UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Influencia de las fake news en marcas y medios en la web

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Axel Javier López Esteve

Tutor: Ángeles Calduch Losa

Jorge Serrano Cobos

Curso 2017-2018

Resumen

El presente trabajo investiga el funcionamiento de las páginas web de fake news. Se centra en el estudio de las interconexiones entre estas con la temática de las marcas tecnológicas observando los enlaces que hay contenidos en sus artículos. De qué tipo suelen ser estos enlaces, si lo normal es que apunten a más dominios fake, a las redes sociales, a periódicos digitales o a otro tipo de dominios. Se utilizará web scraping para la extracción de los enlaces de estas páginas web fake y se realizará un análisis posterior con la técnica de Social Network Analysis para estudiar las relaciones. Con la muestra de páginas analizadas se observó que los dominios fake recurren de forma frecuente a las redes sociales y que estos a su vez suelen apuntar a dominios utilizados una sola vez a pesar de que el estudio sea sobre la misma temática.

Palabras clave: Noticias falsas, análisis de redes sociales, web scraping.

Abstract

This project investigates the behaviour of the fake news websites. It focuses on the study of the interconnections between these ones under the technological brands topic by looking at the links contained in their news articles. What kind of links are these usually, if it is common to point to more fake domains, social networks, digital newspapers or other kind of domains. Web scraping will be used to extract the links from these fake web pages and an analysis will be performed afterwards using the Social Network Analysis technique to study the relationships. With the set of pages analyzed, it was found that false domains frequently points to social networks, and they also aims to domains used just once despite the fact that the study is about the same topic.

Keywords : Fake news, social network analysis, web scraping.

Tabla de contenidos

Índice de figuras

1.Introducción

1.1 Motivación

Las fake news (“historias falsas que aparentan ser verdaderas, difundidas por internet o utilizando otros medios, normalmente para influenciar la tendencia de voto o como una broma” según el diccionario de Cambridge) están a la orden del día, a pesar de que existen desde el nacimiento del periodismo. Se ha cambiado la manera de difundir las noticias, el medio, ahora de una manera mucho más rápida gracias a las nuevas tecnologías que provocan que estén cada vez más presentes en nuestro día a día aunque muchas veces no nos demos cuenta o no se les dé suficiente importancia a priori. Donde más impacto causan suelen ser en las redes sociales debido a su rápida difusión cuando en Facebook o Twitter se enlaza un link con un titular llamativo y que incluso sin llegar a leerlo o clickar se comparte entre los allegados y que poco a poco se va difundiendo.

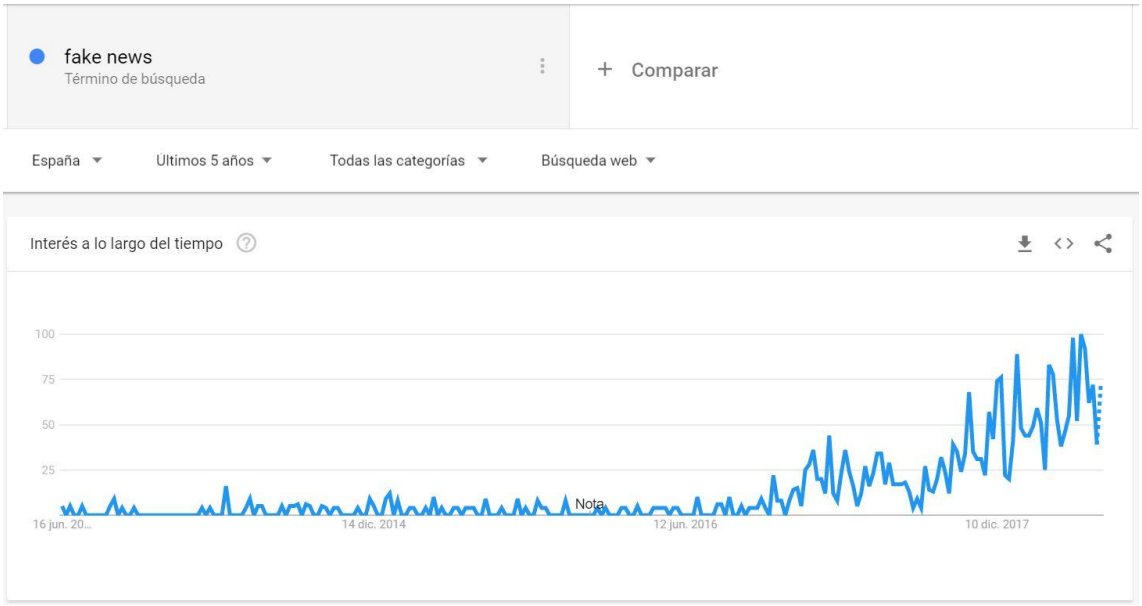

Figura 1. Interés de los usuarios en las fake news. Fuente: Google Trends.

La anterior captura muestra el Interés de los usuarios de internet en la búsqueda del término ‘fake news’ desde junio de 2013 hasta junio de 2018 en la herramienta online Google Trends según una puntuación relativa.

Si esto sucede a gran escala pueden ser importantes las consecuencias de las fake news no solo en la actualidad, como por ejemplo la falsa muerte de Phil Collins, un famoso músico británico que en los años 90 (no solo en el presente con el poder de internet) una famosa cadena de radio difundió una noticia acerca de su muerte, sin contrastar las fuentes formándose un gran revuelo por aquel entonces. El mismo caso es el de César Millán (encantador canino) aunque mucho más reciente, que también se difundió sobre todo a través de redes sociales su falsa muerte en 2014.

Más nuevo es el caso de las últimas elecciones a la presidencia de los Estados Unidos, donde explotó el tema de las Fake news por su más que posible influencia en la de tendencia de voto, por ejemplo noticias en las que se difundió información falsa para apoyar a Trump a conseguir la presidencia, dos de las más importantes fueron: “El papa Francisco aprueba a Trump” y “Clinton aprueba la venta de armas a islamistas radicales”, a pesar de que Trump mencionara varias veces a las Fake news como algo negativo, algunos estudios muestran que las fake news pudieron ser el causante de la victoria del empresario americano.

No siempre son relacionadas con el tema político o con falsas muertes, ya que pueden ser utilizadas de diferentes maneras y con diferentes finalidades, como por ejemplo en noticias relacionadas con marcas al difundirse falsos bulos que pueden perjudicar su imagen y posición económica, movidas por servicios de marketing digital que se encargan de difundir información falsa o distorsionada.

En los artículos de noticias falsas que se encuentran dentro de estas páginas de fake news suelen haber numerosos enlaces que dirigen a otras páginas fake, redes sociales, incluso a periódicos con información que pueda ser más verídica para manipular la información o hacer creer al lector que la información puede ser verdadera, distorsionando la realidad en beneficio de sus intereses. El estudio de estos enlaces puede mostrar de qué forma las páginas de fake news funcionan para que sus objetivos se cumplan y beneficien a sus intereses.

1.2 Objetivos

El objetivo del presente trabajo es mostrar un análisis de cómo funcionan las páginas web de fake news enfocándose en el estudio de los enlaces que dichas páginas utilizan en sus artículos como fuente o para apoyar a estos mismos artículos; es decir, enlaces que se encuentran dentro del contenido de los artículos, y los cuales pueden dirigirse a otras páginas.

El objetivo principal es obtener un conjunto de datos lo suficientemente grande para realizar un posterior análisis de las diferentes interconexiones de estas páginas web fake y así saber un poco mejor cómo funcionan, basándose en este aspecto de las relaciones. Si suele haber dominios “comunes” que usen la mayoría de estas páginas web fake, qué tipos de dominios son, si son también dominios fake o si por el contrario son otro tipo de dominios de mayor renombre para dar más credibilidad. También si las redes sociales pueden tener incidencia en las noticias de fake news, si estos dominios fake también se suelen apoyar en estas o no a la hora de redactar artículos.

Todo este análisis posterior será realizado en la herramienta open source Gephi que se detallará más adelante y que se encargará de dibujar un grafo con todas las interconexiones entre estas páginas y sobre el que se realizará el estudio, el social network analysis. Cada dominio será un nodo en el grafo y las relaciones entre estos dominios, aristas en el grafo.

2. Estado de la cuestión.

2.1 Introducción al Social Network Analysis.

Actualmente hay diferentes tipos de investigaciones relacionadas con las fake news ya sea algoritmos de detección de fake news, algoritmos de clasificación de fake news, análisis de redes sociales (Social network analysis), entre otros.

Entrando en detalle según el sociólogo John Scott, “el análisis de redes sociales no es solo una guía práctica para hacer amigos o socios de negocio, aunque sea útil en estas situaciones. Más bien abarca a gran escala el análisis sociológico y a un conjunto de técnicas metodológicas cuyo propósito es describir y explorar los patrones que aparecen en las relaciones sociales que los individuos y grupos forman entre ellos”.

Los fundadores de la sociología vieron las sociedades como organismos sociales o sistemas con ‘estructuras’ de instituciones y relaciones, ya sean entre familias o amigos cercanos. Pero solo unos pocos exploraron las estructuras sociales con detalle. Principalmente fueron teoristas alemanes como Simmel, Vierkandt y Von Wiese, los que empezaron a conceptualizar sobre las estructuras sociales una metáfora sobre lo que ahora serían la web y las redes sociales.

Y ahí, en las redes sociales, donde podemos observar algunos ejemplos de la aplicación del social network analysis, cuando Facebook sugiere algunos amigos cercanos a nuestro círculo para añadirlos a nuestra lista de contactos o cuando nos intenta recomendar productos según las páginas que hemos visitado.

Estas conexiones que las redes sociales reconocen entre las personas de nuestro alrededor y entre otros factores como puedan ser nuestros gustos, intereses, dinero o poder, entre otros, son un gran factor a la hora de conocer cómo pensamos y qué hacemos, pero usando métodos estadísticos tradicionales no nos ofrecen una forma óptima de explicar estas conexiones sin el análisis de redes sociales (Social network analysis), como la búsqueda de patrones en estas conexiones, la formación de comunidades, etc.

Este tipo de análisis proporciona mayor ventaja al investigar cómo funciona una sociedad por poder estudiar al mismo tiempo a los individuos de esta y cómo se conectan o desconectan entre ellos, grupos sociales, gustos, y sobretodo a cuantificar las tendencias de los individuos dentro de estas estructuras sociales.

La mayoría de los trabajos relacionados con el social network análisis están enfocados a temáticas muy diferentes como ciencias políticas, sociolingüística, economía, historia, geografía, etc, ya que es una técnica aplicable a una gran variedad de campos diferentes.

A continuación se explicarán algunos trabajos más relacionados de cara a las redes sociales y a las fake news que se han basado o han utilizado la técnica del social network analysis, aunque en la actualidad hay muy pocos proyectos que relacionen estos dos términos y es lo que se pretende hacer en este proyecto. Sobre todo no existen investigaciones en las que las redes sociales no sean el foco principal.

2.2 El auge de los robots sociales.

Algunos trabajos como “The rise of social bots” tratan el impacto que tienen los bots (cuentas de usuario creadas por programas informáticos, también llamadas cuentas fantasma) a la hora de difundir noticias en la red social Twitter. A pesar de que pueden estar diseñados para proveer servicios útiles no quita que en otras ocasiones pueden ser utilizados para ser dañinos a la hora de difundir información no verificadas o rumores por redes sociales. Todo ello observable desde un grafo donde se pueden ver y analizar todas las conexiones que se realizan entre las cuentas de los usuarios, ya sean bots o no.

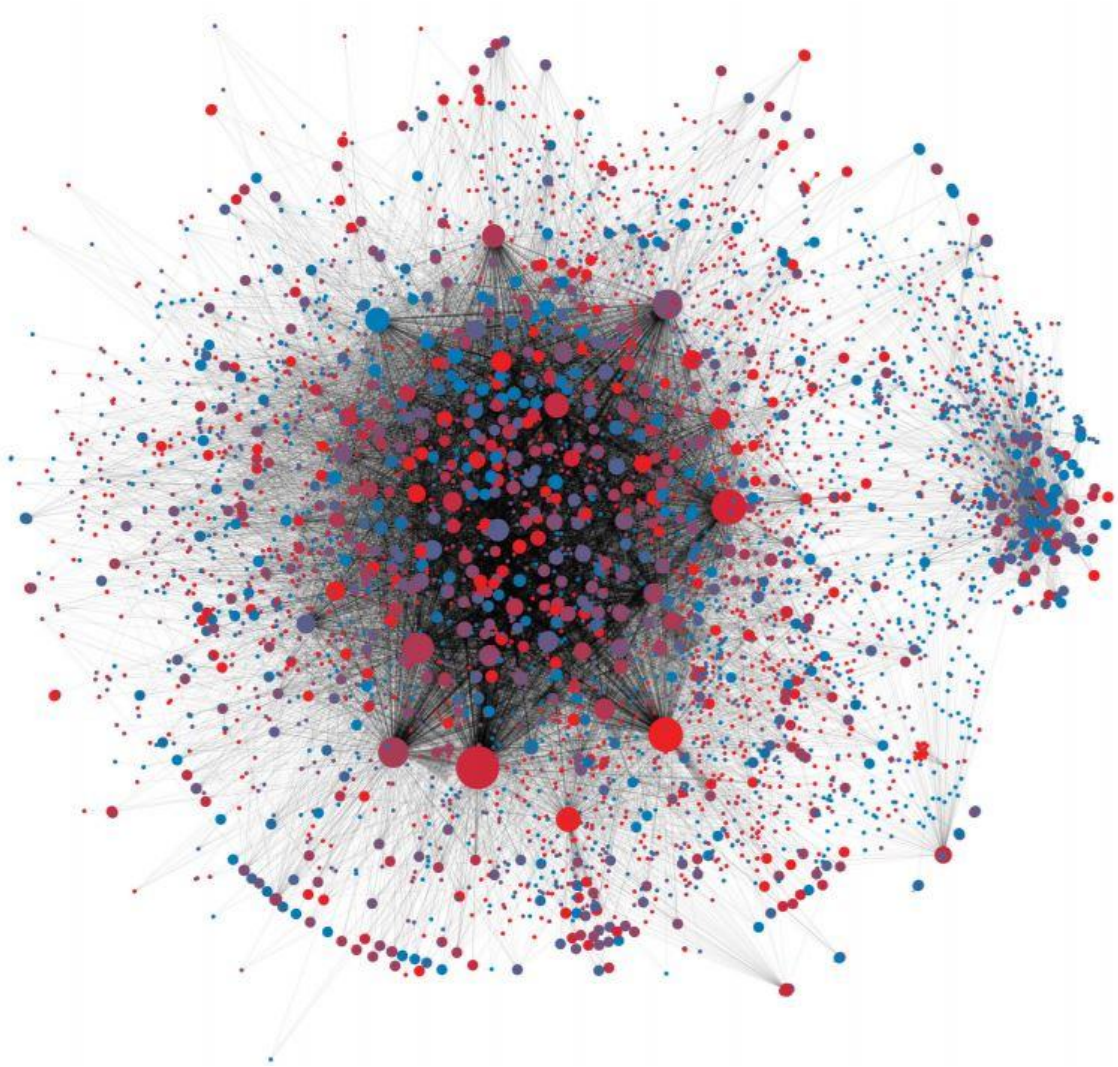

Figura 2. Difusión de tuits en Twitter sobre la ley de vacunación en California . Fuente: The rise of social bots.

Este grafo ilustra cómo los bots son usados para afectar y posiblemente manipular el debate acerca de la política de vacunación sobre una reciente ley en California. Los nodos representan usuarios de Twitter, su tamaño en función de su influencia (número de seguidores), y el color de si son bots o no, los nodos rojos sean posiblemente bots y los azules humanos.

2.3 Difusión de bulos en las redes sociales.

Otro artículo científico llamado “Spread of hoax in Social media” realiza un análisis de las comunicaciones que muestran la propagación del bulo. Cada nodo representa una cuenta de Twitter con el número de seguidores, siendo más grande o menos el nodo dependiendo de este número. El hecho de que cada tuit pueda ser leído por los seguidores es la razón de que una farsa pueda ser propagada tan rápidamente entre personas.

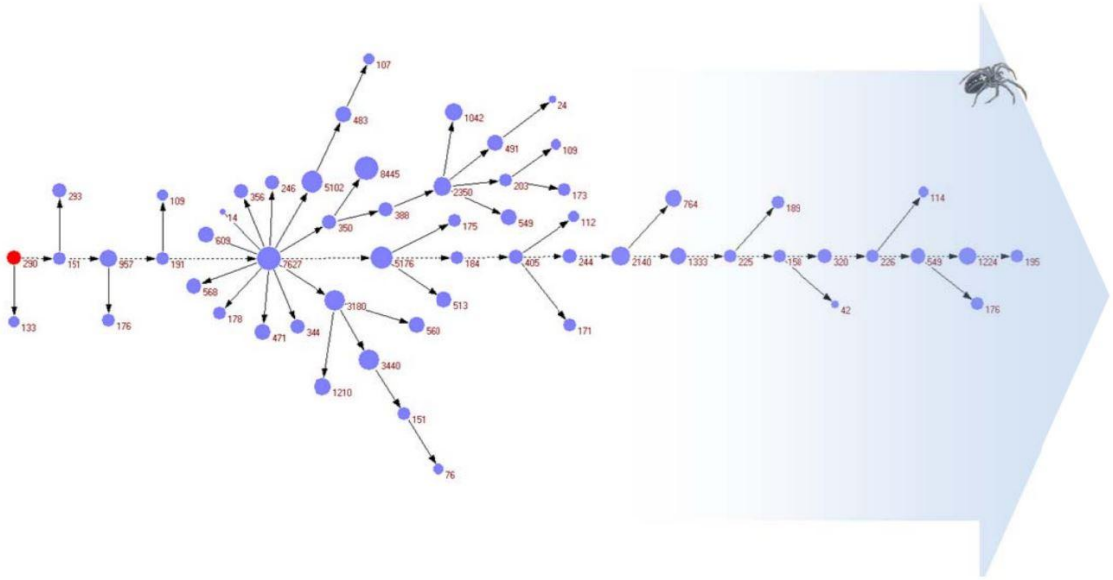

Figura 3. Evolución de un rumor en Twitter. Fuente: Spread of hoax in social media.

Esta imagen muestra una difusión empírica de un rumor por Twitter acerca de la muerte de un personaje famoso. Los nodos tienen etiquetas con el número de seguidores de cada usuario de lo que depende su tamaño también. La flecha azul situada en el fondo a la derecha indica que el rumor ha sido reconocido como una farsa después de que un medio nacional cubriera el tema al mostrar que era definitivamente una mentira.

2.4 Sentido de las redes sociales durante eventos catastróficos.

La siguiente investigación llamada “Sense-making in social media during extreme events” analiza cómo los usuarios usan la red social Twitter durante los eventos catastróficos de: Secuestro en una cafetería en Sidney (2014), el accidente de avión Germanwings (2015) y los ataques terroristas de Bruselas (2016). Una de las partes en la que se centra el trabajo es en la influencia de los usuarios de Twitter mediante un análisis de redes sociales, para así visualizar los patrones que existen en las comunicaciones entre los grupos de usuarios, es decir, organizaciones o individuos; tweets y retweets.

Los resultados del social network analysis revelaron que los medios de comunicación tienen una mayor importancia durante estos eventos, especialmente en el de Sydney y Germanwings, a pesar de que también habían usuarios privados entre los usuarios de Twitter más retuiteados pertenecientes a medios de noticias.

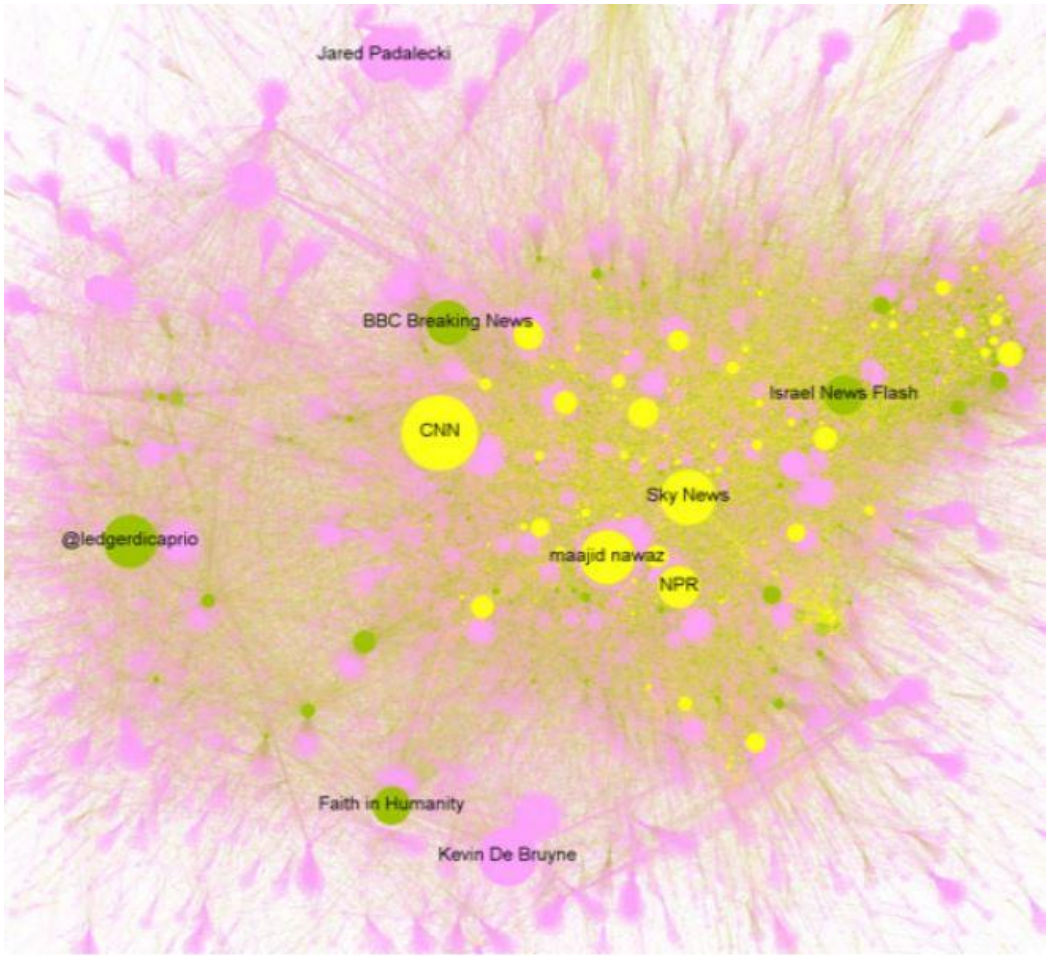

Figura 4. Grafo difusión atentado Bruselas en tuits. Fuente: Sense-making in social media during extreme events.

Esta última imagen muestra la estructura de la red de comunicaciones de tuits sobre el

atentado de Bruselas. Los nodos representan a los usuarios más activo (amarillo), usuarios activos (verde) y poco activos (rosa); el tamaño del nodo varía según el número de retweets que ha recibido ese usuario en concreto. Cuantos más retuits más grande la burbuja.

2.5 Minería de opinión sobre tuits en Twitter.

El último trabajo también estaría relacionado con las redes sociales, que son las fuentes principales para obtener información acerca de la opinión de la gente y sus intereses sobre diferentes temas, en este trabajo concreto sería sobre Twitter.

Este estudio muestra la aplicación del análisis de sentimiento (minería de opiniones) sobre diferentes temáticas realizando consultas a Twitter en busca de tweets con dichas temáticas y realizando un análisis de los tuits encontrados en función de si son positivos, negativos o neutros. El resultado posiblemente cambie si las peticiones se realizan en períodos de tiempo diferentes.

TABLE I. SENTIMENT ANALYSIS RESULTS

Query	Positive	negative	Neutral
Movie	53	11.1	35.8
politics	26.6	12.2	61.1
fashion	38.8	13.3	47.7
fake news	16.3	72.1	11.4
Justice	35.2	15.9	48.8
Humanity	36.9	33.3	29.7

Figura 5. Tabla de análisis de sentimiento. Fuente: Sentiment analysis of Twitter data.

Esta tabla muestra los resultados de la minería de opinión realizando diferentes consultas sobre los términos: “movies”, “politics”, “fashion”, “fake news”, “Justice” y “Humanity”; cada columna muestra el porcentaje de tuits con contenido positivo, negativo o neutro. Los resultados son la media de tres peticiones de búsqueda con los últimos 300 tweets.

Para este caso vamos a centrarnos en utilizar esta última técnica, el social network analysis, pero aplicada a otro caso concreto sin realizar este enfoque a las redes sociales basado en el compartir enlaces o retuits y analizando sus interconexiones en un grafo y viendo los nodos más influyentes (usuarios) dependiendo del tamaño, cuáles de estos son cuentas bots, qué nodo puede ser el origen de la fake news. O analizando las conexiones entre nodos, que serían como los tweets pasan entre usuarios por los retweets.

Está técnica estará aplicada pero a las interconexiones entre páginas web fake y analizando el grafo final como los ejemplos ilustrados en las dos figuras anteriores (2, 3 y 4). Haciendo una analogía al anterior ejemplo, los nodos serían los dominios de estas páginas web fake, las aristas dominios que direccionan a otros, el tamaño del nodo sería según el número de enlaces saliente (aunque se podrá variar el tamaño según diferentes métricas) que posee este nodo, es decir, sería como comparar el

número de retweets con los enlaces salientes de un dominio.

3. Diseño de la investigación

3.1 Metodología

La metodología de esta investigación se podría aplicar a situaciones en las que se quiera realizar un estudio sobre otro tipo de temática diferente que no sean marcas de móviles/tecnológicas que es la temática que se va a utilizar, siguiendo el mismo esquema de pasos, como puede ser por ejemplo el mundo de las marcas de automóviles, de moda, o de otro tipo sin ser necesariamente marcas como con nombre de políticos, deportistas, clubs de fútbol, etc. Ya que los cambios que habría que realizar para hacer la investigación sobre otra temática serían muy pequeños, la metodología sería la misma.

El estudio ha sido dividido en diferentes partes (que en conjunto forman la metodología) las cuáles se abordarán paso por paso de una manera cronológica al trabajo realizado y desarrollando a priori, sin entrar todavía en el trabajo, una pequeña explicación sobre las tecnologías utilizadas en las que se ha apoyado el trabajo.

Las partes globales en las que se ha dividido el trabajo son:

Tecnologías utilizadas: Previo al abordaje del proyecto realizado se realizará una enumeración y breve explicación con relación al proyecto, de las tecnologías que se han utilizado.

Obtención de enlaces: Ya entrado en la investigación el primer paso será realizar peticiones de búsqueda personalizadas a Google para buscar artículos en un conjunto de dominios fake relacionados con una palabra clave determinada como lo serán las marcas móviles/tecnológicas, para así obtener una base de enlaces que sean de una página web fake y que contengan relación con una marca tecnológica.

Tratamiento de enlaces: A continuación, se realizará una búsqueda de enlaces salientes de toda esta base de direcciones que se han encontrado en el paso anterior, para obtener todas las relaciones dominio entrante - dominio/s saliente/s.

Extracción de datos: Una vez se hayan obtenido todos los enlaces salientes de todos los artículos, se extraerán todas las relaciones entre los dominios estructurada en una hoja de cálculo de dos columnas: Source (dominios entrantes) y Target (dominios salientes).

Importación de datos a Gephi: se explicará de una forma sencilla y superficial como se ha realizado la importación de datos a Gephi.

Análisis: Por último, se analizarán todos los datos contenidos en esta hoja de cálculo con un programa de visualización de grafos (Gephi).

3.2 Tecnologías utilizadas

Para desarrollar el trabajo al completo ha sido necesario el uso diferentes tecnologías para diferentes partes del proyecto, ya sean entornos de desarrollo software, librerías u otro tipo de programas. La explicación de estas tecnologías se realizará siguiendo el orden cronológico en el que se hayan ido utilizando para realizar el proyecto.

Figura 6. Logo de Pydev y Eclipse. Fuente: José Manuel Gil Pérez.

Pydev: es un IDE (entorno de programación software) open-source creado en 2003. Está integrado en la plataforma software de código abierto Eclipse, la cual posee un junto de herramientas para la programación open-source multiplataforma para desarrollar aplicaciones. Este IDE es usado para la programación con el lenguaje Python, que será el lenguaje con el que realizaremos una parte de la investigación. Además ofrece funcionalidades como el refactoring, debugging y análisis estático entre otras.

Para este proyecto, Pydev será utilizado para la realización de scripts durante las tres primeras fases de la investigación: obtención de enlaces base, tratamientos de los enlaces y extracción de datos.

Figura 7. Logo Selenium. Fuente: seleniumhq.org.

Selenium: es un framework portable de software testing para aplicaciones web creado en 2003. Es un software multiplataforma que puede ser ejecutado para realizar pruebas en navegadores actuales y que proporciona un lenguaje de dominio específico (DSL) para realizar pruebas utilizando diferentes lenguajes de programación como Java, C#, Python, Ruby,etc.

Selenium está formado por diversos componentes, cada cual teniendo un objetivo específico en el desarrollo de pruebas software para aplicaciones web. En este proyecto se utilizará el componente llamado Selenium WebDriver. De una forma sencilla lo que realiza Webdriver es aceptar comandos que el usuario le solicita, los envía a un navegador (como Chrome, Safari o Firefox) y recupera los resultados.

Al ejecutar Selenium Webdriver se abre el navegador y se empiezan a realizar los comandos como si fuera un usuario normal, solo que está automatizado. Al tener la ejecución de Selenium una retroalimentación visual ha servido para destacar que páginas web están más relacionadas con política y cuáles no tanto, ya que a la hora de realizar el primer paso del trabajo de investigación, la obtención de enlaces base, es importante saber qué dominios escoger para determinar qué páginas fake nos pueden servir, porque hay algunas páginas que solo realizan artículos de noticias relacionados con la política, a pesar de que en algunas situaciones las marcas móviles puedan estar relacionadas con esta.

Con este paso previo, se descarta un número determinado de dominios que más adelante no se tendrán en cuenta a lo largo del proyecto, por ser páginas web fake dedicadas a un tipo de temática o que entre las temáticas que tratan sus artículos no se encuentra aquella que interesa.

Figura 8. Logo Requests. Fuente: docs.python-requests.org.

Requests es una librería para Python con la que se pueden realizar todo tipo de peticiones HTTP a páginas web de una forma muy sencilla.

Además de poseer diferentes características como enviar encabezados personalizados para evitar que se detecte el programa como un bot y que algunas páginas denegarán el acceso devolviendo el código de error 403, también admite parámetros a la hora de realizar una petición get como pueden ser la verificación SSL, el tipo de codificación, etc.

En este proyecto se utilizará esta librería para poder realizar peticiones a todos los artículos que utilizaremos como base para el estudio de interconexiones entre las

diferentes páginas web fake, concretamente en la segunda parte de la investigación: el tratamiento de los enlaces.

Figura 9. Logo Beautiful Soup. Fuente: sixfeetup.com.

BeautifulSoup es otra librería para Python cuya finalidad es la de analizar documentos HTML y XML.Construye un árbol con el contenido de todo el documentos para así poder navegar por los nodos contenidos en el fichero, buscar elementos que sean de interés, recuperarlos según las necesidades que tenga el usuario.

Además convierte automáticamente los documentos a UTF-8, por lo que no hay que preocuparse acerca de la codificación, siempre y cuando el contenido del documento que se requiera recuperar tenga una codificación especificada, ya que si nos Beautiful Soup no podrá detectarla y habrá que especificarla manualmente.

Su gran utilidad reside principalmente en todas las posibilidades que nos ofrece para buscar elementos sobre un fichero html que se extraerá de una página web, y es esta la principal razón por la que ha sido elegido para este trabajo.

A la hora de buscar permite multitud de variedades. Búsqueda por nodos en cualquier dirección, búsqueda por tipo de atributos (para por ejemplo buscar los enlaces salientes, buscaremos todos los nodos etiquetados con el atributo a, usado para los hiperenlaces), búsqueda de nodos que contengan una cadena de texto determinada, etc.

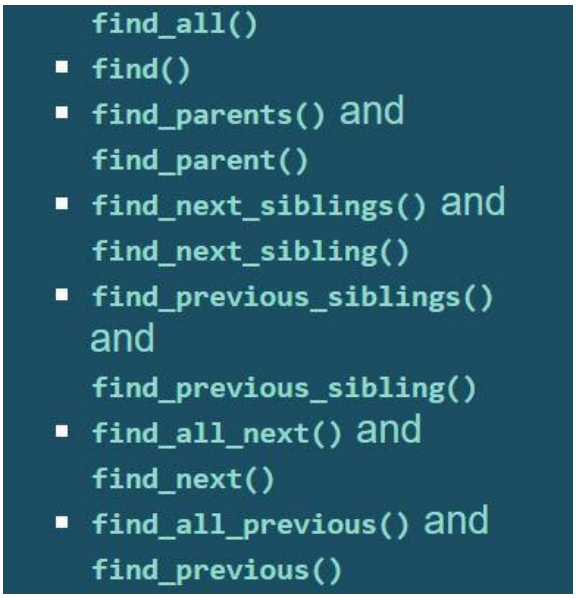

Figura 10. Algunas de las funciones "find" que ofrece la librería BeautifulSoup. Fuente: crummy.com.

Posiblemente sea la tecnología que más influencia tendrá en la parte principal del proyecto, la de tratamiento de enlaces. Con esta librería se buscarán enlaces salientes por todos los artículos que le pasemos como entrada al programa y se irán recuperando y almacenando en una lista que más tarde se extraerá en formato csv.

Había una alternativa conjunta a la utilización de BeautifulSoup y requests que era Scrapy, un framework open-source para realizar web scraping (extracción de datos de páginas web).

A pesar de que es bastante más nueva y más eficiente en términos de rendimiento se optó por las otras herramientas por tener una curva de aprendizaje menos elevada, y al ser más antiguas la comunidad que hay formada a su alrededor es mucho mayor a la hora de buscar dudas, información, bugs, etc. Por otra parte el rendimiento no es un factor tan importante por la limitación de búsquedas que Google permite en un periodo de tiempo. Y porque no se trabajará con un volumen de datos relativamente grande.

TIdextract: es un módulo Python que sirve para separar el dominio superior genérico (.com, .org, .edu) o el dominio superior geográfico (.uk, .es, .jp) del dominio o subdominio de una URL, mediante la utilización de la lista pública de sufijos "*https://www.publicsuffix.org/*".

Puede ser fácil obtener el dominio sin la utilización de herramientas externas de páginas como youtube.com, twitter.com o facebook.com, en las que obteniendo la penúltima palabra de la cadena se obtendría el dominio , pero hay otras en las que se acumulan subdominios y dominios superiores como amazon.co.uk o forums.bbc.co.uk en las que este método no funcionaría, y por esto ha sido necesaria la utilización de esta herramienta.

En esta investigación se utilizará este módulo para la obtención de los dominios de los

enlaces y para comparar dos direcciones diferentes para saber si comparten el mismo dominio, a la hora de considerar enlaces salientes o no de un artículo, ya que si por ejemplo se está analizando una noticia de una página: example.com y dentro de la noticia hay un enlace que redirige a blogs.example.co.uk se necesitará saber el dominio para tener en cuenta dicho enlace o no.

Figura 11. Imagen CSV. Fuente: zamzar.com.

CSV (comma separated values): es un formato de tipo open source destinado para leer y escribir datos en una tabla. Tal como su nombre indica los valores en las columnas se separan mediante comas.

Es uno de los formatos más comunes a la hora de importar y exportar datos de hojas de cálculo. Los datos se almacenan por columnas separadas por comas y filas como una hoja de Excel de forma que en la primera fila se establecen los nombres del tipo de variable que se va a guardar en dicha columna. Suelen estar formados para almacenar un gran número de datos

Así que la librería csv para Python implementa diferentes módulos que ofrecen la posibilidad de leer y escribir datos en el formato csv, por lo que si exportamos todos los enlaces en formato csv utilizando esta herramienta, luego desde la herramienta de visualización de grafos Gephi se podrán importar de una forma sencilla.

En la última parte del proyecto previa al análisis se creará y rellenará un archivo .csv con toda la información extraída, dos columnas en una tabla de datos, la primera con los dominios entrantes y la segunda con los salientes.

Figura 12. Gephi logo. Fuente: snola.es

Gephi es un programa open source destinado a la visualización y exploración de grafos. Permite personalizar los nodos y aristas de muchas maneras, como aumentando el tamaño del nodo según el número de ocurrencias de un mismo dominio o establecer un rango de colores entre los nodos según el nodo sea más grande o más pequeño. Por ejemplo en el caso de las aristas su tamaño puede variar si hay varias ocurrencias de enlaces entre los dos mismos nodos.

Ofrece la posibilidad de customizar todo el grafo. También posee funciones físicas las cuales aplicadas correctamente podrán organizar el grafo de diferentes maneras, ya sea por grupos, comunidades para darle mejor estética.

Básicamente el objetivo de Gephi es que los analistas de datos que utilicen la herramienta puedan descubrir patrones, estructuras, llegar a hipótesis, etc, con la exploración, entendimiento y análisis de los grafos.

En este trabajo se utilizará Gephi para la última parte de la investigación. Cuando hayamos obtenido una hoja de cálculo formada por las relaciones entre las dos columnas con los dominios entrantes y salientes, se realizará un social network analysis al final del proyecto, utilizando como apoyo esta herramienta.

3.3 Obtención de enlaces base

En todo análisis de datos se necesita un conjunto de datos como entrada para que puedan ser investigados y analizados. Es por esto que la primera fase del trabajo consiste en la obtención de una base de enlaces de noticias que puedan servir como estudio, que sean la prueba sobre la que realizar la investigación.

El estudio de este trabajo está enfocado a la influencia de las fake news en marcas en la web. Es por esto que para poder realizarlo será precisa por una parte un conjunto de dominios conocidos o etiquetados como fake sobre el que se puedan investigar los artículos que contienen y por otra parte una lista de marcas de móviles/tecnológicas sobre las que buscar noticias relacionadas dentro de estos dominios para darle la temática particular de las fake news en el caso de las compañías de móvil. Ya que es algo que hoy en día no es tan visible como lo puedan ser las fake news relacionadas con política que son el foco principal en la gran mayoría de páginas de fake news y bulos, pero que existe entre las marcas como marketing digital.

La lista de marcas sobre las que se buscarán noticias fake será un conjunto de las empresas más importantes del sector tecnológico: Apple Iphone, Amazon, Blackberry, Facebook, Microsoft, Nokia, Samsung, Siemens, Sony y Twitter.

Para encontrar noticias se han utilizado como apoyo tres datasets encontrados por internet y que forman parte de repositorios de Github:

- Bs Detector es una extensión de Google Chrome y Mozilla Firefox que al navegar por una determinada página web indica si el contenido de dicha página puede ser de dudosa veracidad. En el interior del repositorio hay un archivo sobre dominios de páginas web fake que se utiliza para entrenar la herramienta y que se utilizará para este trabajo[1].

- Repositorio para la recolección de datos para la investigación de noticias falsas[2].

- Dataset formado por millones de artículos de noticias extraídos de una lista fuente de dominios[3].

De estos tres datasets se han extraído un conjunto de 35 dominios marcados como fake news, que no estuvieran caídos y que tuvieran noticias relacionadas con este tipo de temática tecnológica, ya que algunas solo están centradas en política.

Una vez claros los dominios y marcas a utilizar, se explicará la metodología para este

1 https://github.com/bs-detector/bs-detector
2 https://github.com/KaiDMML/FakeNewsNet
3 https://github.com/several27/FakeNewsCorpus

paso. A la hora recuperar artículos en estos dominios fake relacionados a las palabras de las compañías tecnológicas se ha optado por realizar un pequeño programa que realice peticiones de búsqueda personalizadas a google. Estas peticiones serán lanzadas con un tipo de url compuestas por varias partes para así obtener solo resultados relacionados con la marca de móvil indicada y que busque solo en el dominio que se le pasa.

Primero se extraerá la parte de la url base que se utiliza para realizar consultas al buscador:

https://www.google.es/search?q= +

Segundo indicaremos la palabra sobre la que queremos encontrar resultados, en nuestro caso al ser marcas tecnológicas optamos por Microsoft por ejemplo:

\+ Microsoft +

A continuación irá otra parte estática formada por el símbolo '+' para indicar un espacio seguido de la palabra site: que es un comando para que busque solo sobre el dominio que le pasamos a continuación y por último los caracteres '%3A' que representaría el símbolo ortográfico de los dos puntos ":":

\+ +site%3A +

Para terminar se adjunta al final el dominio sobre el que se desea realizar la búsqueda, en nuestro caso se tomará como ejemplo la página web rightwingnews.com:

\+ rightwingnews.com

Conjuntamente la url sería la siguiente:

https://www.google.es/search?q=Microsoft+site%3Arightwingnews.com

Que devolvería estos resultados:

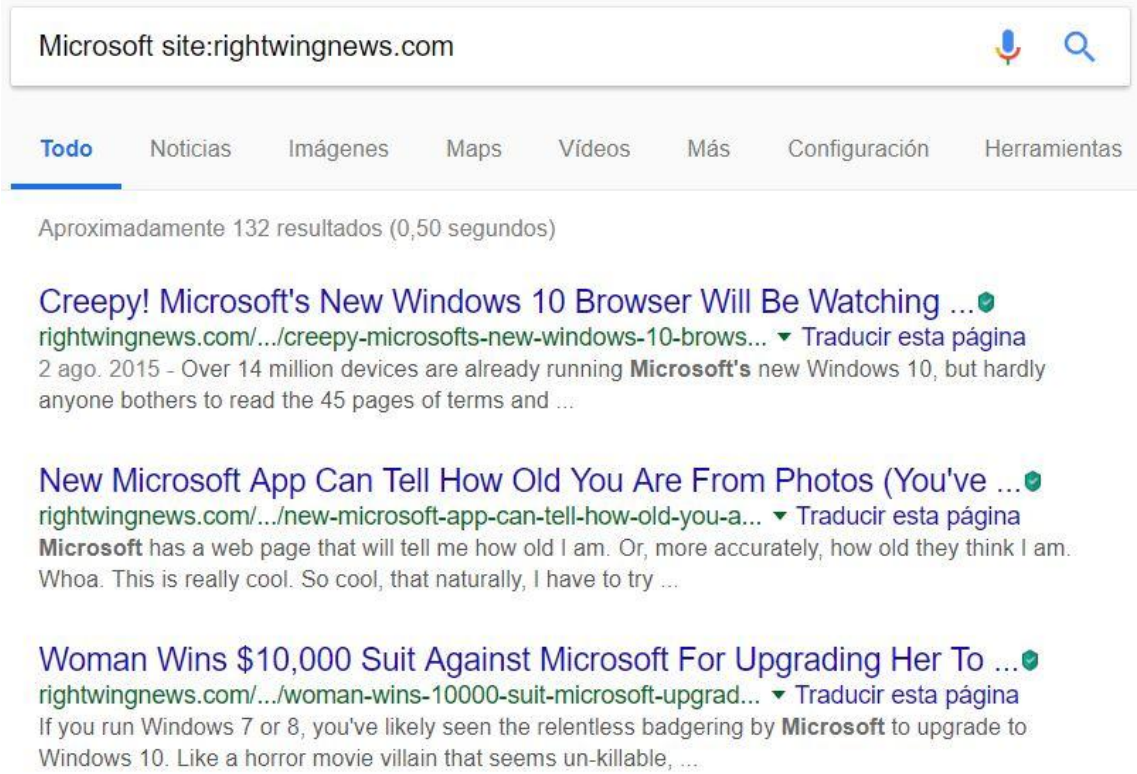

Figura 13. Resultados búsqueda personalizada.

Al realizar la búsqueda se obtendrá una lista de enlaces que contienen la palabra Microsoft dentro del dominio rightwingnews.com. Al ser una página dedicada a la difusión de noticias fake, todos o casi todos serían artículos de noticias. Puede que en algunas palabras determinados dominios devuelvan por ejemplo la palabra “Apple” en relación a la propia fruta, en estos casos como el de Apple se realizará la búsqueda con las palabras Apple Iphone, para evitar en la medida de lo posible estas situaciones.

Antes de obtener la lista de enlaces con noticias falsas hay que mirar qué dominios nos sirven al contener este tipo de temática en sus artículos. Los datasets nos ayudan a discernir qué tipo de páginas web fake son, ya sean satíricas, conspiracionales, distorsión de realidad,etc, pero no sabemos si son multi temáticas o al contrario solo tratan un tema en concreto como suele ser la política en el mayor de los casos, y si la temática por la que buscamos no tiene relación directa con algún político, no se obtendrán apenas resultados. Por esto el primer paso será realizar un primer barrido con la marca tecnológica de más renombre para obtener los dominios sobre los que obtendremos enlaces de noticias falsas o posiblemente falsas.

Pasando a la parte de programación se han tomado la marca y el dominio como dos variables para que la búsqueda de enlaces sea automática. Así, se crean dos listas, una con los nombres de las marcas tecnológicas y la otra con el conjunto de dominios. Entonces se podría realizar un barrido con todas las posibilidades posibles entre estas dos y de este modo obtener el conjunto de enlaces que se utilizaría para más adelante sacar los enlaces salientes de cada enlace obtenido en este paso.

Para realizar este paso se ha realizado un pequeño programa script en Python el cual se encargará de realizar peticiones de forma automática a Google mediante Selenium.

Como se ha dicho, se utilizarán dos listas como entrada a la función, una llamada mobileBrands que contendrá las marcas de móviles a analizar y la siguiente lista será la de los dominios fake (en la captura solo se muestra una parte de los dominios utilizados como entrada):

```
url = ["conservativebyte.com","washingtonsource.org","clashdaily.com","yesimright.com",
       "politicot.com","smag31.com","americanpoliticnews.com","ladylibertysnews.com",
       "neonnettle.com","usafirstinformation.com","now8news.com","realnewsrightnow.com",
       "tdtalliance.com","beforeitsnews.com","conservativedailypost.com",
       "investmentwatchblog.com","pakalertpress.com","sputniknews.com",
       "thecommonsenseshow.com","thetruthseeker.co.uk","threepercenternation.com",
       "vigilantcitizen.com"]

mobileBrands = ["Apple Iphone", "Amazon", "Blackberry", "Facebook",
                "Microsoft", "Nokia", "Samsung", "Siemens", "Sony", "Twitter"]
```

Figura 14. Conjunto de dominios de noticias fake y marcas tecnológicas usadas en el proyecto.

Una vez tengamos ambas listas, se llamará a la función getBaseLinks, que realizará la función de búsquedas Google que se detallará a continuación:

```
def getBaseLinks():
    while(i < len(mobileBrands)):
        j = 0
        while(j < len(url)):
            urlSearch ='https://www.google.es/search?q='+mobileBrands[i]+'+site%3A'+url[j]
            driver.get(urlSearch)
            linksList = driver.find_elements_by_xpath("//a[@href]")
            for link in linksList:
                link = link.get_attribute('href')
                if(link not in hyperlinks):
                    domainRef = getDomain(link)
                    if(sameDomain('www.'+url[j], link)):
                        hyperlinks.append(link)
            j += 1
        i += 1
```

Figura 15. Función getBaseLinks().

Antes de llamar a la función se crea un objeto Selenium, en este paso es utilizado Selenium porque los enlaces de la búsqueda son cargados dinámicamente y no forman parte del html que podemos recoger al hacer la petición con la librería “requests” y luego transformarlo a html y buscar los enlaces con Beautiful Soup.

Que Selenium sea más lento que otras opciones y perjudique así el rendimiento del programa no es un grave problema, porque Google, por sus condiciones de servicio limita el número de peticiones que se pueden realizar por lo que al llegar a un número determinado las bloquea y se debe esperar unas horas hasta poder volver a realizar peticiones automatizadas hasta que se vuelva a llegar al límite otra vez.

Básicamente lo que hace la función es lanzar la petición con la marca de móvil y dominio especificado, todo ello dentro de dos bucles para que vayan recorriendo todas

las posibles combinaciones, realiza la petición y se le indica que busque todas las referencias del atributo href (hiperenlace que especifica la URL a la que la página se dirige al hacer click) por toda la página y las guardamos en una lista. Se realiza una iteración por toda la lista para ver qué links de toda la página deben guardarse y cuáles descartarse, ya que solo interesan los que tengan el dominio que le pasamos como parámetro.

Primero comprobamos que el link no está guardado ya en la lista de dominios, si no está ya guardado se pasaría a la siguiente condición, en la cuál la función sameDomain() verifica que el dominio del enlace que se está analizando es el mismo dominio que se le ha pasado al buscador de Google. Esto es necesario porque aunque el tipo de búsqueda personalizada que se realiza es para que busque solo en el dominio que se le pasa, en la página puede haber otros muchos tipos de enlaces que pueden dirigir a Google images, maps, Wikipedia, etc. O por ejemplo cuando los dos primeros enlaces son anuncios. Por está razón se comprueba que los enlaces sean del mismo dominio.

La salida de la ejecución de esta función nos devolverá una lista con todos los enlaces obtenidos. Con esta lista base de una manera muy simple la copiaremos y la introduciremos como input en la siguiente parte del programa y así se podrá realizar el tratamiento de los enlaces para poder extraer los enlaces salientes de cada dominio y analizarlos.

Durante el proceso es posible que algunos enlaces no dirijan a una noticia sobre dicho dominio y se refieran a otro tipo de contenido en la página, en este tipo de situaciones el posterior programa está programado para que en este caso no detecte ningún enlace saliente de dicha dirección en la mayoría de las situaciones, y así dicha página analizada no se tenga en cuenta en el análisis final.

3.4 Tratamiento de enlaces

El siguiente paso será realizar la búsqueda de enlaces salientes de todos los enlaces que se han obtenido. Para ello realizaremos varios pasos que estarán englobados todos en un script principal en el lenguaje de programación Python sobre el IDE Pydev en Eclipse. En resumen este script nos generará un archivo .csv con dos columnas, la primera estará formada por filas que tendrán el dominio que hay como entrada y la segunda columna tendrá todos los dominios saliente de dicho dominio que se le pasa como entrada.

En resumen el script estará formado por un bucle principal que recorrerá todos los enlaces que le pasamos como entrada (todos los enlaces base, del paso anterior) y de cada enlace lo analizará y sacará los hiperenlaces que contenga el texto de la noticia en su interior.

Esta parte del proyecto está estructurada en cuatro módulos (archivos, clases) Python diferentes para organizar mejor el trabajo y el código está mejor estructurado y tenga mayor visibilidad general:

-**main**: módulo principal desde el que se irán llamando a las funciones de los demás módulos. Y esto es porque se encuentra el bucle principal de tipo While que irá iterando todos los enlaces que se le pasen como entrada.

-**pageSetup**: tendrá todas las funciones para realizar la petición a las páginas web, transformar su contenido a un objeto Beautifulsoup para poder manejar su contenido, y también aquellas relacionadas con la extracción del dominio de un enlace para evitar que se devuelvan enlaces del mismo dominio que del enlace que se le pasa entrante.

-**articleSetup**:está dividido en dos funciones principales. En resumidas cuentas la primera está encargada de encontrar la pieza más grande de texto de una página web y la segunda para buscar todos los hiperenlaces de un artículo dentro de dicha página.

-**dataExtraction**: contiene las funciones relacionadas con la extracción de datos a una “hoja de cálculo” .csv. Las llamadas a esta función serán realizadas una vez el bucle principal haya terminado y los enlaces se encuentren en dos listas ya preparados para ser exportados a este archivo.

Antes de entrar a comentar paso por paso lo que realiza el bucle principal, se crearán tres variables:

-**urlSource**: lista de urls. Básicamente serán todas las urls obtenidas en el paso anterior y que utilizaremos como input de este programa.

-**sourceLinks** y **tragetLinks**: serán también dos listas, pero esta vez vacías de inicio. Su función será principalmente la de ir almacenando todos los links entrantes

(sourceLinks) y salientes (targetLinks) que se vayan encontrando iteración a iteración del bucle. Más adelante será explicado con más detalle su funcionamiento.

Por partes, este bucle principal estará formado por numerosas funciones en su interior que se irán llamando con el transcurso de la ejecución y que se detallarán a continuación:

// MAKE_SOUP ****

Primero se creará un objeto de tipo BeautifulSoup realizando la llamada a la función makeSoup(), pasándole como parámetro un enlace de la lista total de enlaces.

```
soup = makeSoup(urlSource[i])
```

Figura 16. Llamada función makeSoup().

Esta función se encargará de devolvernos el contenido de dicho enlace, por lo tanto primero realizará una petición mediante el módulo requests al enlace especificado, le añadiremos un atributo cabecera (headers) que le pasaremos como parámetro a la llamada de requests.get() simulando que esta petición la realiza una persona humana y no un bot, más que nada porque que en algunas páginas se devolverá el código de error 403 (prohibido), en el cual se nos denegará el acceso a dicha página.

Segundo, una vez obtenido el contenido, crearemos un objeto de tipo BeautifulSoup, a partir del cual se podrá utilizar para navegar por todo la página, buscar enlaces, etc. Una vez creado el objeto se retornará y se guardará en la variable soup del bucle principal.

// GET_DOMAIN ****

De todos los enlaces que vamos a analizar necesitamos conocer su dominio, porque cuando realicemos el análisis de todos los enlaces salientes que hay en un enlace necesitaremos descartar aquellos enlaces salientes que redirijan a la misma página. Estos enlaces no interesan porque no serán representativos en el análisis de datos posterior en Gephi, añadirían ruido a las interconexiones entre las diferentes páginas que sería lo más relevante, son innecesarios.

Para obtener el dominio se ha creado la función getDomain(), a la cual le pasaremos como parámetro la url de la que queremos extraer el dominio y cuyo resultado guardaremos en la variable de tipo string: domain.

```
domain = getDomain(urlSource[i])
```

Figura 17. Llamada función getDomain().

```
def getDomain(urlSource):
    extract = tldextract.extract(urlSource)
    return extract.domain+'.'+extract.suffix
```

Figura 18. Función getDomain().

En primer lugar, esta función creará un objeto de tipo tldextract pasándole la url (urlSource[i]) como parámetro. Una vez creado el objeto se devolverá el dominio de la concatenado al sufijo de la url.

// FIND_LONGEST_TEXT ****

Para encontrar todos los enlaces incluidos dentro de un artículo, primero tendremos que detectar todo el texto que corresponde al artículo dentro de la página, porque dentro del contenido de una página hay texto que proviene de las cabeceras, de artículos relacionados, o de la sección de comentarios, por ejemplo, y el cual no interesa al estudio porque solo queremos obtener los hiperenlaces que están dentro del artículo y forman parte de la noticia.

Para ello se ha optado por realizar la función findLongestText(), la cual recibe como entrada la variable soup que se ha instanciado previamente y que se utilizará para buscar los hiperenlaces y el texto de la noticia. La función nos devolverá como salida el nodo en el cual se encuentra obtenido dentro de este, todos los nodos que contendrán en su interior el texto y los hiperenlaces del artículo, es decir, retornará el nodo que representará el artículo de la noticia, aquí se expone un ejemplo de un artículo real.

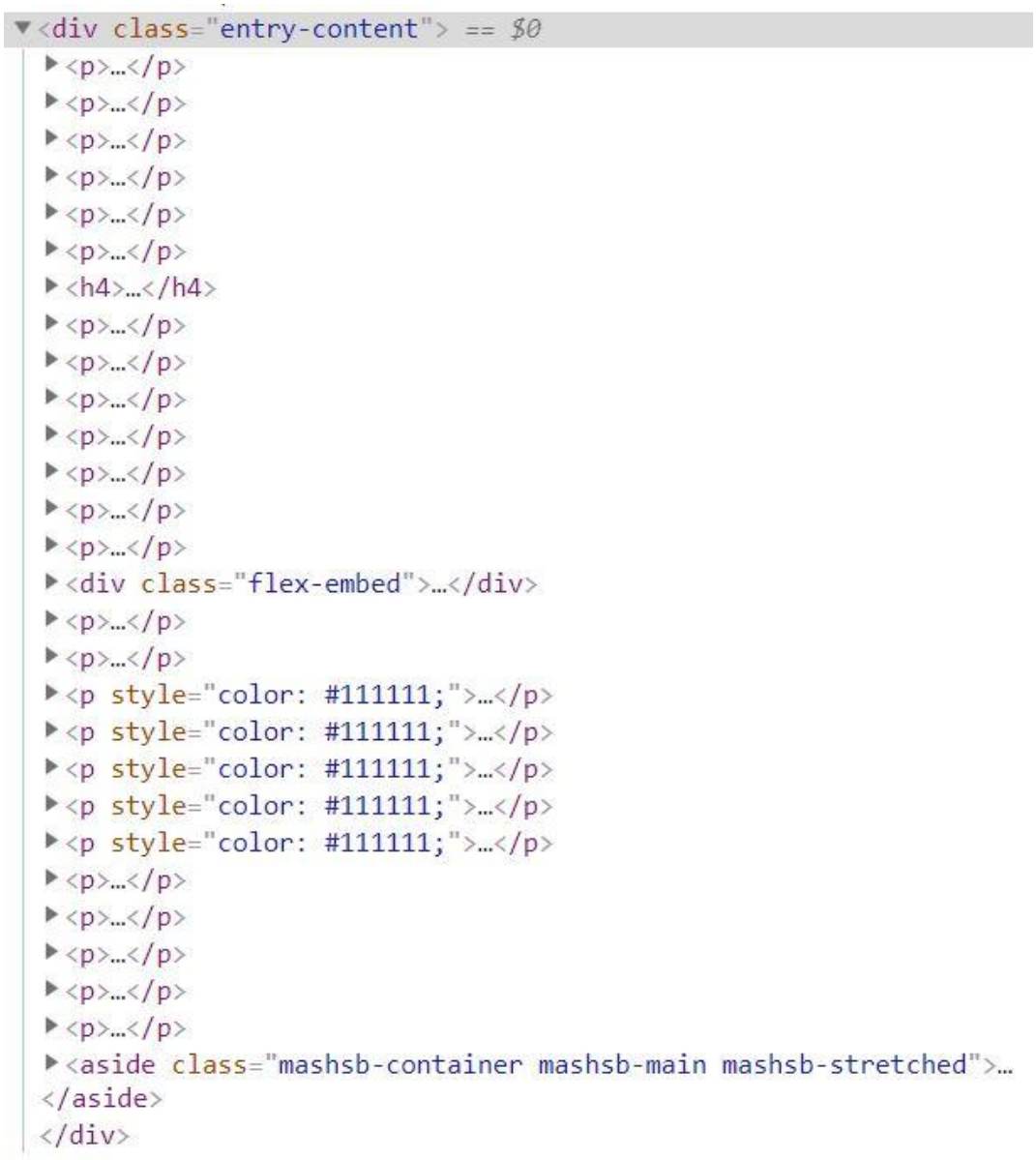

Figura 19. Estructura de una noticia de un dominio fake. Fuente: 21stcenturywire.com.

Cada nodo etiquetado con <p> representaría aparentemente (aparentemente porque cada página utiliza un esquema diferente y el texto puede estar contenido en un nodo diferente. No sería un grave problema porque la búsqueda de texto no se realiza por la etiqueta sino por texto plano)un párrafo de texto en el artículo, dentro de cada uno podría haber texto plano, hiperenlaces,etc. El objetivo es encontrar el nodo padre a partir del cual se pueda buscar todos los enlaces contenidos que son hijos de este nodo. Para ello primero se habría de rastrear el texto plano que hay en una página web e ir comprobando cada “trozo de texto” hasta encontrar uno que sea el más largo pero que también cumpla unas ciertas condiciones que se explicarán más adelante. En este caso el nodo “Div class=’entry-content’” sería el nodo padre objetivo.

En resumen esta función está programada para que primero encuentre todos los nodos con texto plano con la llamada a la función “soup.findAll(text=True)”, todas las ocurrencias que encuentre las guardaremos en una lista.

Esta lista la iteramos dentro de un bucle 'for' para ir descartando el texto que no cumpla las condiciones a través de los diferentes 'if' anidados que hay dentro del bucle.

```
for str in texts:
    if(isTextElement(str) == True):
        actualText = str
        if(len(actualText) > len(previousText)):
            #nodeInArticle = str
            if(nodeInsideArticle(str) == True):
                nodeInArticle = str
                previousText = actualText
    longerText = previousText
```

Figura 20. Bucle 'for' de la función findLongestText().

Dichas condiciones son:

- El texto plano no debe formar parte de una etiqueta la cual pueda ser de: 'style', 'script', 'head', 'title' o 'blockquote' para descartar todo el texto que pueda provenir de nodos con etiquetas no pertenecientes al artículo, ya sea el texto del título, la cabecera de la página, el texto perteneciente a un script, etc. Por ello la primera de las comprobaciones se realizará con la función isTextElement(), a la cual se le pasa el nodo que contiene el texto actual como parámetro (str) y retornará True o False, según el nodo forme parte de un nodo con dichas etiquetas o no.

```
if element.parent.name in ['style', 'script', 'head', 'title',
                           'meta', 'blockquote' , '[document]']:
    return False
```

Figura 21. Condición 'if' método isTextElement().

- La longitud del texto actual debe ser mayor al anterior analizado en la lista, ya que lo más probable, aunque obviamente no en el cien por cien de los casos, es que el texto con mayor longitud de toda la página se encuentre dentro de la noticia. Por ello se usa la condición de "len(actualText) > len(previousText)".

- La última condición es la más difícil de satisfacer, y es la que comprueba si el nodo con texto escogido y que ha pasado las otras dos condiciones está dentro del artículo. Esta función no funciona en el cien por cien de los casos ya que no se basa en el contenido del texto si no en la estructura de la página. La realización de este último paso ha sido motivada por la necesidad de evitar que sean escogidos nodos que forman parte de comentarios, porque ha sucedido en un número pequeño de casos de entre el 5%-15% en los que la función escoge como nodo a partir del cual realizar la búsqueda de enlaces, un comentario u otra parte de la página.

Lo que pretenden estas condiciones es determinar que el nodo de texto esté dentro de un esquema de nodos propio de un artículo de texto y no de un comentario.

Por ello se ha realizado la función con pesos nodeInsideArticle() basada en la observación empírica de otras páginas en las que el programa había fallado para comprobar que la mayoría de veces se escoge un nodo del artículo y no de cualquier otra parte de la página.

Está función devolverá True (se aceptará el nodo) en el caso de que cumpla dos de las tres condiciones que tiene la función. Estas condiciones son:

1. Que el nodo padre del nodo de texto que se analiza tenga al menos tres hijos.
2. El nodo padre debe estar rodeado de dos nodos directamente hermanos con texto plano que superen una cierta longitud. (Esta condición solo se cumple si la primera lo hace). Esta sería la condición más subjetiva y variable por el valor numérico a partir del cual se considera un nodo válido o no. Este valor puede suponer el elegir como nodo de texto una parte que se encuentra en el artículo o está por otra parte de la página.
3. El nodo padre debe tener al menos un hiperenlace en su interior.

Una vez haya un nodo que haya transcurrido por todas las condiciones, obtendremos el nodo con el párrafo de texto más largo de todo el artículo. De dicho nodo obtendremos el nodo padre que será el que contenga en su interior todos los nodos de texto de la noticia, y sobre el que tendremos que buscar todos los hiperenlaces.

En el caso de que la función no haya encontrado ningún nodo que cumpliera todas las condiciones se determinaría que el enlace de la página web no es una noticia y devolvería None (Null) descartando el enlace y pasando al siguiente en la lista que contiene todos los enlaces de entrada, urlSource.

// GET_ARTICLE_HYPERLINKS ****

En caso de que se obtuviera un nodo “válido” en el paso anterior, se utilizará para buscar todos los enlaces salientes, pues dicho nodo contendrá en su interior todos los atributos ‘href’ que a la vez estarán incluidos en otros nodos repartidos por el artículo y que indicarán los enlaces que tiene y que podremos buscar gracias a BeautifulSoup.

Para ello este método recibe como parámetros dos variables:

- nodeInArticle, será el nodo que utilizaremos como base para buscar en su interior todos los enlaces.

- domain, será el dominio del enlace que estamos utilizando como entrada en esta iteración y del que hemos obtenido el dominio mediante la función getDomain() anteriormente desarrollada.

Esta función devolverá una lista de strings con todos los hiperenlaces que haya encontrado en el interior del nodo nodeInArticle.

```
def getArticleHyperlinks(nodeInArticle, domain):
    #Get all the links of the article
    linksArticle = []
    for a in nodeInArticle.find_all('a', href=True):
        url = a['href']
        #Avoid repeated links
        if(url not in linksArticle):
            #To avoid difference between http o htpps
            if(url[0:7]=='http://' or url[0:8]=='https://'):
                if (sameDomain(domain,url) == False):
                    linksArticle.append(a['href'])
```

Figura 22. Función getArticleHyperlinks.

El primer paso de esta función será crear una lista vacía de tipo string llamada linksArticle, que contendrá todos los enlaces salientes que puedan haber dentro de dicho nodo, y que se irá rellenando según un bucle 'for.

Este bucle iterará sobre todos los atributos 'a' (etiqueta que sirve para definir un hiperenlace) que se encuentren dentro del nodo sin importar la profundidad a la que se encuentren, y cuya etiqueta 'a' contenga un 'href' dentro (href = True), cuyo valor será el hiperenlace al que apunta. En otras palabras iteraremos todos los enlaces incluidos en el nodo que le hemos pasado como parámetro a la función getArticleHyperlinks.

De cada atributo 'a', obtenemos el enlace con a['href'] y lo guardamos dentro de la variable url -> "url = a['href']".

La primera condición (el primer if) será la de comprobar que en la variable linksArticle que guardará todos los enlaces salientes, no tenga enlaces repetidos ya que una noticia con varios enlaces salientes que sean iguales, en el análisis que se realizará posteriormente en Gephi, a un nodo que representa un dominio se le podría dar una mayor importancia de la que realmente tiene (aumentando su tamaño visualmente), por la repetición de dichos enlaces. Aunque es algo poco común.

A continuación verificaremos, en el siguiente if, que el enlace saliente es un enlace que direcciona a una página web y no es otro tipo de enlace como un vídeo incrustado en el artículo o una foto, un enlace para enviar un mail ("mail.to/..."), etc. Como los navegadores utilizan las dos tipos de peticiones http y https, se programa para que tenga en cuenta ambas a la hora de confirmar o descartar la dirección, en la cual no sea un link a una página web.

La última condición es la llamada a una función, sameDomain(), condición para verificar los dominios entrantes y salientes son diferentes, pasándole el dominio del enlace entrante como parámetro (domain), y la url que se está analizando en este momento (HREF). Este método devolverá True si considera que a pesar de tener diferente dirección comparten el mismo dominio, y False en caso contrario. Esta comprobación se hará utilizando la librería comentada anteriormente, tldExtract, en la función sameDomain() para así obtener el dominio del enlace y compararlo con el

enlace anterior:

```
def sameDomain(domain,HREF):
    domainHREF = tldextract.extract(HREF)
    domain
    if(domain == domainHREF):
        return True
    else:
        return False
```

Figura 23. Función sameDomain().

// PARTE FINAL ****

Si la variable url que representa un enlace ha pasado por todas estas condiciones se añadirá a la lista linksArticle que será la cuál se utilice para ir volcando los enlaces iteración a iteración a la lista final targetLinks donde se almacenarán todos estos enlaces. Es decir, linksArticle solo tiene en cuenta los enlaces salientes de la actual iteración, solamente del enlace contenido en urlSource[i] y se sobreescribe el contenido de esta variable a cada iteración. En cambio en la otra variable, targetLinks, se almacenarán el total de los enlaces salientes de todo el análisis. Será la variable que se utilizará cuando se extraigan los datos en una hoja de cálculo. Así con linksArticle, podremos realizar la llamada a la función getArticleHyperlinks() y guardar el resultado en esta variable y también calcular el número de enlaces salientes de cada enlace midiendo su longitud.

Al tener estas dos variable cada iteración se volcarán los datos de esta manera en esta variable: “targetLinks += linksArticle”.

La hoja de cálculo final estará formada por dos columnas, las direcciones entrantes y salientes. La columna de enlaces salientes ya está lista por la variable targetLinks que contiene todos estos enlaces. Ahora se necesita calcular la otra lista sourceLinks, con todos los enlaces entrantes, de forma que cada enlace entrante tenga a su derecha (en la tabla) el conjunto de enlaces saliente a los que direcciona, por ejemplo:

sourceLinks -(lista por rellenar)	targetLinks(lista ya terminada)
rightwingnews.com	elconfidencial.es
rightwingnews.com	cnn.com
rightwingnews.com	nytimes.com

Figura 24. Tabla ejemplo de la estructura de datos final.

Por lo que habrá que repetir el valor del dominio del enlace actual en esta iteración, urlSource[i], un número de veces equivalente a la cantidad de enlaces salientes que se hayan detectado del enlace de la actual iteración: “numLinks = len(linksArticle)”.

numLinks es la variable que contendrá el número de enlaces salientes detectados para

este urlSource[i] calculados según el número de elementos en la lista inksArticle utilizando la función length.

Según el número que posea esta variable numLinks, se repetirá ese número de veces el domino de urlSource[i] en sourceLinks. Si el número es igual a 0, será porque no se ha encontrado ningún enlace saliente de esta dirección urlSource[i].

```
while numLinks > 0:
    sourceLinks.append(urlSource[i])
    numLinks = numLinks - 1
```

Figura 25. Bucle 'while' del bucle principal.

De esta manera cada enlace saliente estará tendrá a su izquierda el dominio de donde proviene, y quedarán listos para la extracción de todos los datos a la hoja de cálculo.

3.5 Extracción de datos.

// CSV_MAINDATA ** **

El último paso de este capítulo es la extracción de datos para su importación desde Gephi. Para ello trataremos las dos listas de enlaces sourceLinks y targetLinks como dos columnas en el archivo .csv final. Por eso llamaremos a la función csv_MainData(),que se explicará con más detalle más adelante, con las dos listas como parámetros, obteniendo como salida de la función la hoja de cálculo con las dos columnas, una ocupada por cada lista y que formarán las relaciones entre cada enlace entrante y saliente.

```
def csvMainData(sourceLinks, targetLinks):

    sourceLinks = extractDomain(sourceLinks)
    targetLinks = extractDomain(targetLinks)
    with open('domainRelations.csv', 'w', newline='') as csvFile:
        writer = csv.writer(csvFile)
        writer.writerow(['Source','Target'])
        i = 0
        for str in targetLinks:
            writer.writerow([sourceLinks[i],str])
            i += 1
```

Figura 26. Función csvMainData().

Para que la visualización en Gephi no sea molesta por la cantidad de nodos, los enlaces son acortados para que muestren solo el dominio, el netloc, eliminando la información redundante, como por ejemplo:

https://www.elmundo.es/ **->** elmundo.es

Para ello la primera parte de la función serán dos líneas dedicadas a la extracción del dominio de ambas listas de enlaces mediante la función exctractDomain(), una función muy sencilla que tomará como parámetro una lista, iterará con un bucle 'for' todos los elementos que se encuentren en ella y a cada uno le aplicará la función getDomain() explicada anteriormente, para extraer el dominio de todos los links de la lista, tanto a sourceLinks como a targetLinks, ya que dichas variables las sobreescribimos al llamar a esta función, extractDomain().

```
def extractDomain(urlList):
    i = 0
    for str in urlList:
        urlList[i] = getDomain(str)
        i += 1
    return urlList
```

Figura 27. Función extractDomain().

Con ambas listas normalizadas al dominio se realizará la escritura en la hoja de

cálculo digamos, primero se le indica que queremos crear un archivo de escritura ('w') llamado domainRelations.csv. Luego crearemos un objeto de tipo escritura el cual convertirá los datos de las listas en strings dentro del archivo domainRelations que hemos creado.

Se escriben las cabeceras que tendrá la tabla, la primera será Source y la segunda Target, especificando la relación dominio de entrada (Source) -> dominio saliente (Target). La orden que se utilizará para escribir una fila en el archivo será writerow, con las cadenas 'Source' y 'Target' como parámetros para que escriba en la primera fila las cabeceras:

A continuación ya solo queda rellenar la tabla con ambas listas, para ello iteraremos con un bucle 'for' todos los strings de la lista targetLinks, también es válido realizarlo iterando la lista de sourceLinks. Utilizaremos la variable i para ir extrayendo todas los dominios de las listas y mediante la función writerow, se irá rellenando la tabla fila por fila según ambas listas.

```
i = 0
for str in destinyLinks:
    writer.writerow([sourceLinks[i],str])
    i += 1
```

Figura 28. Bucle 'For' función csvMainData().

Una vez finalizado el proceso se habrá creado un fichero llamado domainRelations.csv en el directorio del proyecto con todos las relaciones entre dominios que utilizaremos para el análisis siguiente en Gephi.

Con la terminación de esta última parte se habrá terminado la ejecución del programa, pero aparte de haber obtenido un fichero .csv se habrán obtenido una lista total de enlaces salientes. Está lista la volveremos a utilizar (ejecutando otra vez el programa) como input del paso 3.2 (tratamiento de enlaces) para volver a realizar este procedimiento y obtener otra lista de enlaces salientes de los propios enlaces salientes que habíamos obtenido en la iteración general anterior.

Este proceso lo repetiremos tres veces para obtener posibles relaciones entre dominios de fake news o posibles relaciones entre ellos a la hora de apoyarse en la redacción de noticias. Una vez hayamos terminado todo el proceso se juntarán manualmente todos los archivos .csv en uno solo con el total, que será el que se utilizará en Gephi. En resumen lo que se está realizando es aumentar la profundidad de búsqueda de enlaces a tres niveles de profundidad.

4. Importación de datos a Gephi.

La penúltima parte de la investigación será importar el archivo domainRelations.csv con Gephi. Se ha utilizado este formato .csv por la facilidad tanto de crear desde cero y escribir datos sobre él en Python tanto por ser un formato soportado por Gephi, el cual tiene en cuenta el peso de las aristas en la visualización del grafo, así que estás aristas serán más gruesas cuanto más conexiones haya entre dos dominios iguales.

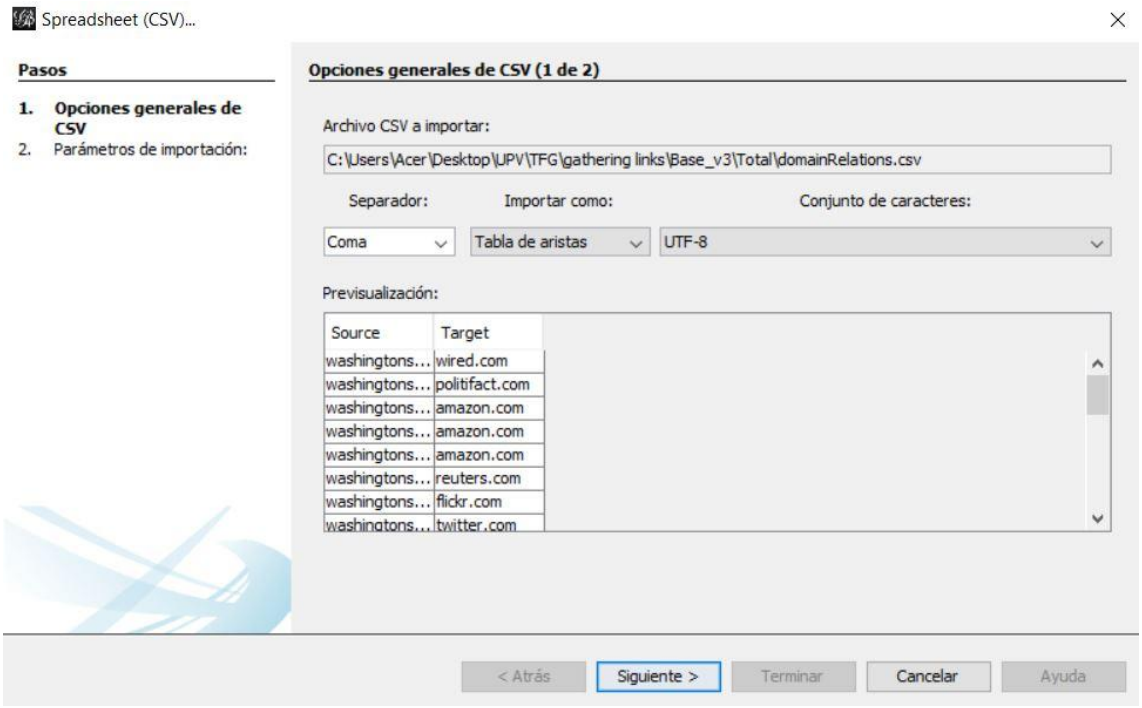

Figura 29. Página 1 importación datos a Gephi.

De una forma abreviada lo que se realizará en Gephi para tener una primera visualización de este archivo como un grafo será abrir el archivo domainRelations.csv, Gephi lo detectará como un archivo de formato .csv y nos mostrará dos pantallas para configurar el grafo. En la primera se le indicará la utilización de una coma para la separación de columnas ya que es el tipo de separación en este formato y es el que se ha utilizado en este caso; la importación será como una tabla de aristas para que construya el grafo con las conexiones entre las dos columnas y con la codificación UTF-8.

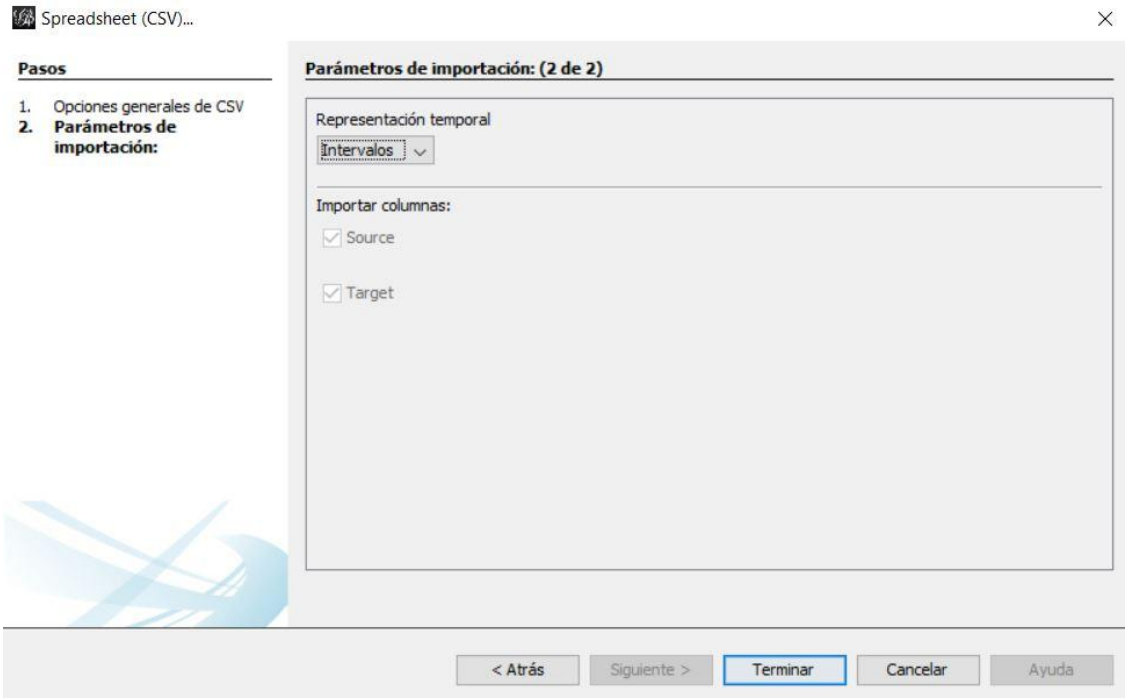

Figura 30. Página 2 importación datos a Gephi.

En esta penúltima pantalla se dejarán los opciones preestablecidas ya que ya están marcadas las opciones de importar las columnas Source y Target que se han indicado en la pantalla anterior. La opción de representación temporal está destinada para grafos dinámicos lo cual no será este caso y será irrelevante la opción a seleccionar.

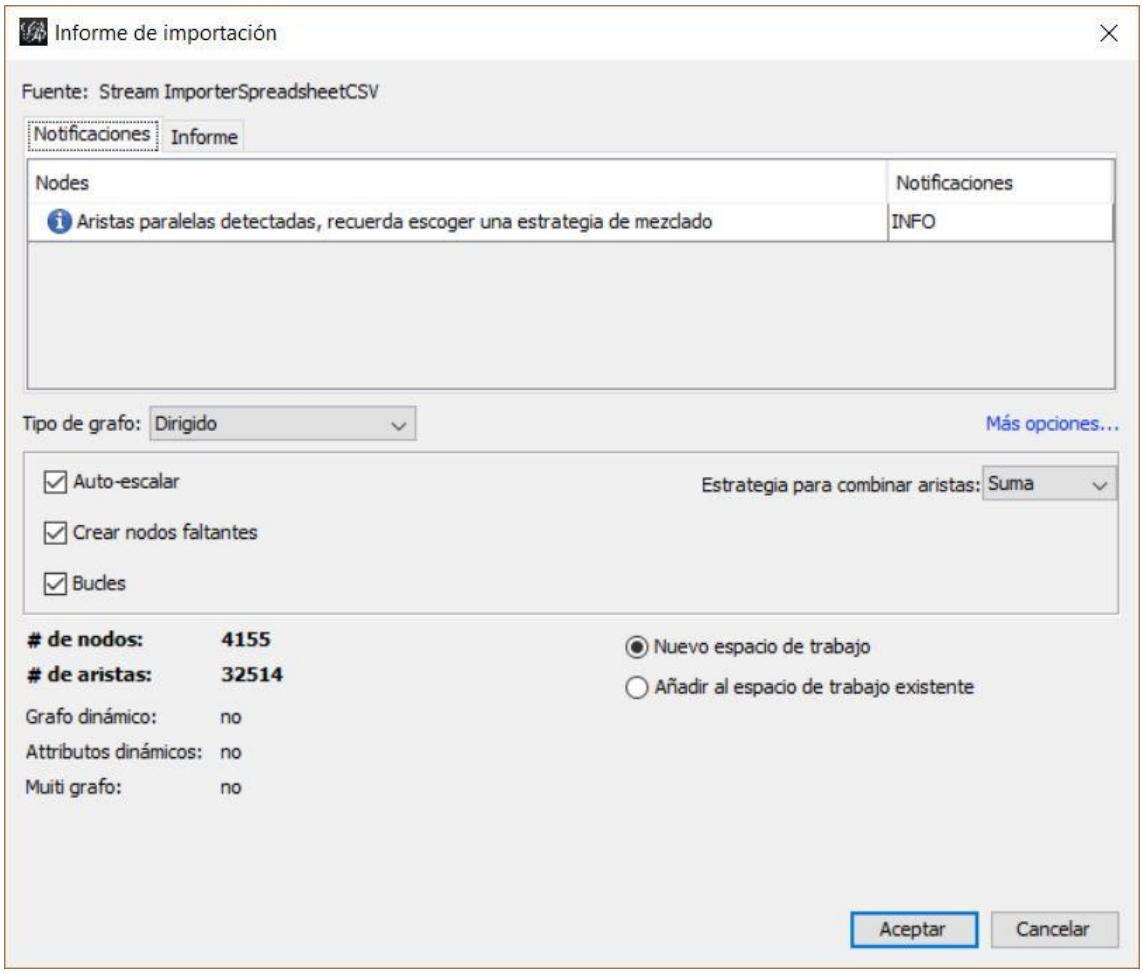

Figura 31. Informe de importación de datos de Gephi.

Por último antes de terminar la importación del grafo, se realizarán unos pequeños ajustes. Se le indicará que el grafo es dirigido, que con las aristas paralelas se efectuará como estrategia de mezclado una suma entre ellas para que así el tamaño de la arista varie. También se auto-escalará el grafo según el número de nodos y aristas que contenga, se permitirán los bucles y se crearán nodos faltantes encontrados como destino u origen de aristas. En el caso de que hubiera nodos faltantes, se indicarían estos nodos en las notificaciones. Como en el archivo domainRelations no hay nodos faltantes no hay ninguna notificación.

Al pulsar en “Aceptar” se generaría una versión, inicial, del grafo a partir del archivo domainRelations con todos los nodos del mismo color, tamaño, sin etiquetas y distribuidos de una forma aparentemente aleatoria. Para que el grafo tenga mejor apariencia primero se cambiará el tamaño y color de los nodos según el grado de entrada de estos, será según el grado de entrada para saber cuales son los dominios más redirigidos, también se les añadirá a los nodos las etiquetas, cada etiqueta representará el dominio, para que se puedan identificar y por último se les proporcionará una distribución para darle forma al grafo y que tenga una mayor visibilidad.

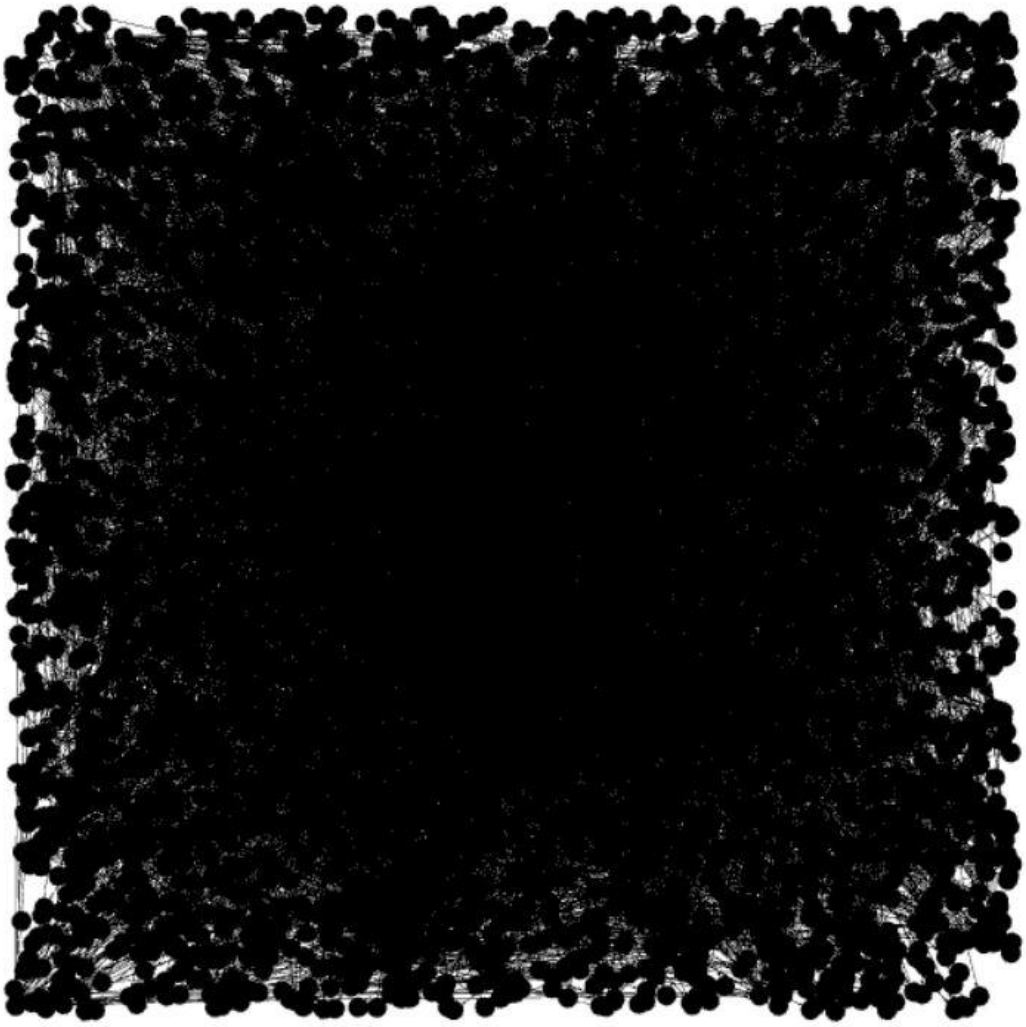

Figura 32. Grafo inicial.

5. Análisis

5.1 Análisis inicial, grado de entrada

Una vez importado los datos y teniendo el grafo listo se procederá a su preparación para mejorarlo visualmente. Primero se cambiará el tamaño de los nodos del grafo según el grado de entrada para obtener una visión primeriza general sobre los dominios que más han sido redirigidos. Para ello seleccionaremos un ranking para que el tamaño vaya variando según el grado de entrada y no sea fijo. Se fijará un intervalo de tamaño relativo entre 1 y 30.

También se añadirán las etiquetas a los nodos para poder identificarlos y se le cambiará el color a amarillo para que puedan ser más legibles teniendo un color diferente a los nodos y al fondo (el color de los nodos y de las etiquetas irá variando según el tipo de métricas que se irán aplicando más adelante). De igual modo se le proporcionará a las etiquetas un tamaño equitativo al tamaño del nodo al que representan.

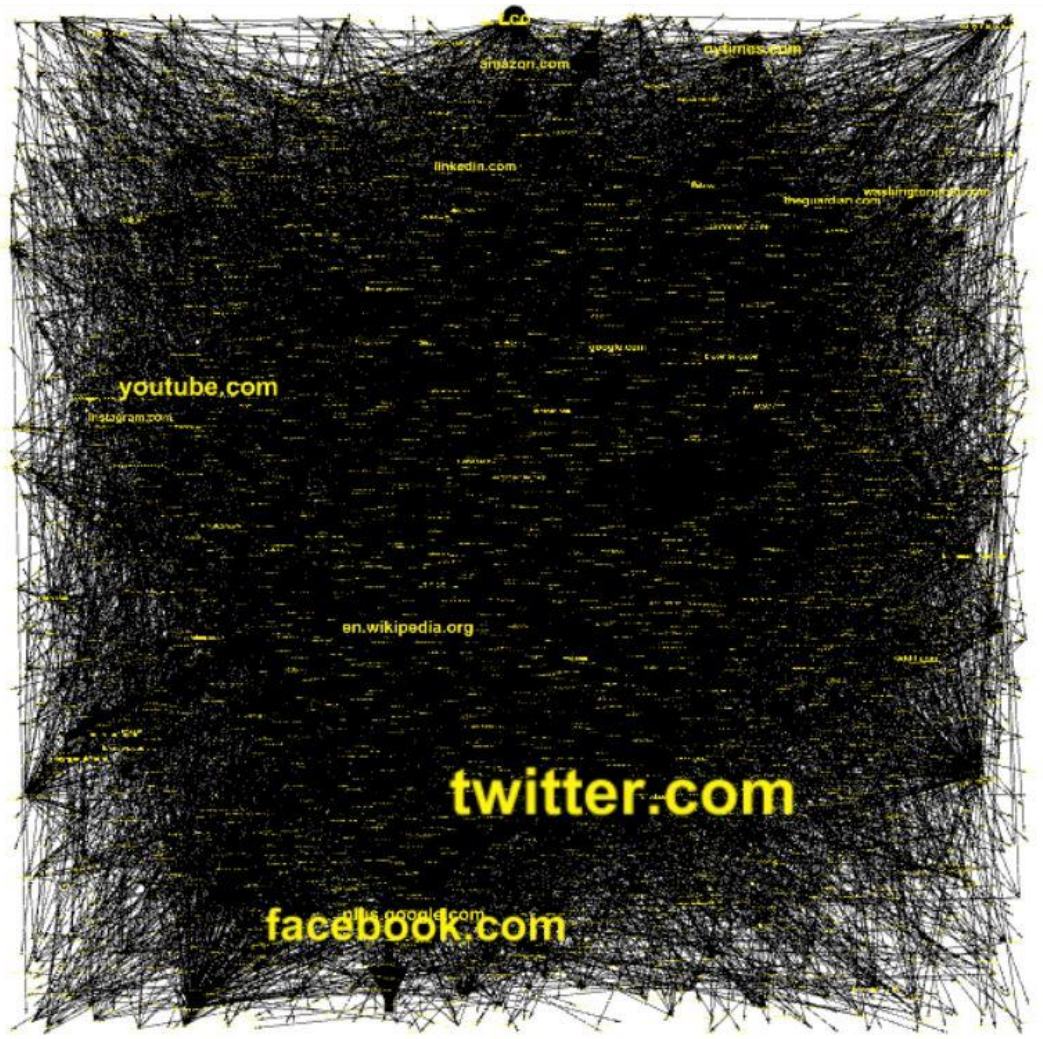

Figura 33. Grafo con cambio de tamaño de nodos y etiquetas.

Una vez aplicados los cambios se observa que el grafo ha mejorado algo su visibilidad ya que ahora se pueden apreciar qué nodos son más predominantes que otros por el tamaño.

Una vez listo el tamaño y las etiquetas en cada nodo, ya se pueden apreciar qué dominios son los más recurrentes como fuente o base para los artículos y en los cuales se apoyan a lo largo de la noticia. Ahora faltará reorganizar los nodos para darle una forma al grafo más estética que ayudará al posterior análisis, por lo que se pasará a la distribución de los nodos. Aunque más adelante se pueden volver a personalizar detalles relacionados con el aspecto como se puede hacer con el color para algo tan simple como el tamaño del nodo o para distribuir los nodos en comunidades diferentes, cada una de un color. Con la distribución se buscará que el grafo sea lo más legible posible, de tal manera que los nodos entrantes tengan mayor visibilidad y se aprecie las relaciones entre los nodos.

Para distribuir los nodos sobre el grafo se utilizará el algoritmo de Force Atlas 2, basado en la primera versión del Force Atlas pero diseñado para grafos con un mayor número de nodos ya que la complejidad del algoritmo es reducida y el rendimiento mejorado. Este algoritmo está diseñado para generar las llamadas Redes de mundo pequeño, tipo de grafo basado en el experimento del mundo pequeño del psicólogo Stanley Milgram que espetaba que se podía conectar a dos personas de Estados Unidos con solo seis personas intermediarias de media. Aplicando esto al grafo, sería uno en el cual la mayoría de nodos no serían vecinos entre sí y en cambio la mayoría de nodos podrían ser alcanzados desde cualquier nodo origen sin realizar un número elevado de saltos. En los parámetros de este algoritmo se modifica el escalado, que sería la repulsión entre los nodos para hacer el grafo más disperso. También se selecciona la opción “Evitar el solapamiento” para impedir que un nodo pueda esconder a otro.

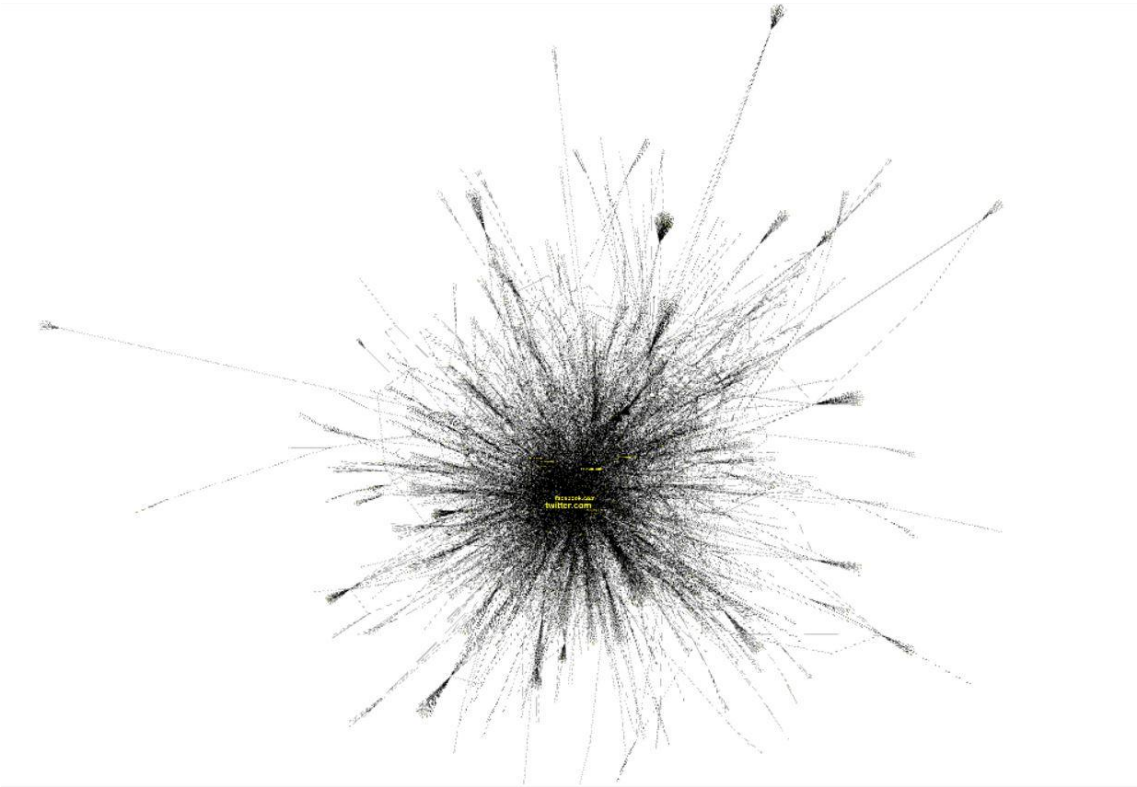

Figura 34. Grafo distribuido mediante el algoritmo Force Atlas 2.

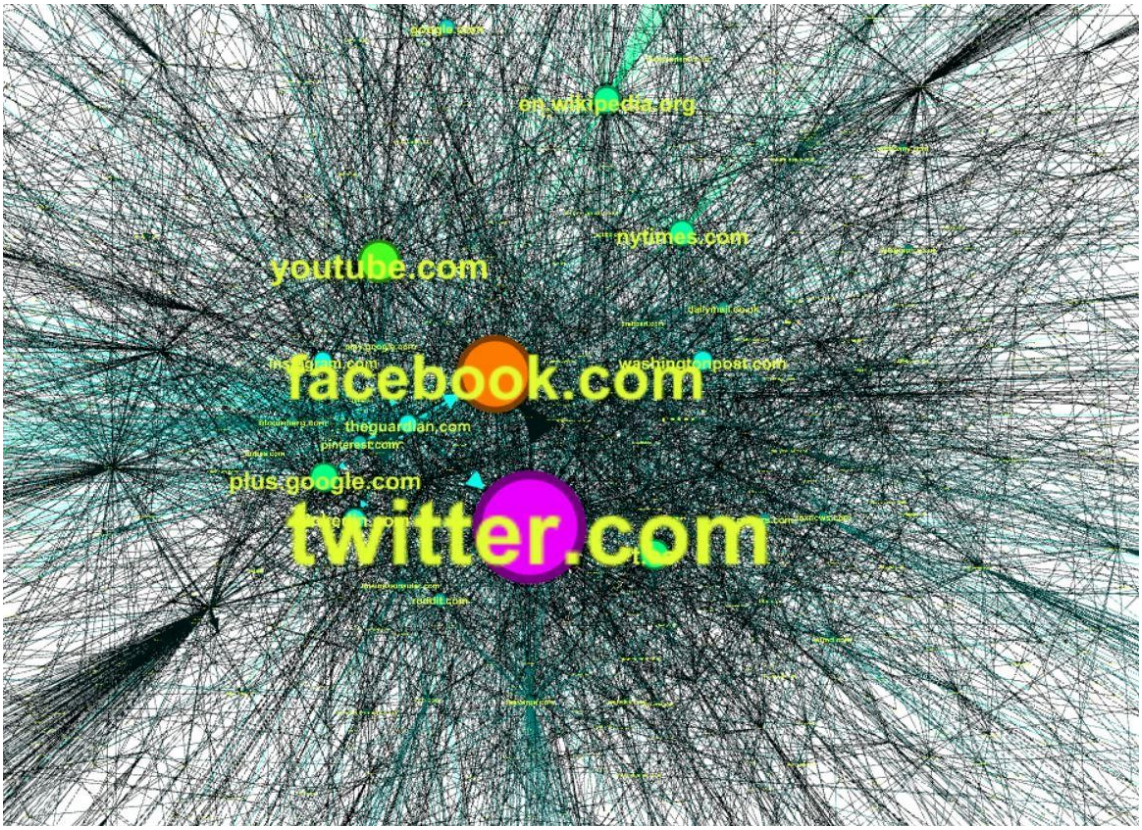

Figura 35. Grafo zona central.

Antes de entrar en estadísticas de la red, de nodos y aristas, el grafo generado después de aplicar el algoritmo Force Atlas 2 ha situado a los nodos que son utilizados

de una manera más o menos uniforme por todos los dominios falsos en una posición relativamente central al grafo, ya que la mayoría de dominios “beben” de estos nodos que se encuentran en el centro del grafo. Es por ello que los dominios como Twitter, Facebook, Youtube, Google plus, The Guardian, New York Times o Washington Post serían los más direccionados por la mayoría de páginas web fake. Eliminando a las redes sociales y a algún dominio especial como Wikipedia, se encontraría la mayoría de periódicos digitales online de más renombre en esta parte más central y que han sido bastante utilizados, si no su nombre no sería tan visible en el grafo.

Por otra parte los nodos más alejados del centro y de los cuales salen muchas aristas (tienen un grado de salida elevado) serían algunos de los dominios de noticias fake que se han utilizado como input del programa. Cuanto más alejados del centro del grafo están, menos utilizan estos dominios mencionados anteriormente y más otros tipo de dominios que en la mayoría de casos solo se direcciona a ellos desde este propio nodo, es decir, si hay un nodo, página web, que está muy alejado del centro indicaría que sus artículos direccionan a dominios que ningún otro dominio más utiliza como apoyo de sus artículos, o sólo muy pocos.

Este es el caso de lo que ocurre con el nodo que representa a vdare.com, uno de los más alejados del grafo. Lo curioso de este dominio es que no es uno de los que formó parte como dominios de entrada a partir del cual se buscaron noticias. Si no que es un nodo que ha aparecido por haber sido direccionado por otro dominio, además al buscar en los datasets con dominios de noticias fake ha aparecido marcado como un dominio fake de tipo “bias” (imparcial) y “hate” (odio). Como se puede observar en la siguiente captura, vdare.com direcciona a muchos dominios diferentes, aunque la mayor parte, la que se muestra en la captura, son sólo usados por vdare.com, es por esto que se encuentra más alejado del centro del grafo. A pesar de que la investigación que se está realizando sea sobre una temática en particular, los enlaces contenidos en “vdare.com” no se basan en las fuentes que otros dominios si puedan utilizar y que se encuentren más al centro.

Figura 36. Nodo "vdare.com".

Gracias a los algoritmos que organizan y distribuyen los nodos en el grafo para darle una forma más estética se puede llegar a conclusiones simples y algo subjetivas. Por lo que el siguiente paso será utilizar las métricas que proporciona Gephi para obtener más información ya más objetiva acerca del grafo.

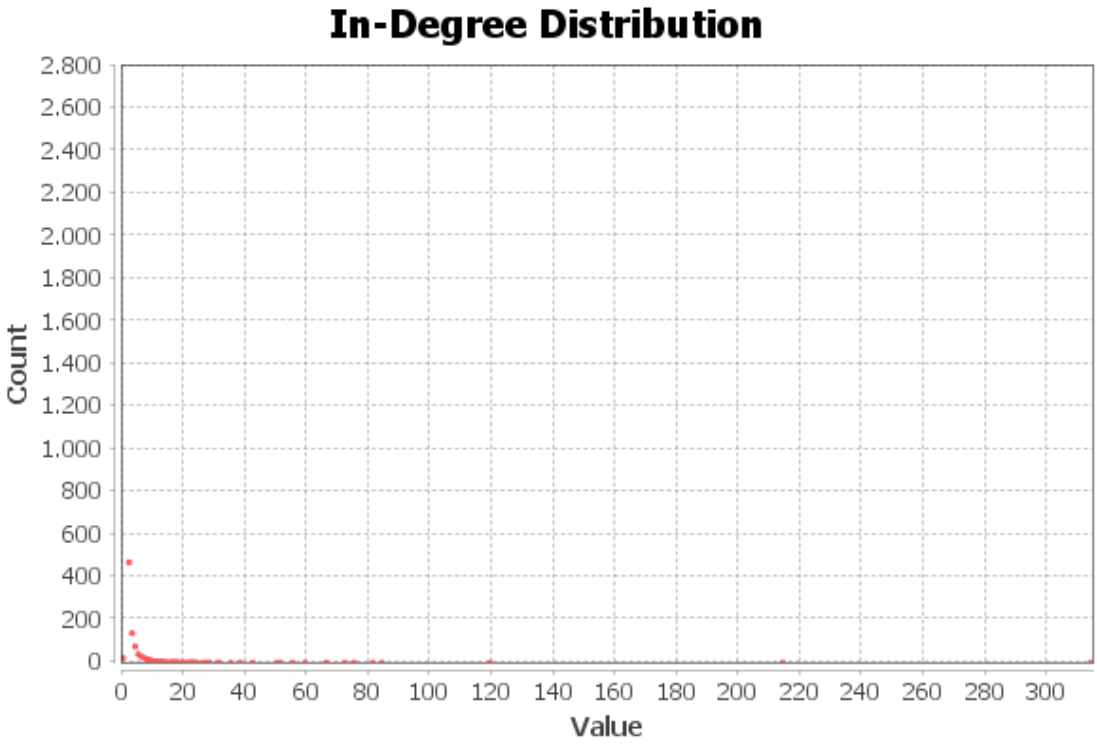

Figura 37. Gráfico distribución de los nodos según grado de entrada.

En este gráfica y en las siguientes el eje horizontal representa el valor que se está midiendo y el eje vertical el número de nodos con dicho valor. Por ejemplo en esta gráfica que mide la distribución del grado de entrada por el grafo, muestra que por ejemplo hay unos 3.000 nodos aproximadamente con un grado de entrada 1.

Esta gráfica corrobora parte de la información que hemos deducido antes y que el grafo muestra a simple vista. Hay muchos nodos con un grado de entrada muy bajo cercano a 1 (77,98%), 2 (11,36%) 3 3,37%) y 4 (1,85%) que en total sería un 94,43% del total de los nodos:

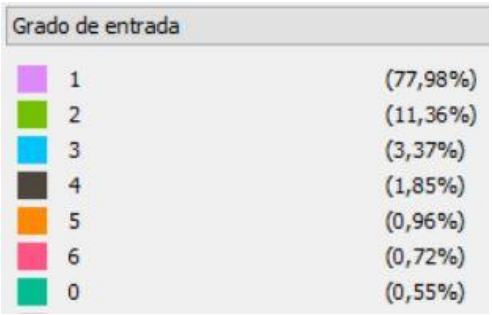

Figura 38. Distribución del valor del grado de entrada (de 0-6) según porcentaje del total de nodos del grafo.

Esto último mostraría que pese a haber realizado el estudio con una misma temática, la de las marcas tecnológicas, hay muchas noticias que se apoyan en dominios que son solo utilizados una vez, o muy pocas.

5.2 Análisis según grado de salida

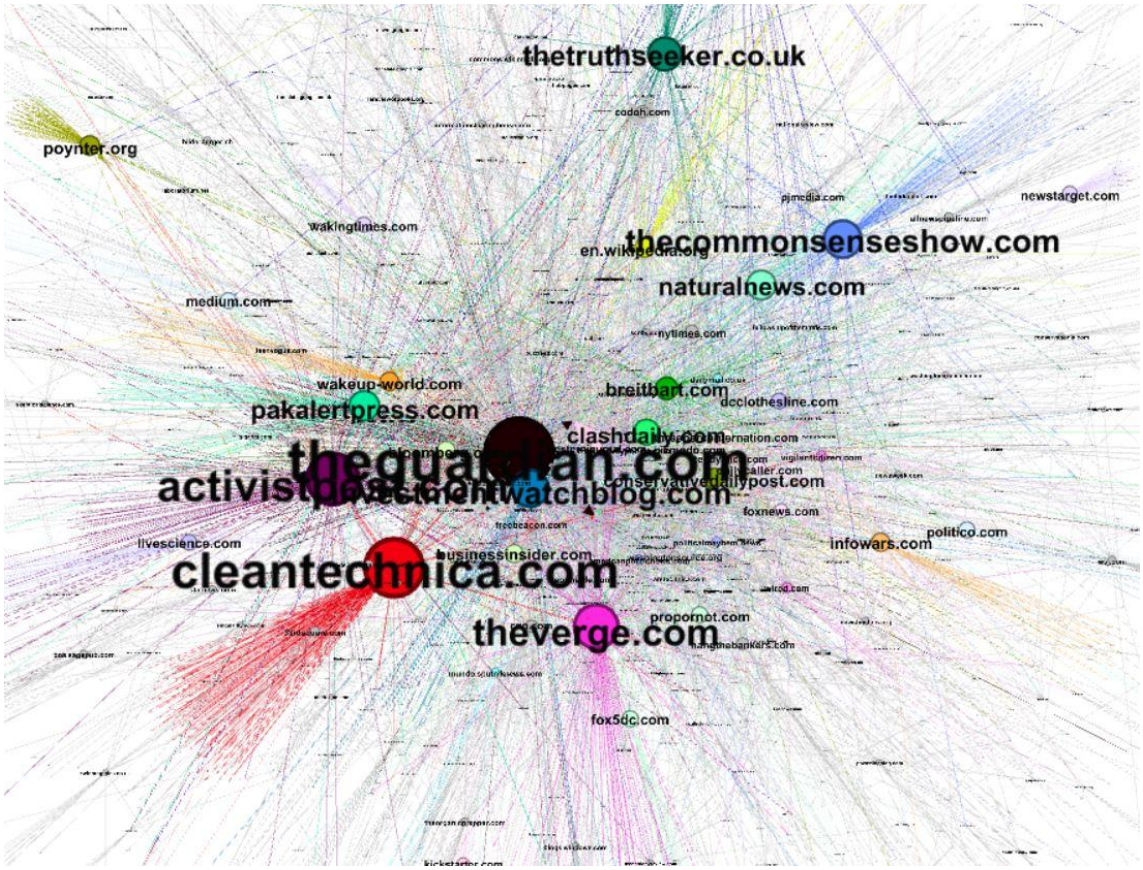

Figura 39. Grafo con nodos coloreados y tamaño según grado de salida.

Para este grafo se ha cambiado el tamaño de los nodos según el grado de salida y se han coloreado los 35 (ya que 35 fueron los dominios fake que se utilizaron para investigación) nodos con mayor grado. Entonces se ha realizado una investigación a cada nodo manualmente y se les ha clasificado en tres tipos según su etiqueta: dominio fake utilizado en la investigación, dominio fake no utilizado en la investigación (se ha corroborado que es un dominio fake al buscar dicho dominio en los datasets utilizados en el trabajo) y “otros” dominios, entre los otros dominios puede que haya algún dominio que también sea fake pero que no esté contenido en estos datasets, por lo que la clasificación puede que sea aproximada y no perfecta. Una vez se ha identificado a cada nodo de los 35 con mayor grado de salida de todo el grafo se ha realizado una tabla con el número de nodos y el porcentaje que representan de los 35 nodos en total con mayor grado de salida:

Tipo de nodo	Número de nodos	Porcentaje
Dominio fake utilizados	9	25,7 %
Dominios fake no utilizados	9	25,7 %
Otros dominios	17	45,6 %
Total	35	100 %

Figura 40. Tabla con la clasificación de los 35 nodos con mayor grado de salida del grafo.

Es muy llamativo el hecho de que uno de cada cuatro nodos de los 35 pertenezca a un dominio fake no utilizado y que haya aparecido por haber sido redirigido por otros nodos, posiblemente también fake. Los nodos no utilizados que han aparecido son: “activitypost.com”, “vdare.com”, “naturalnews.com”, “breitbart.com”, “infowars.com”, “dcclothesline.com”, “newstarget.com”, “wakingtimes.com” y “dailycaller.com”. Al haberse colado nueve nodos que no formaban parte de la investigación inicial, es muy probable que haya bastantes nodos de dominios fake repartidos por toda la red.

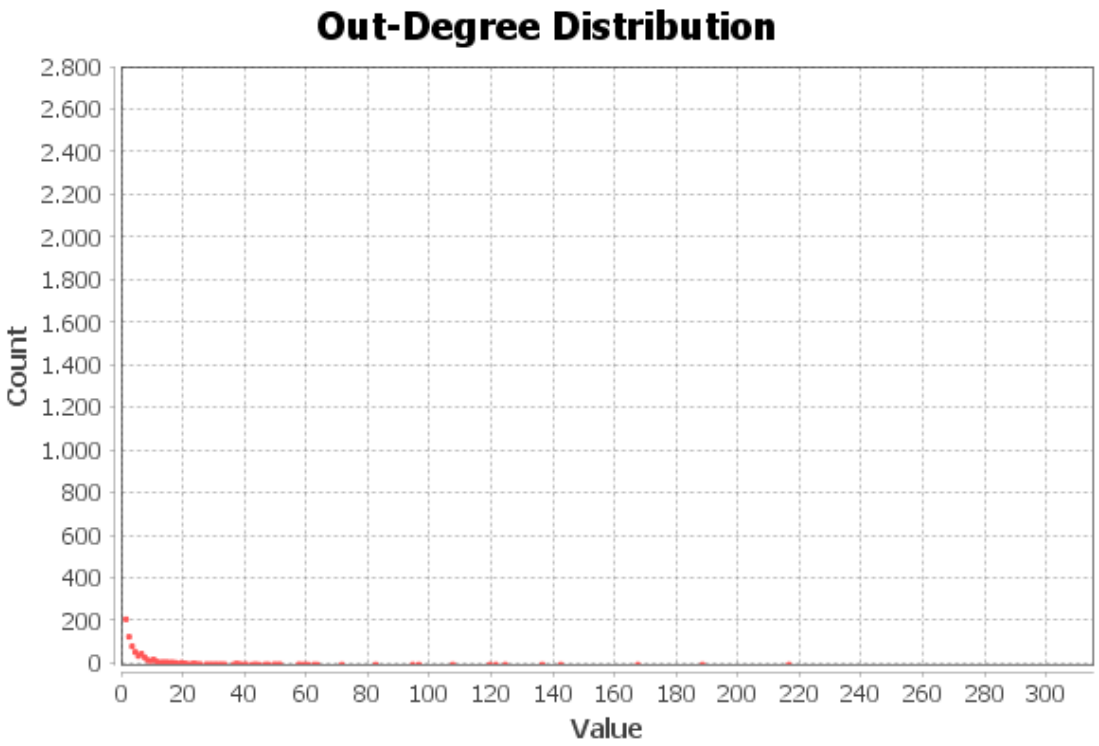

Figura 41. Gráfico distribución grado de salida.

Pasando a analizar la gráfica con la distribución de nodos según el grado de salida, la gran mayoría de nodos, un 90%, tendría un grado de salida de: 0 (78,82 %), 1 (5,15

%), 2 (3,18 %), 3 (2,09 %) o 4 (1,44 %). Esto hace presagiar que el grafo estará lejos de ser un grafo completo y de tener un coeficiente de agrupamiento elevado que pueda predecir la formación de comunidades entre dominios fake. Habrán muchas partes de la red con nodos hoja (grado igual a 1), que tendrán un grado de entrada 1 y grado de salida 0, ya que en la métrica anterior el porcentaje de vértices con grado de entrada 1 era del 77,98%, mientras que en esta el número de vértices con grado de salida 0 es 78,82.

Esto último corrobora lo deducido anteriormente, hay muchos dominios (ya puedan ser fake o no) que se apoyan en páginas que son solo utilizadas una sola vez y de las cuales no salen más enlaces. Bien es cierto que si el análisis se realizará con un conjunto de dominios fake mucho mayor de 35, es probable que no fuera tan común el gran número detectado de nodos hoja. Se puede observar esta última parte desde una vista general del grafo con este tipo de nodos hoja, con estos dos ejemplos:

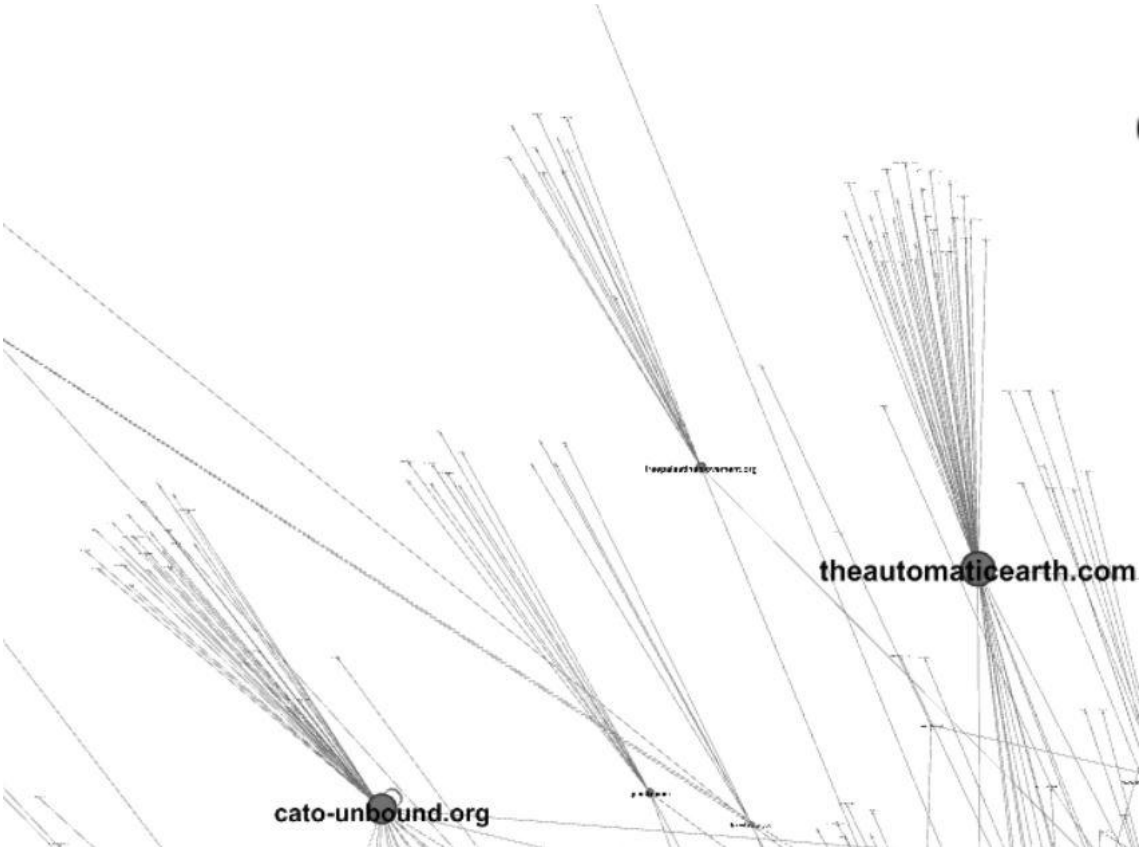

Figura 42. Parte externa del grafo formada por muchos nodos hoja.

Figura 43. Parte del grafo formada por una gran cantidad de nodos hoja.

5.3 Coeficiente de clustering (agrupamiento)

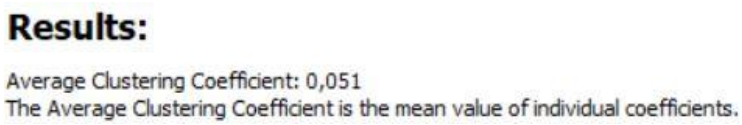

Figura 44. Coeficiente de clustering.

El valor del coeficiente de Clustering representa como los nodos están incrustados entre sus nodos vecinos, de tal manera que su valor medio, en este caso de 0,051, da una visión general del clustering en la red. En este caso particular, este valor indica que de media los nodos de este grafo están pocos integrados en la red global. Esto puede ser debido al gran número de dominios que solamente tienen un 0 u 1 de grado, lo que impide que estén conectados a toda la red en particular.

El valor tan bajo en esta métrica también puede indicar que con los dominios que se han utilizado como entrada para este proyecto no se suelen formar comunidades entre ellos o no sería lo habitual, de manera que haya muchos dominios que se redirijan entre ellos mismos, aunque esto no significa que como antes se ha explicado un dominio pueda direccionar en una o varias ocasiones a otros dominios fake.

5.4 Análisis según eigenvector centrality.

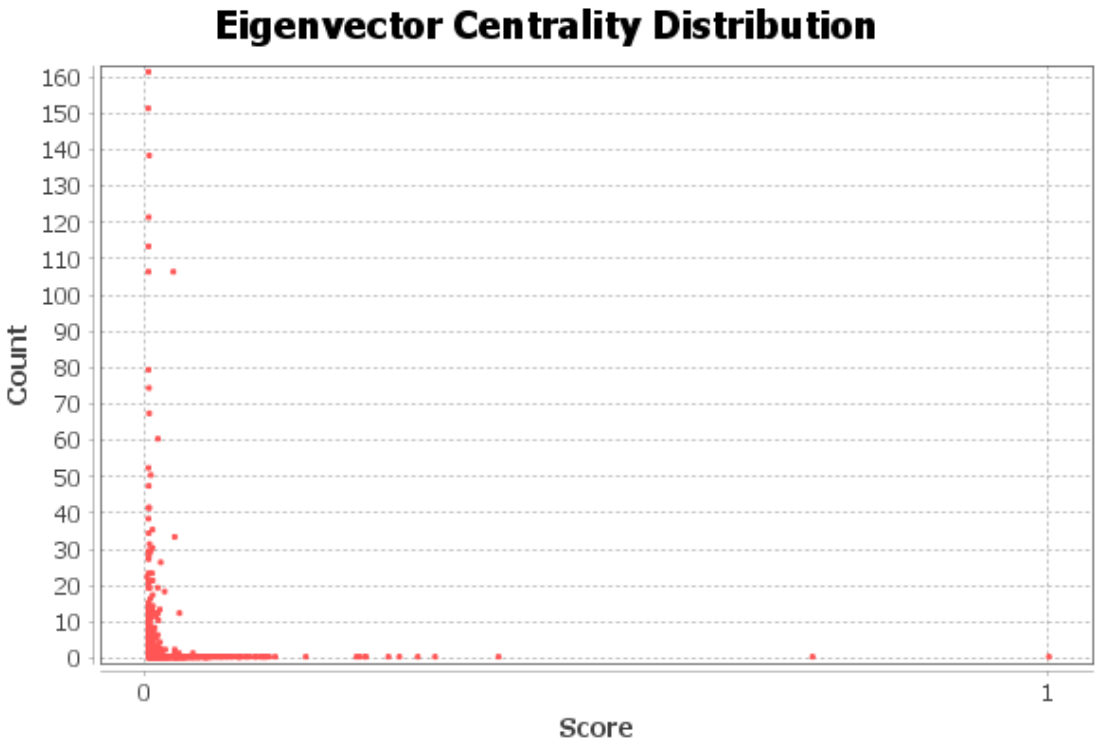

Figura 45. Gráfica eigenvector centrality.

Esta gráfica indica cual es la influencia de los nodos en una red. Se le asigna una puntuación relativa a cada nodo según su importancia en el grafo. Esta importancia se determina según el grado de un nodo y cómo de bien conectados a su vez están los nodos adyacentes (si los nodos adyacentes tienen a su vez influencia o no), es un algoritmo que considera el grafo como no dirigido.

Es aquí uno de los lugares donde se muestra el poder de las redes sociales, viendo como los nodos de Twitter y Facebook tienen una influencia mucho mayor a cualquier otro nodo de la red. Twitter con una puntuación relativa de 1, el máximo, y Facebook con uno 0,8 aproximadamente, son destacadas del resto, aunque los siguientes también pertenezcan a redes sociales.

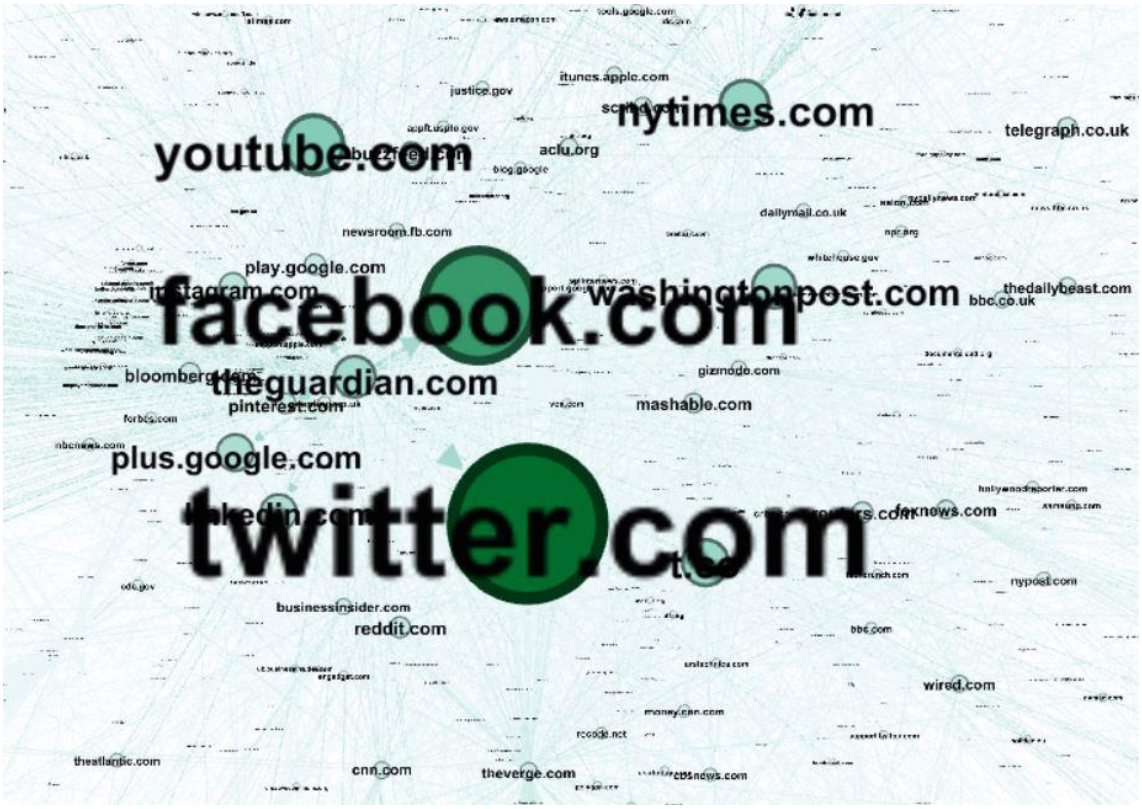

Figura 46. Grafo Eigenvector centrality.

Aparte de ser una métrica que muestra la influencia de las redes sociales y también de muchos periódicos digitales como: New York Times, The Guardian, Telegraph, CNN o el Daily Mail entre otros, también delata a los dominios de las páginas fake, ya que no hay ninguna página web fake destacada en ninguna parte del grafo, porque gran parte de los nodos a los que está conectado un dominio fake (los nodos hoja) no tienen influencia en el grafo y por esto están tan desapercibidos visualmente al haber configurado el tamaño de los nodos del grafo según esta métrica, eigenvector centrality.

6. Conclusiones e investigación futura

6.1 Conclusión

El análisis de dominios fake ha permitido descubrir un poco mejor su funcionamiento a la hora de difundir noticias. Gracias al análisis final se puede deducir que la principal base en la mayoría de estas páginas web fake son las redes sociales, que son la base "común" por la gran mayoría de dominios y a la vez la más recurrente, destacadas Twitter y Facebook del resto, a pesar de que también pueden recurrir a periódicos digitales importantes también un gran número de dominios fake pero con mucha menos frecuencia, como el New York Times o el Washington Post.

En la visualización del grafo también se ha detectado que algunos dominios (los más alejados del centro) suelen apoyarse en dominios que ninguna otra página utiliza y que no suelen nutrirse tanto de las redes sociales o de periódicos digitales, todos ellos en el centro del grafo.

Por último cabe destacar, que también hay dominios fake que redirigen a su vez a otros dominios fake, pero que no ha sido lo más común para el conjunto de dominios analizados en esta investigación.

6.2 Investigación futura

De cara a seguir mejorando el proyecto se podrían optimizar diferentes partes del código en favor del rendimiento y para que la búsqueda de un nodo que forme parte del artículo fuera más efectiva. También mejorar la recogida de dominios fake sobre la que se realiza la investigación para automatizar este proceso lo máximo posible, para que no se manualmente a las diferentes datasets.

También sería interesante poder optar por aislar los dominios fake de todo el resto de dominios en la visualización en Gephi, para así observar qué tipo de conexiones se realizan entre estos, si se puede afirmar objetivamente que se forman comunidades entre estos.

6.3 Valoración personal

En un futuro cercano es muy posible que la manera de difundir las noticias fake ya no solo sea mediante texto sino de una forma más visual y más directa, con técnicas para automatizar gestos faciales y voz de políticos o personalidades famosas, programadas automáticamente para producir noticias a su antojo, manipular la realidad y conseguir un tipo de fake news mucho más verídicas que una simple noticia y causando un impacto más potente.

Esto podría agravar todo lo que engloba a las fake news ya que si en el presente las noticias ya tienen suficiente fuerza como para ser difundidas solo por tener llamativos titulares, a pesar de poseer un contenido falso con una fuente no verificada, en unos pocos años podría empeorar la situación.

Puede que alguien de una corta edad que se ha criado con tecnologías y videojuegos pueda detectar este tipo de noticias como algo poco creíble y pueda tener dudas por las animaciones que puedan ser, pero una gran parte de la sociedad puede estar ajena a ello y ver este tipo de noticias totalmente verdaderas.

Las noticias fake siempre han existido, lo que cambia es el medio y la manera de difundirlas por lo que anticiparse a estos cambios puede ser la clave para disminuir su impacto.

7. Bibliografía

- Definición Fake News por diccionario de Cambridge. Consultado: 03/07/2018

https://dictionary.cambridge.org/es/diccionario/ingles/fake-news

- Término “Fake news” en la herramienta Google Trends. Consultado: 04/07/2018

https://trends.google.es/trends/explore?date=today%205-y&geo=ES&q=Fake%20news

- Social network Analysis por John Scott.

https://books.google.es/books?hl=es&lr=&id=i5EmDgAAQBAJ&oi=fnd&pg=PP1&dq=social+network+analysis&ots=DCQd9SVfg6&sig=6AWdtgCofsMfH281n-rfthIUcwk#v=onepage&q=social%20network%20analysis&f=false

- The rise of social bots.

https://dl.acm.org/citation.cfm?id=2818717

- Spread of hoax in social media

https://pdfs.semanticscholar.org/5e8d/465142f3c1e5d840ba03d1c3f2ae8af87e76.pdf

- Sense-making in social media during extreme events.

https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-5973.12193

- Sentiment analysis of twitter data

https://arxiv.org/abs/1711.10377

- Bs detector: a browser extension that alerts users to unreliable news sources. Consultado: 04/07/2018.

https://github.com/bs-detector/bs-detector

- Dataset for fake news detection research. Consultado: 04/07/2018.

https://github.com/KaiDMML/FakeNewsNet

- Dataset of millions of news articles scraped from a curated list of data sources. Consultado: 04/07/2018.

https://github.com/several27/FakeNewsCorpus

- Documentación de Selenium.

https://www.seleniumhq.org/docs/

- Documentación de Requests.

http://docs.python-requests.org/en/master/

- Documentación de Beautiful Soup.
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Tldextract.
https://github.com/john-kurkowski/tldextract

- CSV en Python documentación.
https://docs.python.org/3/library/csv.html

- Gephi documentación.
https://gephi.org/users/

- Métricas Eigenvector y PageRank. Consultado: 04/07/2018.
https://cambridge-intelligence.com/eigencentrality-pagerank/

- Fake news, cómo se fabrican y sus consecuencias en el mundo.
https://www.telesurtv.net/news/Fake-news-como-se-fabrican-y-sus-consecuencias-en-el-mundo-20170728-0061.html

- Fake news: un fenómeno viejo con nuevas consecuencias.
https://ijnet.org/es/blog/fake-news-un-fen%C3%B3meno-viejo-con-nuevas-consecuencias

- A new study suggests fake news might have won the 2016 electoral selectional.
https://www.washingtonpost.com/news/the-fix/wp/2018/04/03/a-new-study-suggests-fake-news-might-have-won-donald-trump-the-2016-election/?noredirect=on&utm_term=.8aef385cccc6