiento parecido se realiza en el proceso de Revisión Contextual, sólo considerando la conexión al repositorio externo WordNet. Debido a que se utilizaron las mismas estructuras de consultas que las mostradas anteriormente, este proceso también mostró resultados satisfactorios al momento de prueba.

### 6.2.3. Llamada desde Node JS

Finalmente, la última prueba requerida para este filtro es corroborar que la función de Node JS que se mostró en la Figura 5.10 en el capítulo anterior. Al ser este un proceso imprescindible para poder realizar la integración final a la extensión, es necesario hacer esta verificación.

```

const path = require('path')
const scriptFilename = path.join(__dirname, 'semantic', 'analisis_seman.py')
import { spawn } from 'child_process'

async function semanticScore(tweet_text: String):Promise<number>{
  return new Promise((resolve, reject) => {
    const test = spawn('python', [scriptFilename, tweet_text])
    try {
      test.on("close", (data:any) => {
        return resolve(data)
      })
      test.on("error", (err:any) => {
        return reject(err)
      })
    } catch(error){
    }
  })
}

```

Figura 6.19: Función para invocar script de Python

```

147 semanticScore('Según Izkia Siches algunos de los "altos cargos de empresas en Chile" ganan 500 mi
148 .then(result => console.log(result))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Procesamiento_datasets> node integration_test.js
0.49779550103504966

```

Figura 6.20: Resultado obtenido de ejecución en segundo plano

Como se puede observar en la Figura 6.20, una vez ejecutada la función por consola, el filtro semántico devuelve el valor calculado dentro de los parámetros esperados (valor perteneciente a  $[0,1]$ ). Con esto se corrobora que la llamada del script a Python a través de Node JS se hace correctamente.

### 6.3. Integración

Como se planteó en el Capítulo 4, en la sección 4.3.3, las pruebas para corroborar que la integración se haya logrado correctamente son las siguientes: (1) **Captación de datos de entrada**, (2) **procesamiento** e (3) **integración a fórmula global**.

Primero se debió ejecutar el servidor local para comenzar las pruebas, como se puede ver en la Figura 6.21:

```
> www-back-end@1.0.0 start E:\Tesis\BackEnd-Integration
> nodemon --watch src --exec ts-node src/server.ts

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): src\**\*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node src/server.ts`
Server listening at port 3000
```

Figura 6.21: Servidor local desplegado

Para poder realizar de manera más sencilla la ejecución y controlar que el servidor se encargue de analizar sólo un tweet para esta prueba, se utilizó “Postman” para poder lograr enviar una única solicitud de análisis al servidor.

Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings
Query Params						
	KEY	VALUE				
<input checked="" type="checkbox"/>	weightBadWords	0.30				
<input checked="" type="checkbox"/>	weightMisspelling	0.20				
<input checked="" type="checkbox"/>	weightSemantic	0.20				
<input checked="" type="checkbox"/>	weightSpam	0.30				
<input checked="" type="checkbox"/>	weightSocial	0.33				
<input checked="" type="checkbox"/>	weightText	0.34				
<input checked="" type="checkbox"/>	weightUser	0.33				
<input checked="" type="checkbox"/>	tweetid	1568688384348155904				
<input checked="" type="checkbox"/>	maxFollowers	2000000				

Figura 6.22: Formato de solicitud en Postman

En la Figura 6.22 se puede ver la solicitud que se le enviará al servidor local con todos los parámetros, estos correspondiendo a la importancia al momento de considerar los diferentes componentes que conforman la fórmula global de credibilidad, tomando en cuenta el peso de los parámetros de la credibilidad de texto como tal, y el peso de las credibilidades calculadas de usuario, social y texto. En el parámetro “tweetid” se ingresa el id del tweet que se quiere analizar.

Una vez enviamos esta solicitud al mismo puerto que está funcionando el servidor (puerto 3000), se obtuvieron los siguientes datos que fueron puestos para ser impresos en consola:



```

> www-back-end@1.0.0 start E:\Tesis\BackEnd-Integration
> nodemon --watch src --exec ts-node src/server.ts

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): src/**\*
[nodemon] watching extensions: ts,json
[nodemon] starting 'ts-node src/server.ts'
Server listening at port 3000
Cuenta: tommyinnit
Predicción: bot
Texto entrada: { text:
  'Every American I meet says "Sorry about the Queen dying."\n\nyou're pushing me to the edge and i will fall',
  lang: 'en' }
Puntaje semántico: 13.883818001224022

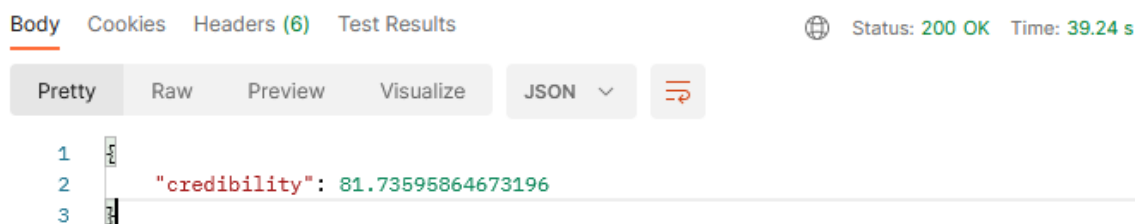
```

Figura 6.23: Resultados obtenidos tras analizar solicitud de Postman

Como se puede ver en la Figura 6.23, primero aparecen los datos del filtro de detección de bots, mostrándose que el filtro reconoció al usuario del tweet como “bot”, además de mostrar que capturó el screen name de la cuenta a la que se está analizando.

Más abajo tenemos los mismos datos impresos para el filtro semántico, que obtuvo una puntuación de 13.88 (aproximado) para el análisis realizado del texto del tweet en particular que se analizó, mostrando también la entrada que recibió la función encargada de realizar la credibilidad de texto, indicando el contenido y el idioma de éste.

Con esto podemos verificar que los datos capturados por la función de Node JS que se encarga de hacer la integración entre la extensión y los scripts de Python funciona de la manera correcta, además de ver que el procesamiento de ambos scripts obtiene resultados en el rango esperado haciendo uso de los datos de entrada correspondientes a su diseño.



```

Body Cookies Headers (6) Test Results
Pretty Raw Preview Visualize JSON
1
2  "credibility": 81.73595864673196
3

```

Figura 6.24: Puntaje obtenido por servidor de extensión

Por último, como puede observarse en la Figura 6.24, tras obtener los resultados de ambos script, se obtiene el puntaje o valor de credibilidad del tweet analizado, corroborando finalmente que, en efecto, los datos de salida de estos scripts son captados por la extensión y utilizados para el cálculo final de la credibilidad del tweet.

Adicionalmente, se hicieron dos pruebas para probar el cálculo de credibilidad haciendo uso exclusivo de la funcionalidad de bots, para hacer una comparación con la

fórmula de credibilidad original. Como se puede observar en la tabla de cuentas en inglés (Tabla 6.3), se realizó un primer cálculo de la credibilidad de las cuentas utilizando el modelo de credibilidad inicial, obteniendo un porcentaje de credibilidad para cada usuario, destacando además el valor de los dos filtros que también componen la credibilidad del usuario "Peso de Verificación" y "Peso de Creación".

usuario	etiqueta	Botometer	credibilidad (original)	credibilidad (extendida)	Detectado como	Puntaje Verificado	Puntaje Creación	Fecha creación
@BarackObama	human	1.9	91.65	86.85	human	50	46.875	2007-03-05 22:08:25+00:00
@NASA	human	2.3	92.01	92.01	human	50	46.875	2007-12-19 20:20:32+00:00
@YouTube	human	1.6	87.49	87.49	human	50	34.375	2007-11-13 21:43:46+00:00
@nytimes	human	3.4	92.00	92.00	human	50	46.875	2007-03-02 20:41:42+00:00
@elonmusk	human	1.2	89.57	89.57	human	50	40.625	2009-06-02 20:12:29+00:00
@ladygaga	human	0.6	80.2575	75.62	bot	50	43.75	2008-03-26 22:37:48+00:00
@TheEllenShow	human	1.2	90.81	86.17	bot	50	43.75	2008-08-14 03:50:42+00:00
@IGisellePizarro	human	0.6	36.19	36.15	human	0	34.375	2011-02-22 04:37:34+00:00
@INicoleromany	human	0.2	48.38	48.38	human	0	28.125	2013-09-05 20:52:02+00:00
@AlekhandraKhan	human	0.4	51.84	51.84	human	0	28.125	2013-02-13 12:53:41+00:00
@DennaMcsparrren	bot	5	27.41	19.16	bot	0	25	2014-03-04 18:11:08+00:00
@YukikoTretter	bot	5	36.87	28.62	bot	0	25	2014-03-02 10:38:13+00:00
@RochelAmaro	bot	5	38.67	30.42	bot	0	25	2014-02-20 22:28:03+00:00
@ElyseKendell	bot	4.4	25.83	17.58	bot	0	25	2014-02-26 08:57:36+00:00

Tabla 6.3: Validación del análisis de credibilidad incluyendo detección de bots (inglés)

Al calcular de nuevo la credibilidad, pero incluyendo el filtro de detección de bots, se observa que la credibilidad se mantiene igual para las cuentas que se clasifican como "auténticas", mientras que las cuentas que se clasificaban como "bot" han bajado efectivamente su credibilidad, según la nueva fórmula de credibilidad de usuario. Como se puede ver en la tabla, hay dos cuentas que fueron detectadas como "bot" a pesar de ser reconocidas como cuentas genuinas por Botometer, estas son @ladygaga y @TheEllenShow. Haciendo un análisis de estas cuentas en Twitter. En el caso de @ladygaga, sus posts tienden a promocionar eventos o productos que corresponden a su carrera, como el maquillaje o la música, además de utilizar al menos un hashtag, una mención y/o una URL en cada post, un comportamiento que podría ser tomado como un spam bot por la máquina entrenada. Para el caso de la cuenta @TheEllenShow, sus posts suelen tener siempre vídeos acompañados de un texto corto, que en su mayoría contienen al menos un hashtag y una mención. Debido

a su gran número de publicaciones en un solo día, podría tratarse de una cuenta híbrida, en la que un usuario genuino crea el contenido de las publicaciones, pero automatiza el tiempo de publicación. Además, los retweets y los likes de los tweets publicados por esta cuenta son considerablemente menores en comparación con las otras cuentas verificadas vistas, lo que también podría afectar al tiempo de clasificación

usuario	etiqueta	Botometer	credibilidad (original)	credibilidad (extendida)	Detectado como	Puntaje Verificado	Puntaje Creación	Fecha creación
@shakira	human	0.9	89.96	89.96	human	50	40.625	2009-06-03 17:38:07+00:00
@andresiniesta8	human	2.1	89.92	89.92	human	50	40.625	2009-11-18 09:33:28+00:00
@Mineduc	human	1.6	76.22	76.22	human	50	37.5	2010-05-05 22:03:42+00:00
@biobio	human	3.8	90.29	90.29	human	50	43.75	2008-05-03 15:15:06+00:00
@RAEinforma	human	3.4	87.79	87.79	human	50	34.375	2011-08-07 18:41:46+00:00
@Rubiu5	human	1.2	76.48	72.31	bot	50	34.375	2011-10-25 21:37:48+00:00
@metrodesantiago	human	3	78.36	73.88	bot	50	40.625	2009-09-11 19:15:54+00:00
@AdriUmbreon	human	0.1	44.04	44.04	human	0	25	2014-02-26 15:03:49+00:00
@AndyChatlani	human	1	38.66	38.66	human	0	34.375	2011-12-12 11:08:18+00:00
@Armunho	human	0.5	54.87	54.87	human	0	37.5	2010-12-28 01:32:48+00:00
@eugeniojaque8	bot	4.7	39.58	39.58	bot	0	0	2022-06-28 02:53:52+00:00
@RecuerdameBot	bot	2.3	48.35	46.28	bot	0	6.25	2020-04-26 16:51:44+00:00
@123trabajo	bot	4.8	53.82	42.48	bot	0	34.375	2011-03-04 15:39:40+00:00
@4geeksmxOfi	bot	3.9	28.79	28.79	bot	0	0	2022-09-02 18:12:55+00:00

Tabla 6.4: Validación del análisis de credibilidad incluyendo detección de bots (español)

La tabla 6.4 muestra un conjunto de cuentas en español. En ellas se observa el mismo patrón de funcionamiento descrito anteriormente para las cuentas en inglés en función de la clasificación realizada por el algoritmo. En este caso, hay dos cuentas "bot" que no bajan su credibilidad, ya que sus valores de Peso de Verificación y Peso de Creación eran cero respectivamente. Curiosamente, en el caso de la cuenta @RecuerdameBot, un bot autodeclarado, fue evaluado con un valor relativamente alto por el botómetro y nuestro modelo consigue clasificarlo correctamente. Al igual que en la Tabla 6.3, en esta prueba con cuentas españolas hay dos cuentas que fueron detectadas como bot a pesar de estar etiquetadas como usuario genuino (y verificadas como tal también por Botometer), y del mismo modo se realizó un análisis del comportamiento de estas cuentas. En el caso de @Rubiu5, suele promocionar eventos en los que participa utilizando hashtags y algunas URLs, además de promocionar sus propios productos, también comentando o retuiteando publicaciones de

cuentas oficiales de organizaciones que tienen que ver con los videojuegos, lo que quizás podría tomarse como un comportamiento bot. Por otro lado, aunque la cuenta @metrodesantiago corresponde a una cuenta oficial de transportes, presenta el comportamiento de una cuenta híbrida, publicando a determinadas horas para indicar el inicio y el final de los viajes en tren, o avisando con la hora exacta sobre un problema en la vía o en una estación.

De igual forma, se realizó una comparación de la credibilidad extendida exclusivamente con el filtro semántico respecto a la credibilidad original de la extensión. En la Tabla 6.5, puede observarse una selección de las cuentas utilizadas en las pruebas anteriormente mostradas para el filtro de detección de bots, teniendo exactamente la misma cantidad de cuentas del mismo idioma para analizar.

Usuario	Credibilidad original	Credibilidad con filtro	Lenguaje
@GennieRossi5062	54.29	57.86	inglés
@Malakai64	28.48	32.09	inglés
@Don_Reuters	61.8	66.38	inglés
@WhiteHouse	78.76	84.30	inglés
@ferdinandoghene	39.17	43.87	inglés
@CrazyPrize66244	37.11	41.28	inglés
@mariachua16	41.14	43.73	español
@ruthyohanna	55.28	58.56	español
@famacali	45.21	46.81	español
@SpainUN	71.49	74.36	español
@yalid12	37.15	40.25	español
@mIrDy_23	55.75	57.36	español

Tabla 6.5: Análisis de filtro semántico con cuentas en inglés y español

Debido a que al momento de la prueba el peso para el filtro semántico es de un 20 % dentro del grupo de filtros que componen la credibilidad de texto, el cambio que se aprecia en la tabla (tanto para idioma inglés como español) no es grande en comparación al original. Sin embargo, para cuentas como WhiteHouse que contienen varias entidades en sus tweets (nombres de personas importantes, lugares, etc.) por ser cuentas oficiales de alguna organización o que se dedican a la publicación de noticias, suelen tener un puntaje mayor en el filtro semántico y se ve reflejado en el resultado de credibilidad global al compararlos con el resto de los tweets en inglés.

Este mismo caso se ve reflejado con la cuenta SpainUN, siendo su aumento en credibilidad notoriamente mayor que el resto. Una observación en los tweets en cuentas de habla hispana resulta que ser que el aumento de puntaje con este filtro es considerablemente menor en comparación con cuentas de habla inglesa, lo que podría deducirse a que las herramientas ocupadas para la implementación de este filtro están más orientadas al procesamiento natural de lenguaje en inglés.

## 6.4. Tiempos de ejecución

Como prueba final, se hizo una revisión de los tiempos de ejecución de cada filtro por su cuenta, analizando posibles puntos de mejoras para su futura optimización.

### 6.4.1. Detección de bots

A continuación, se describe una tabla con los tiempos de ejecución obtenidos tras haber realizado el análisis de credibilidad en la sección anterior de integración.

Usuario	Tiempo (antiguo)	Tiempo (con filtro)
@BarackObama	1041 ms	1091 ms
@NASA	1223 ms	1261 ms
@YouTube	910 ms	950 ms
@nytimes	904 ms	945 ms
@elonmusk	930 ms	966 ms
@ladygaga	1048 ms	1085 ms
@TheEllenShow	857 ms	893 ms
@IGisellePizarro	1044 ms	1083 ms
@INicoleromany	842 ms	885 ms
@AlekhandraKhan	840 ms	885 ms
@DennaMcsparrren	826 ms	863 ms
@YukikoTretter	843 ms	880 ms
@RochelAmaro	854 ms	895 ms
@ElyseKendell	900 ms	942 ms
@shakira	1125 ms	1140 ms
@andresiniesta8	917 ms	932 ms
@Mineduc	912 ms	927 ms
@biobio	944 ms	960 ms
@RAEinforma	909 ms	924 ms
@Rubiu5	1101 ms	1116 ms
@metrodesantiago	840 ms	855 ms
@AdriUmbreon	918 ms	933 ms
@AndyChatlani	931 ms	946ms
@Armunho	982 ms	997 ms
@eugeniojaque8	946 ms	960 ms
@RecuerdameBot	1316 ms	1340 ms
@123trabajo	1017 ms	1031 ms
@4geeksmxOfi	1044 ms	1059 ms

Tabla 6.6: Tiempo de ejecución de análisis de credibilidad de las Tablas 6.3 y 6.4

A primera vista, es evidente que el tiempo de ejecución una vez es implementado el filtro de detección de bots aumenta levemente, considerando que es un nuevo proceso que agrega a toda la funcionalidad ya asentada dentro del proyecto original de T-CREO. Aproximadamente, se aumenta el tiempo de ejecución en un promedio de 40 ms, siendo considerado un tiempo bastante positivo.

Si bien su tiempo de ejecución está dentro de los tiempos estimados en “tiempo real”, hay que considerar que el script de Python que genera todo el proceso de predicción realiza también conexiones al API de Twitter, a pesar de que la aplicación principal lo hace

originalmente; esto precisamente debido a que la información que es extraída para la funcionalidad de credibilidad de usuario no requería previamente el texto del tweet a analizar, lo que podría ser un factor ralentizador para el filtro.

## 6.4.2. Análisis semántico

De igual forma, se calcularon los tiempos de ejecución para el cálculo de credibilidad mostrados anteriormente en la sección del filtro semántico.

Usuario	Tiempo original (ms)	Tiempo filtro (ms)
@GennieRossi5062	982	14158
@Malakai64	904	21510
@Don_Reuters	916	19093
@WhiteHouse	890	27033
@ferdinandoghene	980	21771
@CrazyPrize66244	1043	18180
@mariachua16	898	10403
@ruthyohanna	906	17510
@famacali	916	12835
@SpainUN	899	12245
@yalid12	921	9696
@mIrDy_23	877	12466

Tabla 6.7: Tiempo de ejecución de filtro de análisis semántico de Tabla 6.5

A primera vista, se puede entender que el tiempo de ejecución sobrepasa el límite de los 10 segundos en la mayoría de los casos. Si bien este tiempo puede aumentar considerablemente dado al largo del texto del tweet, se considera que el principal factor corresponde a la funcionalidad NLP que requiere el filtro para funcionar, lo que llevaría a la posibilidad de revisar la implementación del código para la optimización de su ejecución o, en su defecto, considerar alguna librería que tenga un balance entre buena capacidad de NLP y rendimiento. Una nota interesante es que, al comparar los tiempos de ejecución entre español e inglés, se observa que los tweets en español toman relativamente menos tiempo para procesarse. Una posible explicación para ello es el poco soporte al idioma español que se tienen de las herramientas utilizadas para la implementación del filtro, lo que podría generar que no se logran identificar entidades como tal en el proceso en español, lo que lleva a que el filtro realice el proceso de Revisión Contextual solamente, analizando los verbos y sustantivos del texto, por ende, agilizando el proceso.

Cabe mencionar que la extensión actual de T-CREo requiere una refactorización de su código para la integración final de los filtros, ya que, al inicio de su implementación, la aplicación no contaba con las futuras funcionalidades que se le han ido agregando. Esto significa asegurarse que existan funciones que extraigan la información requerida por todas las funcionalidades una sola vez, evitando más conexiones a la API de Twitter posteriores que pueden ralentizar el procesamiento de la credibilidad.

# Capítulo 7

## Implantación

### 7.1. Implantación

A continuación, se presentarán los requisitos necesarios y el ambiente en el que es utilizado el proyecto realizado.

#### 7.1.1. Requerimientos

Los requerimientos necesarios para poder instalar y ejecutar la extensión “T-CREo” requerirá tener instalado en el computador el navegador web “Google Chrome”. Las especificaciones del computador pueden ser obviadas, sólo se requiere que el dispositivo cuente con conexión a Internet.

### 7.2. Documentación

#### 7.2.1. Manual de Usuario

##### 7.2.1.1. Instalación

Para la instalación de la extensión, se requiere realizar una “build” del proyecto. Luego de obtener el directorio con la aplicación, se tiene que ingresar al navegador “Google Chrome” e ingresar en extensiones. Una vez allí, se debe activar el modo desarrollador o “Developer mode”.

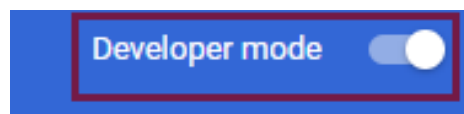


Figura 7.1: Activar modo desarrollador

Una vez realizado esto, se debe presionar el botón “Añadir paquete” o “Load unpacked”.

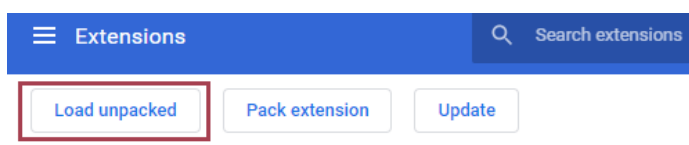


Figura 7.2: Cargar paquete de extensión

Se abrirá una ventana para seleccionar la carpeta donde se encuentra el código de la extensión. Una vez ingresado, debe aparecer la extensión agregada en la ventana de extensiones de Chrome.

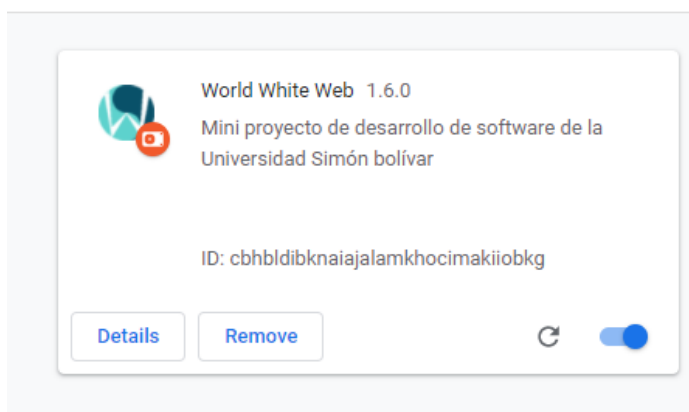


Figura 7.3: Extensión WWW agregada satisfactoriamente

#### 7.2.1.2. Cómo usar

Para utilizar la herramienta, se debe navegar a la página de Twitter e ingresar en algún tipo de hashtag o “hilos” de conversación. Se presiona el ícono de WWW en la esquina superior derecha de la pantalla, en la barra de herramientas, y aparecerá la siguiente ventana:



---

## World White Web

Text to Analyze

I'm sorry professor, but I must not tell lies

Credibility

0%

EN ▾

Verify

---

### You are currently on Twitter

Verify Page Tweets

Verify Page Tweets with Twitter Api

---

Figura 7.4: Ventana extensión T-CREo

Como se puede ver en la Figura 7.4, la parte superior permite ingresar texto para ser analizado por la herramienta, indicando el porcentaje de credibilidad a un costado del campo de texto. Así mismo, permite seleccionar el idioma en que se requiere hacer el análisis de verificación. En la parte inferior se encuentran dos botones. El primer botón corresponde a la verificación de tweets haciendo uso de web scraping, sin embargo, debido a la gran posibilidad de cambios en la interfaz de Twitter, se aconseja utilizar el segundo botón que corresponde a la verificación de tweets haciendo uso del API de Twitter.

Una vez se presione el botón de verificar los tweets de la página con el API de Twitter, este será el resultado:



Figura 7.5: Resultados de cálculo de credibilidad de tweets

Como se puede ver en la Figura 7.5, los resultados obtenidos por el modelo de credibilidad de T-CREo tienen asociado un color: mientras más confiable sea el tweet, más verde es el color del texto; en caso contrario, el color del texto tenderá a volverse cada vez más rojo.

### 7.2.1.3. Parámetros

La extensión T-CREo permite al usuario configurar el peso o importancia de algunos de los filtros para el cálculo de credibilidad, permitiéndole al usuario definir qué es más importante para él al momento de evaluar cada tweet. Para lograr configurar estos parámetros, se debe acceder al ícono dentro de la barra de tareas en la esquina superior derecha de la ventana del navegador.

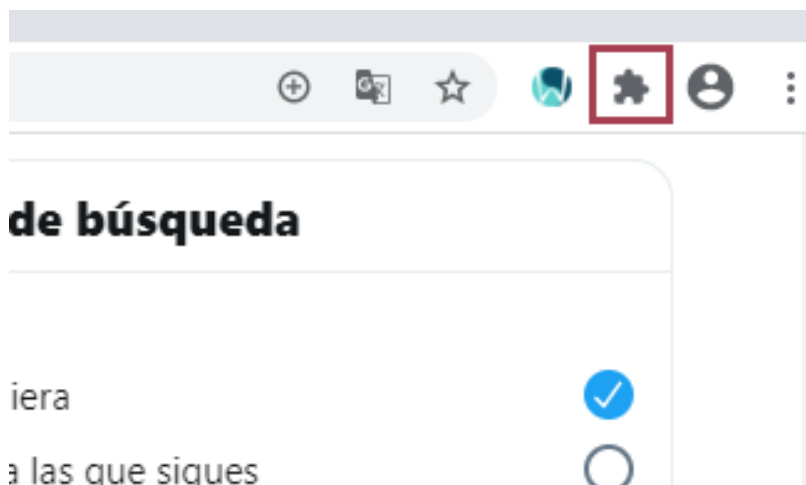


Figura 7.6: Administración de extensiones

Esto mostrará la ventana de administración de extensiones, donde se encuentran todas las extensiones instaladas en el navegador de Chrome. Luego, debe presionar el botón “Detalles” o “Details” de la extensión WWW.

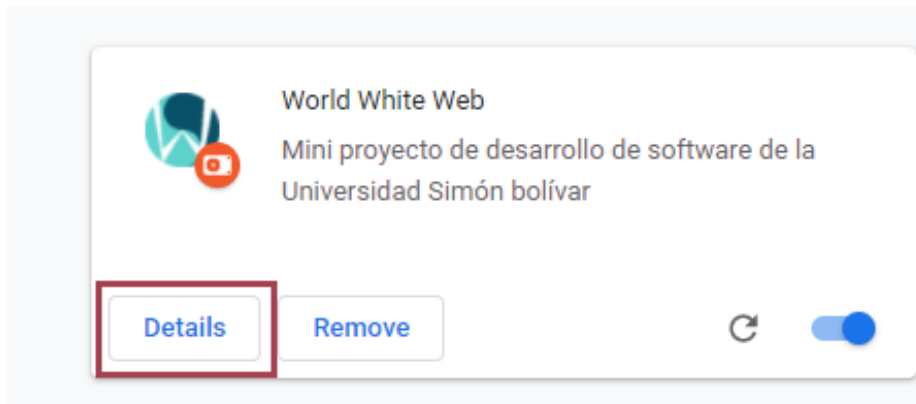


Figura 7.7: Vista de proyecto en la ventana de extensiones

La ventana que aparecerá luego mostrará una gran cantidad de información respecto a la herramienta. Se debe bajar por la ventana hasta llegar a un cuadro que dice “Opciones de extensión” o “Extension options” y hacer click al ícono al costado derecho.

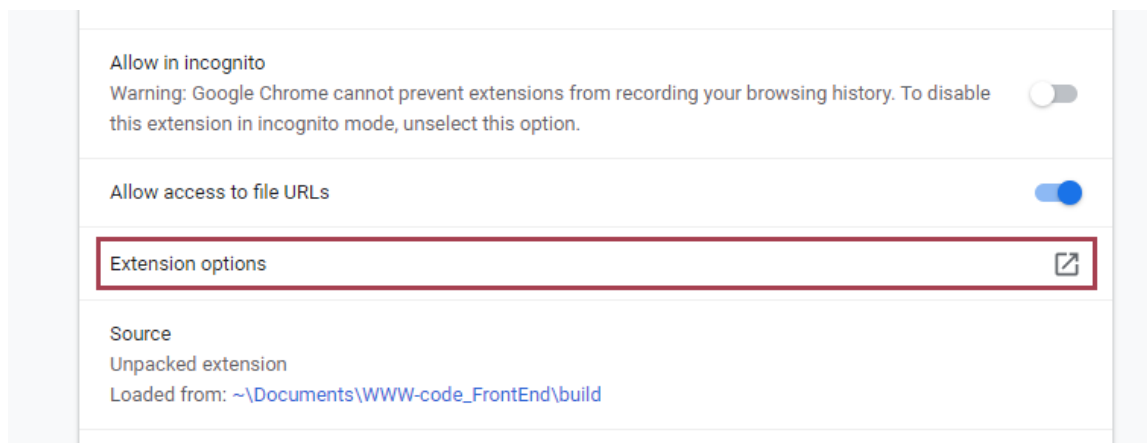


Figura 7.8: Opciones de extensión

Finalmente, la página que se mostrará a continuación es la mostrada en la Figura 7.9. Como se puede ver, aquí se permite distribuir los pesos de los filtros de credibilidad de texto dependiendo del interés o preferencia del usuario. Cabe mencionar que la suma de todos los pesos de los filtros debe ser 1, o de otra forma los parámetros estarán mal configurados.

Así mismo, se permite distribuir el peso de las tres “secciones” del modelo de credibilidad: credibilidad de texto, usuario y social. Al igual que en el caso de arriba, la suma de los tres pesos debe corresponder a 1. Finalmente, se encuentra el parámetro de máxima cantidad de seguidores que afecta directamente a la credibilidad social.

A continuación, se muestra el formulario donde se pueden personalizar los pesos de los diferentes filtros para que el usuario pueda definir qué aspecto es el que más le interesa que tenga más relevancia al momento del análisis.

## Customize text credibility parameters

Filters	Weights
Spam detection	<input type="text" value="0.30"/>
Bad words proportion to text	<input type="text" value="0.30"/>
Misspelling detection	<input type="text" value="0.20"/>
Semantic analysis	<input type="text" value="0.20"/>

## Customize tweet credibility parameters

Filters	Weights
Text credibility	<input type="text" value="0.34"/>
User credibility	<input type="text" value="0.33"/>
Social credibility	<input type="text" value="0.33"/>

## Max followers parameter

Max followers	<input type="text" value="2000000.00"/>
<input type="button" value="Save Weights"/>	

Figura 7.9: Configuración de parámetros

Como se observa en la imagen, se puede cambiar el peso que tienen todos los filtros que componen la credibilidad de texto (incluido del filtro de análisis semántico desarrollado en este trabajo) y el peso que tienen los tres factores principales que componen la fórmula de credibilidad global: credibilidad de texto, de usuario y social. Adicionalmente, se puede agregar el parámetro de máximo de seguidores.

Se destaca aquí que el filtro de detección de bots no tiene un peso como el resto de los filtros, ya que realiza una penalización de puntaje de la credibilidad de usuario si es que es detectado como bot, por lo que no se puede determinar su parámetro en estas opciones.

# Capítulo 8

## Conclusiones

Debido al avance tecnológico que hay hoy en día, se ha hecho mucho más fácil el poder compartir información y noticias con el resto del mundo, siendo el principal canal para esto las redes sociales, permitiendo a la persona individual estar informado de los acontecimientos ocurriendo a lo largo del mundo. Sin embargo, esto también facilita a diferentes personas o grupos a utilizar estos medios para poder diseminar información falsa o aprovecharse de aquellos usuarios en redes sociales que no logran discernir con facilidad los riesgos potenciales, como lo son los fraudes y phishing.

Los filtros desarrollados en este trabajo tienen como principal objetivo el realizar una evaluación del tweet que se encuentra leyendo el usuario y poder ayudar a éste mismo analizar la información que está siendo consumida, principalmente en cuanto a noticias o acontecimientos que estén circulando por Twitter. El alto porcentaje de cuentas automatizadas en Twitter es un hecho que ha sido estudiado en varios trabajos académicos y que es evidenciado por muchos usuarios de la misma red social, por lo que la inclusión de estas funcionalidades en la herramienta fue uno punto clave.

Para el desarrollo del filtro de detección de bots, se investigaron las características más importantes para la detección de bots revisando varios trabajos sobre la misma temática y se pueden observar buenos resultados al incluir características sociales más específicas, como retweets, número de amigos, seguidores, etc., obteniendo una precisión por sobre el 80 % en ambos modelos seleccionados tras el entrenamiento. Si bien existen una gran cantidad de trabajos como referencias, un gran desafío fue el encontrar un dataset apto para poder procesar y utilizar para el entrenamiento de las máquinas de aprendizaje, principalmente para el caso del idioma en español; por lo que se tuvo que solicitar acceso a un repositorio para obtener la información y seleccionar los datos necesarios para la creación de dos dataset de entrenamiento y dos más para la validación de este entrenamiento (en inglés y español).

En el caso del análisis semántico de texto, requirió bastante investigación de técnicas y de librerías que apoyaran o facilitaran los procesos que debían realizarse, al tener que

desarrollar este filtro pensando en dos idiomas (español e inglés). La principal dificultad surgió en el desarrollo de los procesos para realizar la desambiguación y contextualización, además de poder realizar un cálculo y asignar un valor respecto al análisis realizado, como también trabajar con los resultados obtenidos tras las consultas a DBpedia, que en algunos casos no tenía el mismo encoding utilizado en el resto del trabajo. Gracias a la librería spaCy, se pudo resolver varios de los desafíos encontrados en un inicio, como procesar texto en ambos idiomas y la función de distancia semántica que permite devolver un valor numérico tras el análisis realizado, como también la librería dbpedia.spotlight que permitió un fácil uso del repositorio DBpedia desde Python para las consultas necesarias.

Se generaron “diccionarios” para poder estandarizar abreviaturas de palabras que se utilizan comúnmente en redes sociales para poder “limpiar” los textos y poder mejorar su análisis, tanto en español como en inglés, como también se crearon un conjunto de reglas que el filtro debe ignorar para no penalizar de manera estricta al lenguaje más informal que es común en las redes sociales, en este caso Twitter, y así poder nivelar de mejor manera el puntaje en las primeras fases de análisis léxico y sintáctico.

Con esto dicho, hay un grupo de mejoras que pueden ser realizadas como trabajos futuros a estas dos funcionalidades desarrolladas: hacer uso de emojis o hashtags para extraer más información dentro del tweet que podría servir para contextualización de la información extraída, hacer la eliminación de hashtags y menciones más selectiva, en el caso que se utilicen ambas como un sustantivo o sujeto dentro del texto y considerar la opción de utilizar los recursos multimedia que adjuntan los usuarios en el tweet. También se podría mejorar el dataset de entrenamiento para la detección de bots, realizando un proceso más exhaustivo de etiquetamiento para poder obtener un dataset con bots más actualizados, considerando que a medida que se van mejorando los métodos de detección, éstos también van mejorando en su comportamiento para no ser detectados fácilmente, o también podrían evaluarse otros clasificadores para la predicción, como redes neuronales o algún clasificador clásico que de mejores resultados bajo otros parámetros que no se han evaluado en este trabajo.

Adicionalmente, se sigue trabajando en más funcionalidades para integrar a la extensión T-CREo, siendo el análisis de tópicos uno de ellos, por lo que se requiere hacer una refactorización de código a la aplicación principal para poder obtener la información requerida para todas las nuevas funcionalidades y poder así optimizar su funcionamiento.

# Bibliografía

- [1] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting automation of twitter accounts: Are you a human, bot, or cyborg?” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 811–824, 2012.
- [2] N. Chavoshi, H. Hamooni, and A. Mueen, “DeBot: Twitter bot detection via warped correlation,” in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Dec. 2016. [Online]. Available: <https://doi.org/10.1109/icdm.2016.0096>
- [3] A. Alarifi, M. Alsaleh, and A. Al-Salman, “Twitter turing test: Identifying social machines,” *Information Sciences*, vol. 372, pp. 332–346, Dec. 2016. [Online]. Available: <https://doi.org/10.1016/j.ins.2016.08.036>
- [4] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, “Semantic enrichment of twitter posts for user profile construction on the social web,” in *The Semantic Web: Research and Applications*, G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer, and J. Pan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 375–389.
- [5] X. Yan, J. Guo, Y. Lan, and X. Cheng, “A biterm topic model for short texts,” in *Proceedings of the 22nd international conference on World Wide Web - WWW 13*. ACM Press, 2013. [Online]. Available: <https://doi.org/10.1145/2488388.2488514>
- [6] L. M. Jose and R. K., “A semantic graph based approach on interest extraction from user generated texts in social media,” in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*. IEEE, Mar. 2016. [Online]. Available: <https://doi.org/10.1109/sapience.2016.7684118>
- [7] K. Nebhi, “Ontology-based information extraction from twitter,” in *Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data*. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 17–22. [Online]. Available: <https://www.aclweb.org/anthology/W12-5502>
- [8] “The paradigm-shift of social spambots: Evidence theories and tools for the arms race,” <https://dl.acm.org/doi/10.1145/3041021.3055135>, 2017.



- [9] M. Mohsin, “10 Social Media Statistics You Need to Know in 2021 [Infographic],” *Oberlo*, Jan 2022. [Online]. Available: <https://www.oberlo.com/blog/social-media-marketing-statistics>
- [10] “Topic: Social media,” Feb 2022, [Online; accessed 21. Feb. 2022]. [Online]. Available: [https://www.statista.com/topics/1164/social-networks/#topicHeader\\_\\_wrapper](https://www.statista.com/topics/1164/social-networks/#topicHeader__wrapper)
- [11] A. M. K. Chew and D. V. Gunasekeran, “Social Media Big Data: The Good, The Bad, and the Ugly (Un)truths,” *Front. Big Data*, vol. 4, 2021.
- [12] S. Castillo, H. Allende-Cid, W. Palma, R. Alfaro, H. S. Ramos, C. Gonzalez, C. Elortegui, and P. Santander, “Detection of bots and cyborgs in twitter: A study on the chilean presidential election in 2017,” in *Social Computing and Social Media. Design, Human Behavior and Analytics*. Springer International Publishing, 2019, pp. 311–323. [Online]. Available: [https://doi.org/10.1007/978-3-030-21902-4\\_22](https://doi.org/10.1007/978-3-030-21902-4_22)
- [13] M. Shafahi, L. Kempers, and H. Afsarmanesh, “Phishing through social bots on Twitter,” in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec 2016, pp. 3703–3712.
- [14] S. Maroofi, M. Korczyński, and A. Duda, “Are You Human? Resilience of Phishing Detection to Evasion Techniques Based on Human Verification,” in *IMC ’20: Proceedings of the ACM Internet Measurement Conference*. New York, NY, USA: Association for Computing Machinery, Oct 2020, pp. 78–86.
- [15] M. C. Benigni, K. Joseph, and K. M. Carley, “Bot-ivism: Assessing Information Manipulation in Social Media Using Network Analytics,” in *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*. Cham, Switzerland: Springer, Sep 2018, pp. 19–42.
- [16] S. Kumar and N. Shah, “False Information on Web and Social Media: A Survey,” *arXiv*, Apr 2018. [Online]. Available: <https://arxiv.org/abs/1804.08559v1>
- [17] M. Himelein-Wachowiak, S. Giorgi, A. Devoto, M. Rahman, L. Ungar, H. A. Schwartz, D. H. Epstein, L. Leggio, and B. Curtis, “Bots and Misinformation Spread on Social Media: Implications for COVID-19,” *J. Med. Internet Res.*, vol. 23, no. 5, p. e26933, May 2021.
- [18] I. Dongo, Y. Cardinale, and A. Aguilera, “Credibility analysis for available information sources on the web: A review and a contribution,” in *2019 4th International Conference on System Reliability and Safety (ICSRS)*. IEEE, Nov. 2019. [Online]. Available: <https://doi.org/10.1109/icsrs48664.2019.8987623>
- [19] F. Martinez, “Arquitecturawww,” Universidad Simón Bolívar, Rep. Tec., 2019.

- [20] —, “Worldwhitewebpresentación,” Universidad Simón Bolívar, Rep.Tec., 2019.
- [21] J. Celaya, “La empresa en la web 2.0,” *Revista Galega de Economía*, 2008. [Online]. Available: <https://www.redalyc.org/pdf/391/39118564013.pdf>
- [22] H. Hütt Herrera, “Las redes sociales: Una nueva herramienta de difusión,” *Reflexiones*, 2012. [Online]. Available: <https://www.redalyc.org/articulo.oa?id=72923962008>
- [23] A. M. d. C. Fernández-Paniagua, “Las redes sociales más utilizadas: cifras y estadísticas,” Dec 2020. [Online]. Available: <https://www.iebschool.com/blog/medios-sociales-mas-utilizadas-redes-sociales/>
- [24] M. Alrubaian, M. Al-Qurishi, A. Alamri, M. Al-Rakhami, M. M. Hassan, and G. Fortino, “Credibility in online social networks: A survey,” *IEEE Access*, vol. 7, pp. 2828–2855, 2019. [Online]. Available: <https://doi.org/10.1109/access.2018.2886314>
- [25] B. Botadra, “Web robots or most commonly known as bots.” [Online]. Available: [https://www.academia.edu/37700458/Web\\_Robots\\_or\\_Most\\_commonly\\_known\\_as\\_Bots](https://www.academia.edu/37700458/Web_Robots_or_Most_commonly_known_as_Bots)
- [26] DataDome, “Bot detection: how to identify and block bot traffic to your websites, mobile apps, and apis,” Webpage, 7 2019. [Online]. Available: <http://datadome.co/bot-management-protection/bot-detection-how-to-identify-bot-traffic-to-your-website/>
- [27] A. Karataş and S. Şahin, “A review on social bot detection techniques and research directions,” 2018.
- [28] “Biggest social media platforms 2022 | Statista,” Sep. 2022, [Online; accessed 28. Sep. 2022]. [Online]. Available: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users>
- [29] E. Alothali, N. Zaki, E. A. Mohamed, and H. Alashwal, “Detecting social bots on twitter: A literature review,” in *2018 International Conference on Innovations in Information Technology (IIT)*. IEEE, Nov. 2018. [Online]. Available: <https://doi.org/10.1109/innovations.2018.8605995>
- [30] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “A Fake Follower Story: improving fake accounts detection on Twitter,” 2014, [Online; accessed 28. Feb. 2022]. [Online]. Available: <https://www.semanticscholar.org/paper/A-Fake-Follower-Story%3A-improving-fake-accounts-on-Cresci-Pietro/b3ad4e9be2a5729462a4f6a2c24d1a5b5742b47d>

- [31] C. Yagemann, S. P. Chung, E. Uzun, S. Ragam, and W. Lee, “On the Feasibility of Automating Stock Market Manipulation,” *ResearchGate*, pp. 277–290, Dec 2020.
- [32] S. Tardelli, M. Avvenuti, M. Tesconi, and S. Cresci, “Characterizing Social Bots Spreading Financial Disinformation,” in *Social Computing and Social Media. Design, Ethics, User Behavior, and Social Network Analysis*. Cham, Switzerland: Springer, Jul 2020, pp. 376–392.
- [33] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Zhao, “Social turing tests: Crowdsourcing sybil detection,” 05 2012.
- [34] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?” *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 6, pp. 811–824, Nov 2012.
- [35] “Text analysis starter guide: What you need to know.” [Online]. Available: <https://monkeylearn.com/text-analysis/>
- [36] B. Pham, “Parts of speech tagging: Rule-based.” [Online]. Available: [https://digitalcommons.harrisburgu.edu/cisc\\_student-coursework/2/?utm\\_source=digitalcommons.harrisburgu.edu/cisc\\_student-coursework/2&utm\\_medium=PDF&utm\\_campaign=PDFCoverPages](https://digitalcommons.harrisburgu.edu/cisc_student-coursework/2/?utm_source=digitalcommons.harrisburgu.edu/cisc_student-coursework/2&utm_medium=PDF&utm_campaign=PDFCoverPages)
- [37] J. Águila, “Funciones del analizador sintáctico,” Sep 2004.
- [38] S. Bayrakdar, I. Yucedag, M. Simsek, and I. A. Dogru, “Semantic analysis on social networks: A survey,” *International Journal of Communication Systems*, p. e4424, Apr. 2020. [Online]. Available: <https://doi.org/10.1002/dac.4424>
- [39] E. C. Juarez, O. C. Villagómez, and D. V. Ayala, “Text analysis using different graph-based representations,” *Computación y Sistemas*, vol. 21, no. 4, 2018.
- [40] S. Salloum, M. Al-Emran, A. Monem, and K. Shaalan, “A survey of text mining in social media: Facebook and twitter perspectives,” *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, pp. 127–133, 01 2017.
- [41] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou, “Understand short texts by harvesting and analyzing semantic knowledge,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 3, p. 499–512, 2017.
- [42] A. Steinskog, J. Therkelsen, and B. Gambäck, “Twitter Topic Modeling by Tweet Aggregation,” 2017. [Online]. Available: <https://www.semanticscholar.org/paper/Twitter-Topic-Modeling-by-Tweet-Aggregation-Steinskog-Therkelsen/89735b06ee5d7bcb469ddc619022bbc9f2443f02>

- [43] W. X. Zhao, J. Jiang, J. He, Y. Song, and X. Li, “Topical Keyphrase Extraction from Twitter,” *ResearchGate*, pp. 379–388, Jun 2011. [Online]. Available: [https://www.researchgate.net/publication/220873589\\_Topical\\_Keyphrase\\_Extraction\\_from\\_Twitter](https://www.researchgate.net/publication/220873589_Topical_Keyphrase_Extraction_from_Twitter)
- [44] Y. Kim and K. Shim, “TWITObI: A Recommendation System for Twitter Using Probabilistic Modeling,” in *2011 IEEE 11th International Conference on Data Mining*. IEEE, Dec 2011, pp. 340–349.
- [45] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, “The State-of-the-Art in Twitter Sentiment Analysis: A Review and Benchmark Evaluation,” *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 2, pp. 1–29, 2022.
- [46] V. Ngoc Phuoc An, S. Magnolini, and O. Popescu, “Paraphrase identification and semantic similarity in twitter with simple features,” pp. 10–19, 2015, [Online; accessed 14. Mar. 2022]. [Online]. Available: <https://cris.fbk.eu/handle/11582/302001>
- [47] B. P. Sharifi, D. I. Inouye, and J. K. Kalita, “Summarization of Twitter Microblogs,” *Comput. J.*, vol. 57, no. 3, pp. 378–402, Mar 2014.
- [48] DataDome, “Organizing content – ontology 101,” Webpage, 1 2012. [Online]. Available: <https://marksprague.wordpress.com/enterprise-seo-2/what-is-an-ontology-101/>
- [49] Y. Cardinale, I. Dongo, G. Robayo, D. Cabeza, A. Aguilera, and S. Medina, “T-CREo: A Twitter Credibility Analysis Framework,” *IEEE Access*, vol. 9, pp. 32 498–32 516, Feb 2021.
- [50] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Perez-Meana, “A web scraping methodology for bypassing twitter api restrictions,” 03 2018.
- [51] K. Daouadi, R. Rebaï, and I. Amous, *Bot Detection on Online Social Networks Using Deep Forest*. Springer, 05 2019, pp. 307–315.
- [52] R. Schuchard, A. Crooks, A. Stefanidis, and A. Croitoru, “Bots fired: examining social bot evidence in online mass shooting conversations,” *Palgrave Communications*, vol. 5, no. 1, Dec. 2019. [Online]. Available: <https://doi.org/10.1057/s41599-019-0359-x>
- [53] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, “Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation,” *Math. Prob. Eng.*, vol. 2021, p. 6640499, Apr 2021.

- [54] A. Karataş and S. Şahin, “A review on social bot detection techniques and research directions,” *Bilgi Güvenliği Derneği*, Oct 2017. [Online]. Available: <https://gcris.iyte.edu.tr/handle/11147/11934>
- [55] M. Heidari, J. HJones, Jr., and O. Uzuner, “An Empirical Study of Machine learning Algorithms for Social Media Bot Detection,” in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, Apr 2021, pp. 1–5.
- [56] S. Feng, H. Wan, N. Wang, and M. Luo, “BotRGCN: Twitter bot detection with relational graph convolutional networks,” in *ASONAM '21: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 236–239.
- [57] O. Beatson, R. Gibson, M. C. Cunill, and M. Elliot, “Automation on Twitter: Measuring the Effectiveness of Approaches to Bot Detection,” *Social Science Computer Review*, p. 08944393211034991, Aug. 2021.
- [58] J. Knauth, “Language-Agnostic Twitter-Bot Detection,” *ACL Anthology*, pp. 550–558, Sep 2019.
- [59] S. Kudugunta and E. Ferrara, “Deep neural networks for bot detection,” *Inform. Sci.*, vol. 467, pp. 312–322, Oct 2018.
- [60] O. Loyola-González, R. Monroy, J. Rodríguez, A. López-Cuevas, and J. I. Mata-Sánchez, “Contrast Pattern-Based Classification for Bot Detection on Twitter,” *IEEE Access*, vol. 7, pp. 45 800–45 817, Apr 2019.
- [61] A. N.S. and S. Surendran, “Identification of malicious bots in twitter using wavelets,” *SSRN Electronic Journal*, 2019. [Online]. Available: <https://doi.org/10.2139/ssrn.3431587>
- [62] S. B. Jr, G. F. C. Campos, G. M. Tavares, R. A. Igawa, M. L. P. Jr, and R. C. Guido, “Detection of human, legitimate bot, and malicious bot in online social networks based on wavelets,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 1s, pp. 1–17, Apr. 2018. [Online]. Available: <https://doi.org/10.1145/3183506>
- [63] M. Swathi, A. Anoop, and B. Rudra, “Fake Profile Detection and Stalking Prediction on Facebook,” in *Soft Computing: Theories and Applications*. Singapore: Springer, 2022, pp. 13–21.

- [64] A. N. H. A. Nasir, S. Ramli, M. Wook, N. A. M. Razali, and N. M. Zainuddin, "CLASSIFYING FAKE PROFILE IN FACEBOOK ACCOUNT USING SUPPORT VECTOR MACHINE," *ZJDSET*, vol. 4, no. 2, 2021. [Online]. Available: <https://zulfaqarjdzet.upnm.edu.my/index.php/zjdset/article/view/53>
- [65] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on Twitter," *Computers & Security*, vol. 91, p. 101715, Apr 2020.
- [66] R. J. Oentaryo, A. Murdopo, P. K. Prasetyo, and E.-P. Lim, "On profiling bots in social media," in *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 92–109. [Online]. Available: [https://doi.org/10.1007/978-3-319-47880-7\\_6](https://doi.org/10.1007/978-3-319-47880-7_6)
- [67] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. System Sci.*, vol. 55, no. 1, pp. 119–139, aug 1997.
- [68] H. Zhu, Z. S. Rosset, and T. Hastie, "Multi-class adaboost," jan 2006, <https://hastie.su.domains/Papers/samme.pdf>.
- [69] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, aug 1996.
- [70] D. H. Moore, "Classification and regression trees, by Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Brooks/Cole Publishing, Monterey, 1984, 358 pages, \$27.95," *Cytometry*, vol. 8, no. 5, pp. 534–535, sep 1987.
- [71] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006. [Online]. Available: <https://link.springer.com/book/9780387310732>
- [72] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, oct 2001.
- [73] J. Schnebly and S. Sengupta, "Random Forest Twitter Bot Classifier," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, Jan 2019, pp. 0506–0512.
- [74] S. Raschka, Y. H. Liu, V. Mirjalili, and D. Dzhulgakov, *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing, feb 2022. [Online]. Available: <https://www.amazon.com/Machine-Learning-PyTorch-Scikit-Learn-learning-ebook/dp/B09NW48MR1>
- [75] "Usar promesas - JavaScript | MDN," Feb 2022, [Online; accessed 28. Feb. 2022]. [Online]. Available: [https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using\\_promises](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using_promises)

# Apéndice A

## Matrices de Confusión

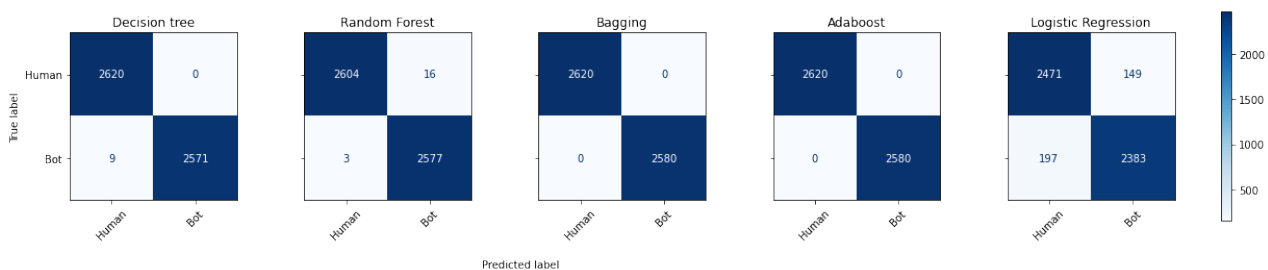


Figura A.1: Matrices de confusión de entrenamiento dataset en inglés

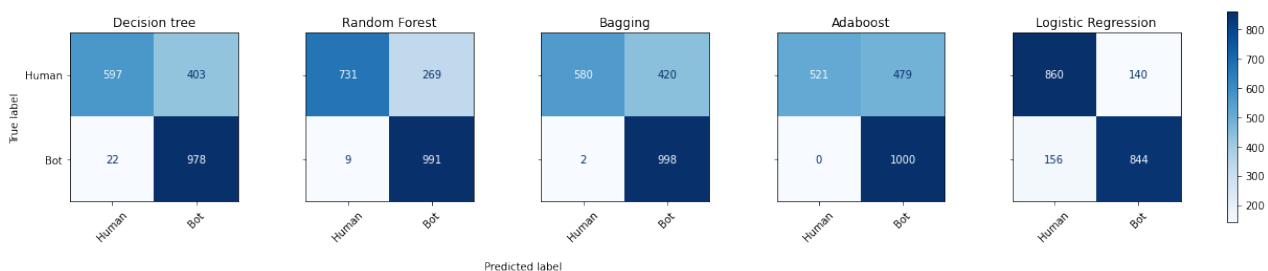


Figura A.2: Matrices de confusión de validación dataset en inglés

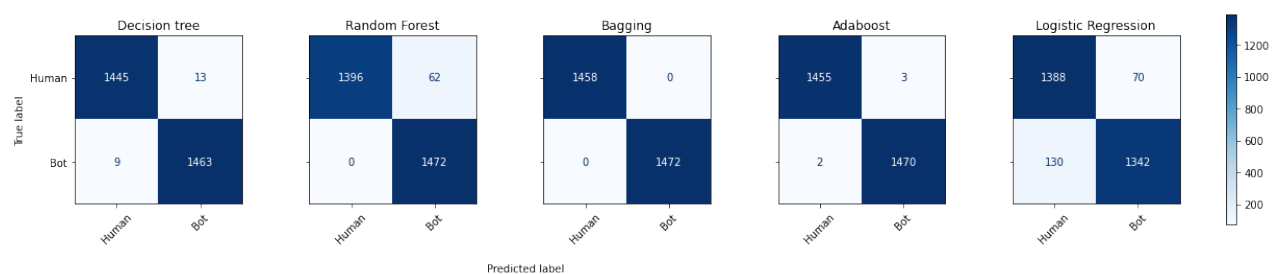


Figura A.3: Matrices de confusión de entrenamiento dataset en español

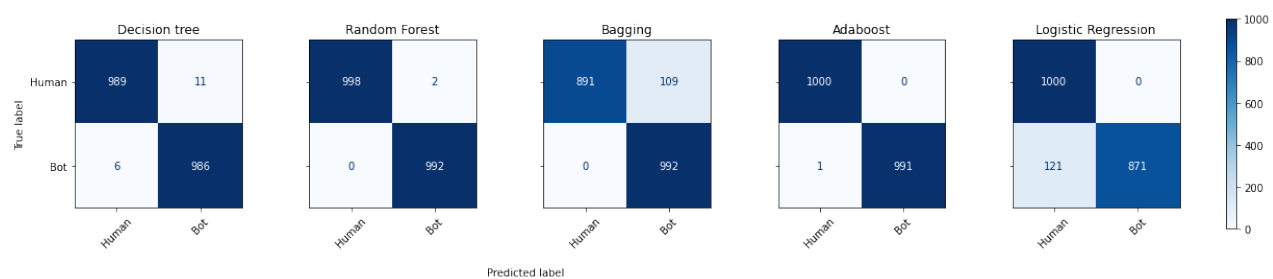


Figura A.4: Matrices de confusión de validación dataset en español