Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

UNA SOLUCIÓN FÁCIL DE USAR PARA ANALIZAR DATOS ALTAMENTE CONECTADOS APLICANDO ALGORITMOS DE GRAFOS Y TÉCNICAS DE VISUALIZACIÓN

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRA EN SISTEMAS COMPUTACIONALES

Presenta: **ING. IVONNE DEL ROCIO LÓPEZ ROBLES**

Asesor **MAESTRO VÍCTOR HUGO ORTEGA GUZMÁN**

Tlaquepaque, Jalisco. septiembre de 2020.

AGRADECIMIENTOS

Mi más profundo agradecimiento al Maestro Víctor Hugo Ortega Guzmán, quien durante el desarrollo de este proyecto siempre buscó por mi aprendizaje, desarrollo técnico y profesional, esto sin mencionar la paciencia para explicarme y ayudarme a comprender mejor los temas aquí desarrollados.

Agradezco a mi amigo Ing. Héctor Aguilar, que fue de gran ayuda durante todo el desarrollo de mi proyecto, de corazón gracias.

Muchas gracias al Dr. Luis Fernando Gutiérrez Preciado que durante el estudio de la materia Análisis y Diseño de Algoritmos fue un gran motivador para tomar un alto ritmo de estudio, lo cual me impuso a actualizar y mejorar mis habilidades técnicas de desarrollo de software.

Gracias a esta gran Institución ITESO y CONACYT por sus apoyos financieros sin los cuales, hubiera sido imposible lograr mi grado como Maestra en Sistemas Computacionales, número de beca 498404.

DEDICATORIA

Dedico esta tesis a mi familia, Cristina López mi hermana favorita, mi mamá Mireida Amparo Robles García y mi papá †Alberto López León, ustedes son el motor de mi vida me mantienen inspirada y motivada a ser una mejor versión de mi cada día.

RESUMEN

Con el objetivo de favorecer la visualización e interpretación de datos altamente conectados, este trabajo de obtención de grado presenta una solución para trabajar con base de datos de grafos mediante una aplicación Web que consiste en una interfaz donde el usuario puede extraer el esquema de la base de datos de grafos(*BDG*), es decir, visualizar sus nodos, atributos, relaciones y atributos de las relaciones.

Permite hacer ejecución de ocho algoritmos de análisis de grafos, entre ellos los algoritmos de centralidad: *Page Rank, Betweeness Centrality, In/Out Degree* tal como lo son algoritmos de detección de comunidades, tal como lo son: *Louvain, Label Propagation y Connected Components*. La interfaz genera para el usuario una sugerencia para ejecutar cada uno de los algoritmos basada en la base de datos que se está consultando en *NEO4J* y usando el lenguaje de consultas *CYPHER*, una vez ejecutados los algoritmos, se suman atributos significativos al grafo, lo cual permite hacer un análisis más detallado y profundo del mismo.

La aplicación *Web* cuenta con un algoritmo de análisis de sentimientos sobre uno o más atributos de los nodos y así sumar un atributo con el resultado, que puede ser: *negativo, positivo o neutro*.

El usuario tiene la posibilidad de hacer una extracción de datos usando el lenguaje de consultas *CYPHER*, el resultado de dicha consulta es direccionado a otra interfaz de la aplicación, donde esté puede elegir uno y/o siete modelos de visualización para hacer análisis de los datos, entre ellos: *Knowledge Map, Zoomable Sunburst, Hierarchical Bar, Hierarchical Edge Bundle, Hierarchical Edge Tree, Hierarchical Edge Vertical y Table Graph*.

Para cada modelo, el usuario puede libremente seleccionar los atributos que quiere analizar, en su mayoría, los modelos de visualización permiten interacción con el usuario lo cual permite una mejor interpretación de los datos. Todos los modelos están acompañados de una tabla complementaria que contiene los datos del modelo.

En términos técnicos, este proyecto fue desarrollado usando la técnica de *Modelo, Vista y Controlador*. Está dividido en tres capas: base de datos, servicio Web e interfaz gráfica. En el *back-end*, usamos *NEO4J* con una base de datos de grafos pre-existente que el usuario final desea analizar En el *middleware* se tiene un servicio *Web* desarrollado en el lenguaje de programación *PYTHON* que incluye credenciales de base de datos y muchas funciones altamente personalizables. Toda la respuesta del servicio se proporciona en el formato *JSON*.

La aplicación *Web* está desarrollada usando *JQUERY, BOOTSTRAP, JAVASCRIPT*. Además, los modelos de visualización se desarrollaron usando las bibliotecas *JAVASCRIPT D3*.

Gracias a la solución generada en este trabajo es posible facilitar en análisis de redes de información sin tener conocimientos de bases de datos de grafos.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS Y ABREVIATURAS

acrónimo o abreviatura		**Significado**
AI		Inteligencia Artificial
API		*Application Programming Interface*
AWS		*Amazon Web Services*
CQL		*Common Query Language*
CSS		*Cascading Style Sheets*
CSV		*Comma-Separated Values*
DOM		Modelo de Objetos de Documento
DW		*Dataware House*
BDG		Base de datos de grafos
HTML		*Hypertext Markup Language*
IoT		*Internet of Things*
IT		*Information Technology*
JSON		*JAVASCRIPT Object Notation*
LSH		*Locality Sensitive Hashing*
MDL		*Modeling by shortest data description*
MDM		*Master Data Management*
MVC		*Model View Controller*
NOSQL		*Not Only SQL*
REST		*Representational State Transfer*
SQL		*Structured Query Language*
SQL		*Structured Query Language*
SSN		*Set of Sumilar Nodes*
SVG		*Scalable Vector Graphics*
TOG		Trabajo de Obtención de Grado
UGS		*Unified Grap Summatization*
URI		*Uniform Resource Identifiers*
WWW		*World Wide Web*
HTTP		*Hypertext Transfer Protocol*
CNM		*Clauset, Newman, and Moore*
VBA		*Visual Basic para Aplicaciones*
DDE		*Dynamic Data Exchange*
GO		*Gene Ontology*
SIF		*Standard Interchange Format*

1. INTRODUCCIÓN

Las empresas, organizaciones, escuelas y gobiernos tienen una gran necesidad de comprender la cantidad masiva de datos que actualmente circula por todas partes, con el objetivo de tomar mejores decisiones, construir y proporcionar soluciones excelentes para servir a las personas[1].

Como se indica en [2] con millones o incluso miles de nodos y relaciones en una base de datos de grafos es casi imposible comprender la información codificada en estos repositorios mediante una simple inspección visual. Para facilitar la interpretación de los datos, este proyecto proporciona una interfaz de usuario que apoya a comprender y analizar los datos semiestructurados obtenidos de un *BDG* [3] - [4]. La interfaz hace posible la ejecución de potentes algoritmos de análisis de grafos que agregan atributos altamente informativos a una base de datos existente. Además, este desarrollo permite que los datos se muestren en diferentes modelos visuales para un análisis e interpretación preciso y así lograr los objetivos de análisis de información establecidos por los analistas de datos.

1.1. Antecedentes

El trabajo es parte de la investigación realizada por los catedráticos del ITESO relacionada al análisis de grafos *sumarizados*, en el que emplean la base de datos de grafos *NEO4J[5]*. Dicho trabajo utiliza atributos del grafo para generar agrupación y lograr dicha sumarización. Inicialmente este proyecto utilizaba archivos *CSV* como entrada, con la información de los nodos, atributos y relaciones, ahora se conecta directamente a *NEO4J* para tomar la información a analizar.

Este trabajo de obtención de grado pretende apoyar la investigación aportando nuevos atributos a los nodos para utilizarlos como insumo en el proceso de sumarización y análisis de información.

1.2. Problema

Actualmente, visualizar e interpretar de modo amigable los datos extraídos de la *BDG* es una oportunidad. Desarrollar una interfaz que se conecte a *NEO4J* permita, ejecutar algoritmos de centralidad y agrupamiento a los nodos, incorporar los valores nuevamente a la *BDG* y analizar los resultados apoyado con diferentes visualizaciones es algo que se requiere para los analistas de datos.

1.3. Objetivos

1.3.1. Objetivo General:

Crear una interfaz para el análisis de información semiestructurada que obtenga los datos de *NEO4J*, permita aplicar algoritmos de centralidad y detección de comunidades al grafo, e incluir los datos nuevos a la *BDG* para visualizar la información resultante iterando las veces que sea necesario hasta contar con la información que permita responder a las interrogantes planteadas para el análisis.

1.3.2. Objetivos Específicos:

1.3.2.1. Una interfaz gráfica que facilite seleccionar distintos algoritmos de teoría de grafos para el análisis de información.

1.3.2.2. Un servicio web que procese las solicitudes y genere consultas a la base de datos y añade nuevos atributos a la misma con los resultados de la ejecución de los algoritmos del análisis de grafos.

1.3.2.3. Un generador de visualizaciones interactivas que permite analizar la información desde diferentes representaciones gráficas.

1.5 Novedad científica, tecnológica o aportación

Durante el desarrollo de este trabajo se estudiaron diferentes bibliográficas disponibles hasta este momento, a continuación, se destacan las aportaciones principales de esta solución:

- Permite una conexión directa entre la interfaz Web y *NEO4J*.
- Permite la exploración de cualquier tipo de grafo con nodos altamente conectados disponible en *NEO4J* y no solo de redes sociales.
- Agrega nuevos atributos al grafo de *NEO4J* como resultado de la ejecución de hasta ocho algoritmos de base de datos de centralidad y detección de comunidades.
- Los algoritmos de base de datos a ejecutar desde la interfaz son sencillamente incrementables con solo manipular un archivo de configuración.
- Es posible agregar un atributo de análisis de sentimiento para cada atributo de texto existente en el grafo.
- Los modelos de visualización usan la librería de *JAVASCRIPT*, *D3* que es orientada a documentos lo cual genera un aspecto muy amigable y manipulable dentro de la gráfica.
- En términos técnicos, el desarrollo en modo *MVC*(*Model View Controller*, por sus siglas en inglés). hace muy sencillo hacer cambios en el modelo, así como también la implementación de la librería *AIOHTTP* dentro del middleware permite hacer uso de cualquier *API REST* que ese disponible. Ambas características permiten que futuras mejoras en la interfaz sean muy sencillas de implementar.

2. ESTADO DEL ARTE O DE LA TÉCNICA

Ahora que tenemos claro el objetivo de esta solución, revisaremos una recopilación de fuentes importantes, ideas, conceptos donde se hace referencia a uno o varios componentes que este trabajo de obtención de grado propone.

2.1 Modelo MVC usando *PYTHON* para *FLASK Framework*

En [6] se propone un estudio que diseña un *MVC,* observe figura 1, este sistema tiene un generador sencillo y rápido que puede hacer una estructura de carpetas *MVC*, el objetivo es proporcionar una fácil creación de sitios Web utilizando el método *MVC* usando el *framework* de *FLASK* con lenguaje *PYTHON*, con este diseño se pretende también ayudar a los desarrolladores Web a mejorar la velocidad y la calidad del trabajo, y proporcionar un *framework* y una plataforma de trabajo que es eficiente para usuarios novatos o experimentados.

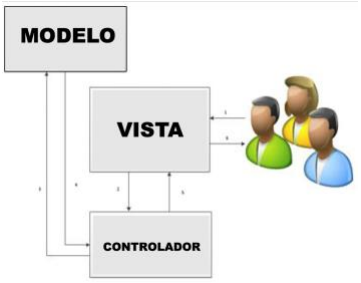


Figura 1- El patrón MVC en este sistema [6]

En este proyecto también hacemos una implementación del modelo *MVC*, ver figura 2, que ha resultado de gran utilidad para el desarrollo controlado y escalable del proyecto, igualmente se hace uso de *CSS, JAVASCRIPT, PYTHON y HTML*, a diferencia del estudio y este trabajo implementa el *framework* AIOHTTP en lugar de *FLASK* ya que la solución que nosotros creamos genera múltiples llamadas a nuestro servicio *Web* por lo tanto, necesitábamos una librería de cliente/servidor *HTTP* asíncrono, compatible con los sockets Web de cliente y servidor listos para usar, por lo tanto AIOHTTP es una elección de una

librería mucho más asertiva y poderosa, que en términos de escalabilidad aporta más valor a este trabajo. La Tabla I muestra las similitudes y diferencias entre [6] y este proyecto.

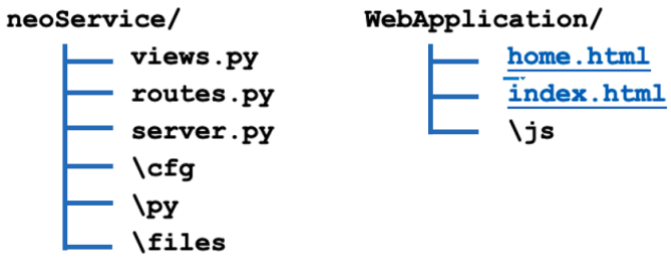


Figura 2 - Estructura de carpetas propuesta en este trabajo de obtención de grado.

	TOG	Publicación
Model View Controller	Si	Si
CSS	Si	Si
JAVASCRIPT	Si	Si
BOOTSTRAP	Si	Si
PYTHON	Si	Si
HTML	Si	Si
FLASK	**No**	**Si**
AIOHTTP	Si	No
Llamadas Asíncronas al servidor *Web*	**Si**	**No**
Conexión con base de datos Grafos	**Si**	**No**
Configurable a base de archivos *FLASK*	**Si**	**No**
Manejo de Servicios Web en dominios cruzados	**Si**	**No**

Tabla I Comparativo entre *TOG* y Publicación [6] Modelo *MVC* usando *PYTHON* para *FLASK Framework.*

2.2 Análisis visual de grandes grafos

En [7] se propone una discusión de varios aspectos para las diferentes etapas del proceso de análisis de grafos visuales, existen tres elementos que forman la base para sistemas efectivos de análisis de grafos visuales y están estrechamente relacionados, esto son: representación visual, interacción del usuario y análisis algorítmico, ver figura 3.

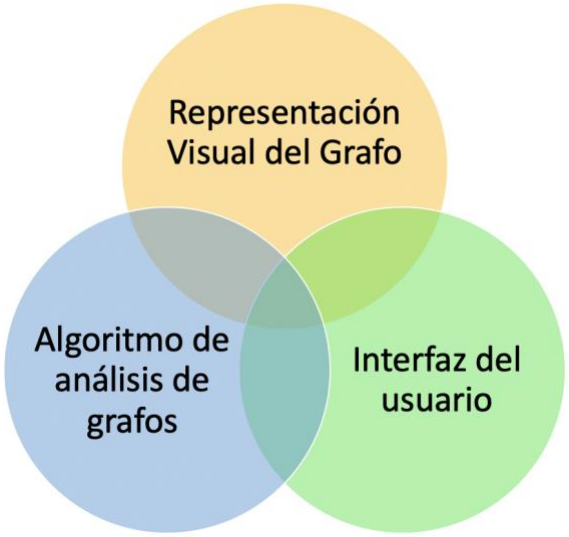


Figura 3 Los principales componentes del análisis gráfico visual

Recientemente, *Visual Analytics* ha sido una importante fuerza para la investigación y el desarrollo de técnicas de visualización interactiva para grandes cantidades de datos, incluidos los grafos, el objetivo de [7] es tomar una perspectiva de *Visual Analytics* en el campo del análisis de grafos visuales al considerar explícitamente de manera unificada los aspectos de la representación visual, la interacción del usuario y el análisis algorítmico. Estos tres elementos forman la base para sistemas efectivos de análisis de grafos que están estrechamente relacionados entre sí.

En la visualización de grafos, el pre-procesamiento algorítmico a menudo incluye su simplificación para reducir su tamaño, mientras se mantiene la estructura principal del mismo. El grafo modificado se usa entonces para una inspección visual más sencilla, ya que los grafos grandes y complejos son difíciles de entender incluso utilizando algoritmos avanzados de posicionamiento de relaciones y nodos, hay dos enfoques principales para la reducción de grafos: filtrado de grafos y agregación de grafos.

Por otro lado, debemos tener en cuenta a la visualización como uno de los principales medios de análisis de grafos exploratorios, incluye el desarrollo de tipos apropiados de representaciones visuales (por ejemplo, diagramas de matriz o enlace de nodo), colocación eficiente de elementos grafos en la pantalla y asignaciones eficientes de atributos visuales (diseño de elementos gráficos para mejorar la legibilidad del dibujo). Por lo tanto, la visualización de grafos exploratorios requiere más de un algoritmo de diseño para revelar las diversas perspectivas sobre las relaciones entre nodos.

La visualización de grafos usando el modelado con árboles son más simples que los grafos generales. Las técnicas para mostrar árboles se pueden dividir en tres grupos principales: relleno de espacio, basado en enlaces de nodo e híbrido como lo muestra la figura 4.

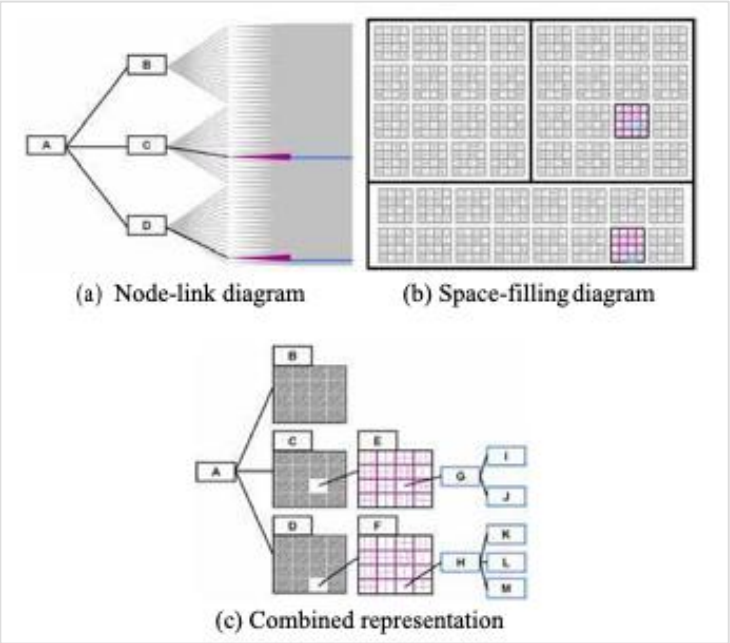


Figura 4 Tres tipos de técnicas de visualización de jerarquía a)Enlaces de nodo, b) Relleno de espacios, c) Híbrido[8].

Las técnicas para mostrar gráficos generales se pueden dividir en tres grupos principales: basados en enlaces de nodo, basados en matriz e híbridos, como ejemplo la figura 5.

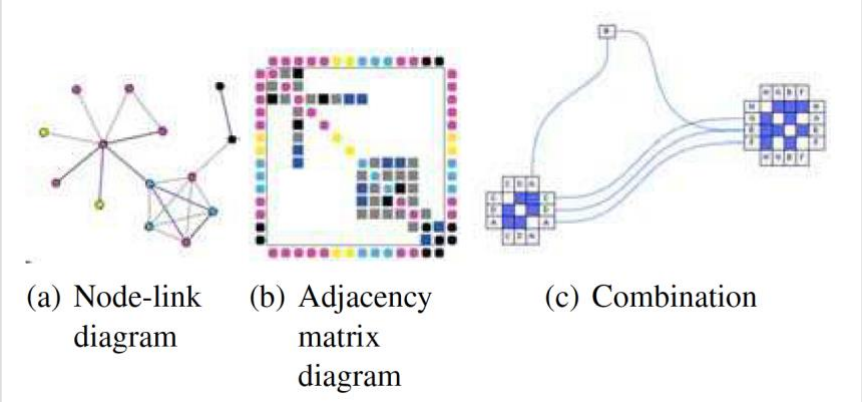


Figura 5 Tres tipos de tecnología de visualización gráfica general a) Diagrama de enlace de nodo, b) matriz de adyacencia, c) combinado [9].

Además de los diseños mencionados, la legibilidad de la pantalla se puede mejorar mediante la agrupación de relaciones, ver figura 6. Para los grafos dirigidos, la representación de las direcciones de las relaciones es importante. Los nodos y relaciones del grafo a menudo tienen atributos asociados que se incluyen en el análisis.

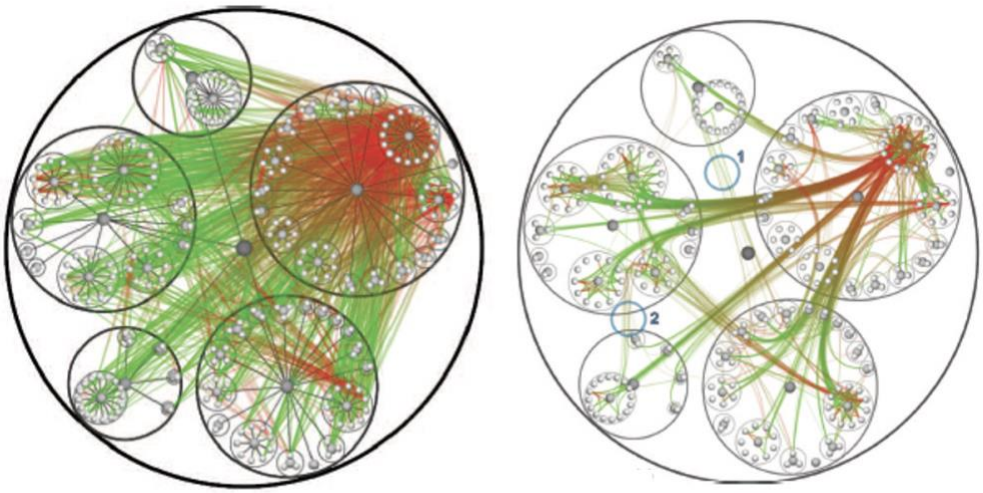


Figura 6 El uso de agrupación de relaciones para mejorar la legibilidad del grafo: a) grafo original b) grafo con agrupación de relaciones [10].

La visualización iterativa, que permite vista panorámica y acercamiento que permiten navegar en cualquier dirección y cambiar el nivel de acercamiento en la vista. Para los diagramas de enlace de nodo y que permite navegar a lo largo de las relaciones de un nodo seleccionado y, por lo tanto, explorar la estructura del grafo.

Básicamente, esta investigación presenta las diferentes estrategias sobre análisis de grafos visuales y se ocupa de los problemas interrelacionados del dibujo del grafo, la presentación gráfica, la interacción humano-computadora y el análisis.

El proyecto aquí presentado, toma como referencia a [7], específicamente con la premisa de la importancia de la visualización de grafos para su análisis, no solo refiere acercamiento a la importancia de la visualización de grafos para su análisis, si no también implementa prácticamente la visualización de grafos y permite que el usuario haga uso del modelo de filtrado del grafo, proponiendo una herramienta de personalización de la gráfica, a la vez que permite al usuario hacer tantas combinaciones como lo desee para el posterior análisis, aunado a esto, permite añadir atributos a grafos implementando algoritmos de grafos para sumar nuevos atributos al grafo, y así contribuye a un mejor análisis durante la visualización. En la tabla II se muestran las similitudes y diferencias entre [7] y este proyecto.

	TOG	Publicación
Visualización de grafos	Si	Si
Filtrado de grafos	Si	Si
Agregación de grafos	Si	Si
Preprocesamiento de grafos	Si	Si
Grafo con enlaces de nodo	Si	Si

Grafo de árbol	Si	Si
Grafo de bordes	Si	Si
Diagrama de enlace de nodo	Si	Si
Visualización iterativa	Si	Si
Implementación de Algoritmos de Grafos	**Si**	**No**

Tabla II Comparativo entre *TOG* y Publicación [7].

2.3 Algoritmos de Comunidades usando *NEO4J*

El descubrimiento de comunidades es fundamental para el análisis de redes sociales, ya que proporciona una forma natural de descomponer un grafo social en otros más pequeños en función de las interacciones entre los individuos. Las comunidades no necesitan ser disjuntas y a menudo exhiben una estructura recursiva, esto último, se ha establecido como una característica distintiva de los grandes grafos sociales, lo que indica una modularidad en la forma en que los humanos construyen sociedades, en búsqueda de un manejador de base de datos de grafos, resulta interesante el estudio de [11] donde los autores explican la comparación de cuatro algoritmos de descubrimiento de comunidades, *Newman-Girvan, Walktrap, Louvain y CNM*. Estos algoritmos se implementaron en el lenguaje de programación Java sobre la base de datos de grafos *NEO4J*.

- **Algoritmo *Newman-Girvan***, se basa en la centralidad intermedia (*edge betweeness algorithm*), una métrica de centralidad de las relaciones que cuenta la fracción del número de los caminos más cortos que conectan dos vértices.
- **Algoritmo *Walktrap***, se basa en el principio del caminante aleatorio. Comenzando desde cualquier vértice aleatorio, el caminante aleatorio eventualmente pasará más tiempo en segmentos de gráficos densamente interconectados, ya que es más probable que una relación seleccionada aleatoriamente conduzca a otro vértice dentro del segmento que a un vértice fuera de él.
- **Algoritmo *Louvain***, un algoritmo de agrupamiento jerárquico que opera en gráficos ponderados, el objetivo es crear comunidades donde la densidad de la relación es alta, mientras que la densidad intercomunitaria sigue siendo baja.
- **Algoritmo *CNM***, también es un algoritmo jerárquico de partición de vértices. Como tal, inicialmente cada vértice constituye una comunidad separada. Luego, las comunidades vecinas se fusionan progresivamente con otras más grandes hasta que ya no sea posible de acuerdo con el criterio estructural.

Después de aplicar los cuatro algoritmos a un mismo grafo complejo y grande, este estudio, demuestra que los algoritmos de *Louvain y CNM* producen menos comunidades que *Newman-Girvan y Walktrap,* y otra observación es que las comunidades tienden a agruparse en tamaño, tal como puede apreciarse en la figura 7.

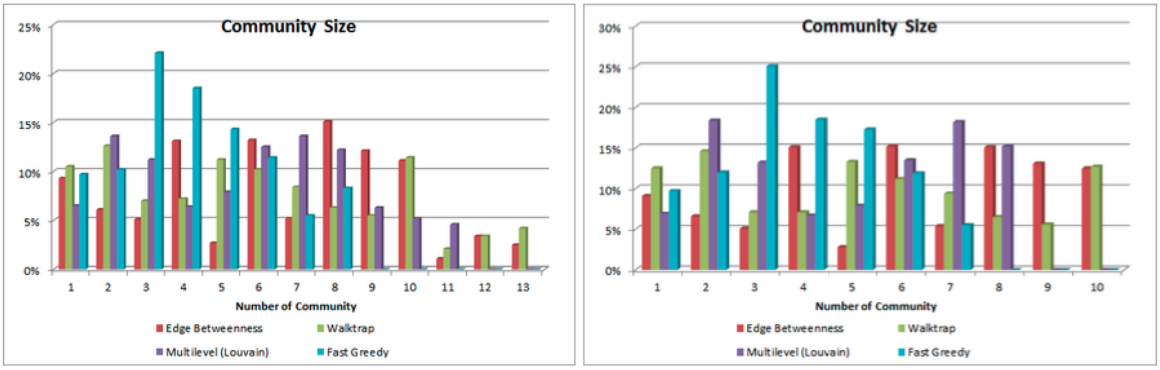


Figura 7 Gráfica de tamaños de comunidad[11]

Como observación general, no existe un valor óptimo único para la cantidad de comunidades, sin embargo, *Newman-Girvan* es consistentemente mejor con *Walktrap y Louvain* siguiendo de cerca y compartiendo la segunda posición. El *CNM* tiene el peor desempeño, lo que puede atribuirse al hecho de que crea menos comunidades, que seguramente serán heterogéneas. Como *Newman-Girvan* suele ser un algoritmo exhaustivo, parece que los algoritmos de *Louvain y Walktrap* son opciones equilibradas.

Es interesante, observar lo complejo que resulta implementar los algoritmos de grafos manualmente, para hacer una posterior comparación y análisis, este proyecto, a diferencia de lo presentado en [11] implementa un total de ocho algoritmos de grafos, de los cuales tres, *Betweenness, Connected Components y Louvain*, permiten la detección de comunidades en la base de datos *NEO4J*, esta poderosa base de datos cuenta con una sencilla Implementación de la librería *NEO4J Graph Algorithms*, que nos permite de manera simplificada ejecutar cualquiera de estos algoritmos, así que gracias a esto esta solución permite escalar la cantidad de algoritmos de grafo a implementar conforme *NEO4J* agregue más algoritmos a su librería, con tan solo modificar un archivo de configuración en la solución y así permite crear un nuevos atributos en el grafo, para su posterior visualización y análisis dentro de la misma solución aquí presentada. En la tabla III se muestran las similitudes y diferencias entre [11] y este proyecto.

	TOG	**Publicación**
Uso de la base de datos *NEO4J*	Si	Si
Algoritmos de Detección de Comunidades	Si	Si
Newman-Girvan	Si	Si
Walktrap	**No**	**Si**
Louvain	Si	Si
CNM	**No**	**Si**
Betweenness	**Si**	**No**
Connected Components	**Si**	**No**
Implementación Manual Exhaustiva	**No**	**Si**
Implementación a través de librería existente	**Si**	**No**

Escalable a más algoritmos	**Si**	**No**

Tabla III Comparativo entre TOG y la publicación [11]

2.4 Herramienta de Visualización de Grafos – Excel

Microsoft Excel es un programa de hoja de cálculo incluido en el conjunto de aplicaciones de *Microsoft Office*. Las hojas de cálculo presentan tablas de valores organizados en filas y columnas que pueden manipularse matemáticamente utilizando operaciones y funciones aritméticas básicas y complejas. Además de sus características básicas de hoja de cálculo, Excel también ofrece soporte de programación a través de *Visual Basic* para Aplicaciones (*VBA*) de *Microsoft*, la capacidad de acceder a datos de fuentes externas a través del Intercambio Dinámico de Datos (*DDE* por sus siglas en inglés) de *Microsoft* y amplias capacidades de gráficos.

En [12] nos explica cómo podemos generar la visualización de grafos usando *Microsoft Excel*, la extracción de nodos puede generarse usando la funcionalidad de tablas dinámicas o comúnmente conocida como tablas pivote, como tiene dos columnas de nodos (por ejemplo, fuente y destino), debe crear dos tablas dinámicas, una para cada columna. Luego puede concatenarlos para crear la lista de nodos, y si existen nodos duplicados, puede eliminarlos.

Una matriz de adyacencia es simplemente una cuadrícula donde los nodos se representan como los títulos de las filas y columnas, y los enlaces se representan como las celdas en una matriz. El número en la celda indica un atributo de enlace como el peso del enlace, esta es una de las visualizaciones de grafo que puede generar *Microsoft Excel* tal como se aprecia en la figura 8.

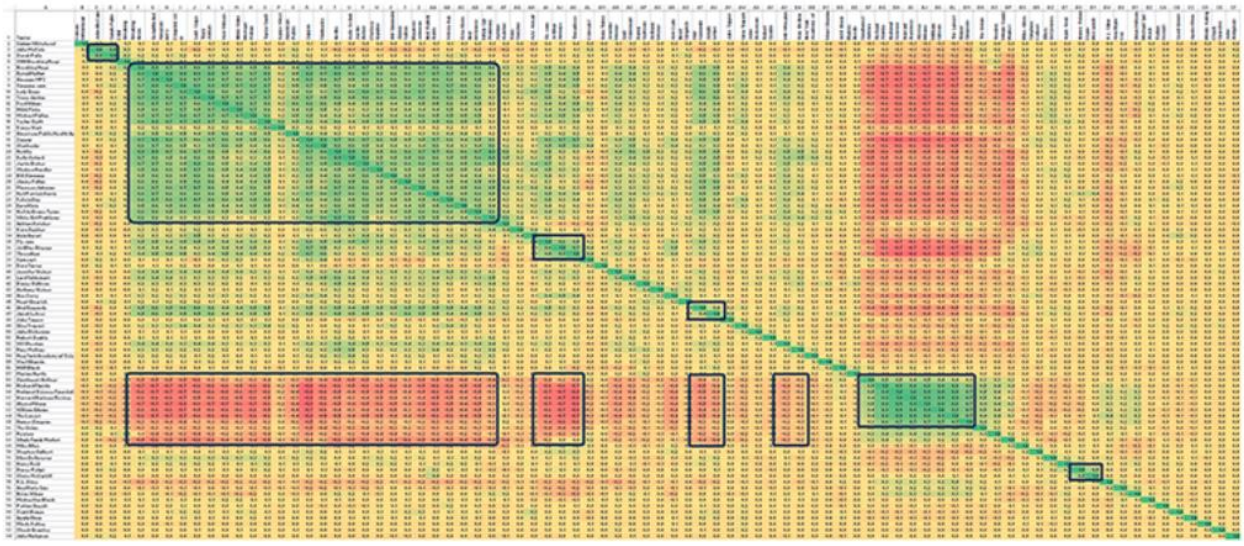


Figura 8 Esta matriz de adyacencia alejada muestra una ordenación sucesiva a través de una macro. Grupos más distintos son visibles[12]

Es importante explicar que para llegar a esta visualización usando *Microsoft Excel*, es necesario conocer muy bien las relaciones entre los datos, hacer una extensiva manipulación manual de las filas y columnas durante la construcción de las tablas pivote, lo cual nos exige también un profundo conocimiento de las funcionalidades de *Microsoft Excel*, este es un interesante punto de referencia al comparar Excel con este proyecto, donde la visualización

del grafo está hecha de forma automática al momento de extraer los datos, además de que la combinación de relaciones y nodos es configurable de modo sencillo, por lo tanto no se exige un conocimiento tan profundo de los datos ya que este se da de forma intuitiva dada la simplicidad de navegación en la interfaz. En la tabla IV se muestran las similitudes y diferencias entre *Excel* y este proyecto.

	TOG	Excel
Fuertes habilidades en *MS Excel*	**No**	**Si**
Extracción de datos desde *NEO4J*	**Si**	**No**
Visualización de relaciones de los datos extraídos	**Si**	**No**
Visualización de nodos de los datos extraídos	**Si**	**No**
Visualización de atributos de nodo de los datos extraídos	**Si**	**No**
Visualización de atributos de relaciones de los datos extraídos	**Si**	**No**
Combinación configurable entre nodos y relaciones para generar visualización	**Si**	**No**
Interfaz interactiva para hacer combinaciones entre nodos y relaciones	**Si**	**No**
Visualización de grafo	Si	Si

Tabla IV Comparativo entre TOG y Microsoft Excel[12]

2.5 Herramienta de Visualización de Grafos – *NodeXL*

NodeXL es una herramienta que se instala como *plug-in* en *Microsoft Excel*, y sirve para recopilar, analizar, visualizar e informar sobre los patrones encontrados en colecciones de conexiones en las redes sociales, siendo este uno de los diferenciadores para *NodeXL* de otro software de grafos de apuntar y hacer clic es su funcionalidad específica para el análisis de redes sociales. También proporciona herramientas que facilitan la adquisición de datos de redes sociales, cargarlos en *Excel*, extraer datos específicos de redes sociales, como los principales hashtags, y seguir enlaces a fuentes Web, algunas de sus características son:

- Personalizar la apariencia del grafo de red
- Acercamiento, escala y panorámica del grafo
- Calculo de métricas de grafos básicos y complejos
- Filtrado dinámicamente nodos y relaciones
- Alterar el diseño del grafo
- Encontrar grupos de vértices relacionados
- Importación y exportación de grafos a una variedad de formatos de archivo
- Obtención de redes sociales utilizando conexiones integradas a *TWITTER*, *FACEBOOK*, *FLICKR*, *YOUTUBE, WIKIS, Blogs*, *INSTAGRAM*, Encuestas de red y correo electrónico
- Automatización de la recopilación y creación de grafos de red.
- Aplica algoritmos de grafos, como *Degree*, *Betwenees Centrality*, *PageRank*.

Ciertamente *NodeXL* analiza y extrae múltiples atributos del grafo que nos permiten manipular y personalizar la visualización desde muchos parámetros, por lo tanto, *NodeXL* contiene ambas propiedades de filtrado y agregación de grafos, a su vez, también exige un profundo conocimiento de *Microsoft Excel* para navegar en todas las funcionalidades de *NodeXL*, entender las formulas, los tabuladores generados.

Cuando analizamos grandes volúmenes de información, puede resultar, abrumador visualizar en una sola gráfica todos los datos, ejemplo figura 9, sin embargo, igualmente *NodeXL* nos permite hacer acercamiento sobre las gráficas para su mejor análisis[12].

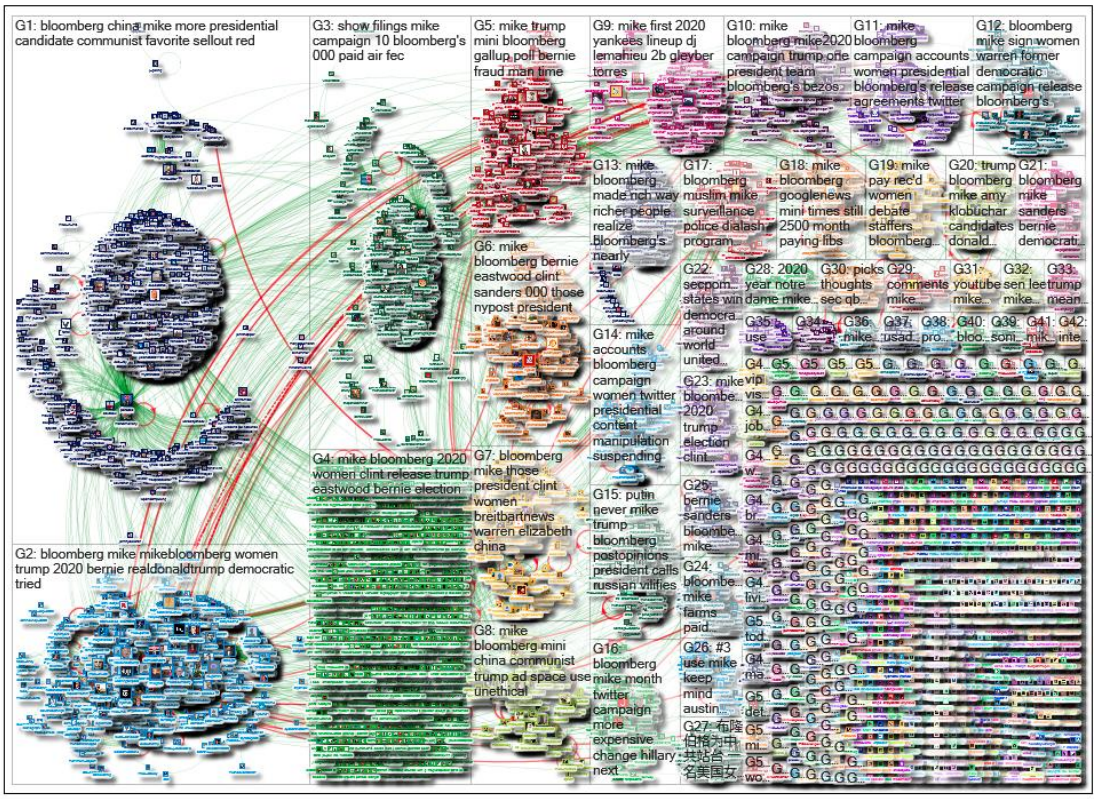


Figura 9 El grafo representa una red de 17,613 usuarios de TWITTER cuyos tweets recientes contenían "Mike 2020", o que fueron respondidos o mencionados en esos tweets, tomados de un conjunto de datos limitado a un máximo de 18,000 tweets[13]

NodeXL puede gestionar diferentes números de relaciones en función de los recursos del sistema informático disponibles, las siguientes son estimaciones aproximadas:

- GB de RAM: redes pequeñas de menos de unos pocos miles de relaciones
- 8 GB de *RAM*: redes medianas de menos de 10-15 mil relaciones
- 16 GB de *RAM*: redes grandes de menos de 80-100 mil relaciones
- 32 GB de *RAM*: redes muy grandes de menos de 200 mil relaciones

A pesar de que *NodeXL* tiene capacidad de contener grafos de hasta 200 mil relaciones, es muy difícil llevar esto a la practica con un equipo de cómputo regular, en su mayoría de las

aplicaciones y casos de uso de *NodeXL* el estándar del tamaño para un análisis es redes pequeñas[12].

Un diferenciador de este proyecto contra *NodeXL*, es que es abierto a cualquier tipo de grafo y no solo de redes sociales, la cantidad de algoritmos de grafos que se pueden aplicar es más extensa que la de *NodeXL*, además es escalable a medida que *NEO4J* enriquece dicha librería, y se agrega a la solución con una sencilla modificación de un archivo de configuración, también se destaca que la interfaz de visualización de atributos del grafo es mucho más intuitiva en la aportación de este proyecto. La tabla V muestra las similitudes y diferencias entre *Excel*, *NodeXL* y este proyecto.

	TOG	***NodeXL***
Fuertes habilidades en *MS Excel*	**No**	**Si**
Dependiente de *MS Excel*	**No**	**Si**
Extracción de datos desde *NEO4J*	**Si**	**No**
Extracción de datos desde *TWITTER, FACEBOOK, FLICKR, YOUTUBE, WIKIS, Blogs, INSTAGRAM*	**No**	**Si**
Algoritmos de Detección de Comunidades	Si	Si
Algoritmos de Detección de Centralidad	Si	Si
Interfaz para personalizar los algoritmos a ejecutar sobre el grafo	**Si**	**No**
Escalable en la cantidad de algoritmos de grafos que puede aplicar al grafo	**Si**	**No**
Limitado en volumen de datos a procesar	**No**	**Si**
Extracción de nodos, relaciones y atributos del grafo	Si	Si
Interfaz de visualizaciones nodos, relaciones y atributos del grafo	**Si**	**No**
Interfaz para personalizar la apariencia del grafo de red	Si	Si
Zoom, escala y panorámica del grafo	Si	Si

Tabla V Comparativo entre TOG y Microsoft Excel-NodeXL

2.6 Herramienta de Visualización de Grafos – *Gephi*

Según explica [14] *Gephi* es una aplicación de código abierto, que se usa como herramienta de visualización de redes para analizar una red en formato de grafo. Muestra diferentes tipos de grafos y tablas para una mejor comprensión y análisis. Proporciona muchas opciones para personalizar el conjunto de datos de trabajo e interactuar con el grafo y personalizar sus

apariencias para identificar patrones. Las siguientes son algunas de las características clave de *Gephi*:

- Visualización en tiempo real
- Algoritmo de diseño
- Marco de estadísticas y métricas
- Cronología de la red
- Personalización de criptografía
- Filtrado dinámico
- Tabla de datos y edición
- Importar y exportar a / desde diferentes fuentes

Gephi es una gran aplicación, y debido a que es compatible con la comunidad *open source*, tiene complementos para trabajar con diferentes fuentes, como bases de datos de grafos; pero está limitado por el hecho de que es una aplicación de escritorio (*Windows, Mac o Linux*). Está diseñado para un solo usuario, y su interfaz de usuario supone tener bases de conocimiento técnico para usarlo correctamente[15], la mayoría de las veces, mientras trabaja con *Gephi* se debe buscar a través de la interfaz para encontrar varias características, a veces enterradas bajo flechas pequeñas o menús desplegables o asociadas con un icono no obvio o una combinación de iconos [12].

Como ejemplo, una vez logrado el manejo adecuado de los atributos y capacidades de *Gephi*, podemos observar las figuras 10 y 11, donde, en la figura 10 podemos ver cómo ser visualiza un grafo original, y en la figura 11, vemos como queda un grafo después de aplicar diferentes características de filtro y agregación de los nodos y relaciones usando la herramienta *Gephi*.

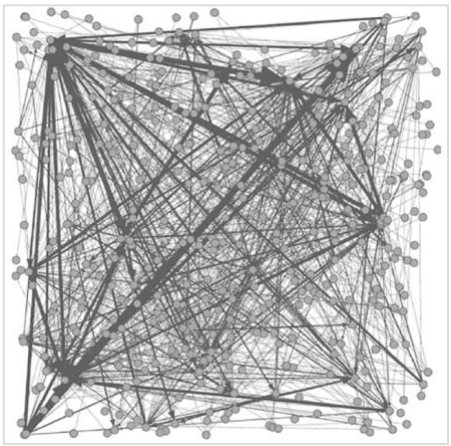


Figura 10 Una visualización gráfica desorganizada de 1000 tweets[15].

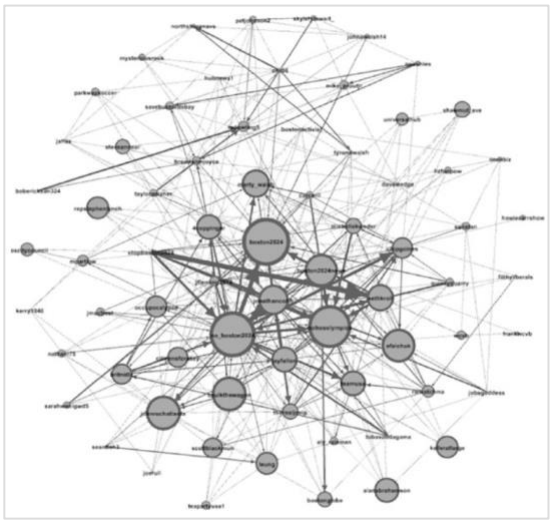


Figura 11 Los nodos se han dimensionado de acuerdo con su puntuación de centralidad de vector propio. Los nodos más grandes tienen la mayor influencia en toda la red [15].

Gephi ha sentado un gran precedente en la realización de este proyecto, la mayoría de las características aquí descritas de *Gephi* están presentes en él, no así las únicas, a diferencia de *Gephi*, este proyecto ofrece una aplicación que puede ser utilizada simultáneamente por varios usuarios gracias a su arquitectura *Web*. Otra notable diferencia es la herramienta de visualización, mientras *Gephi* usa librerías basadas en *JAVA*, este trabajo de obtención de grado explota las funcionalidades de la *Web*, implementando la librería *D3* basado en *JAVASCRIPT* y que da énfasis a todas las capacidades de los navegadores modernos, combinando componentes de visualización potentes y un enfoque basado en Modelo de Objeto de Documento (*DOM* por sus siglas en inglés).

Además de lo anterior, una diferencia más es que, aunque la base de datos *NEO4J* y *Gephi* pueden compartir un formato, la realidad es que es necesario hacer algo de trabajo manual, para logarlo, sin embargo, en este proyecto, tenemos una interconexión entre *NEO4J* y la herramienta de visualización que es transparente para el usuario, con simplemente escribir un *query* en *CYPHER* el usuario ya tendrá los datos disponibles en el visualizador de grafos. La tabla VI muestra las similitudes y diferencias entre *Gephi* y este proyecto.

	TOG	***Gephi***
Interconexión directa con una base de datos de grafos	**Si**	**No**
Visualización en tiempo real	Si	Si
Filtrado dinámico	Si	Si
Tabla de datos y edición	Si	Si
Importar y exportar a / desde diferentes fuentes	**No**	**Si**
Arquitectura *Web*	**Si**	**No**
Visualización basada en *DOM*	**Si**	**No**
Puede escribir la consulta en *CYPHER* y ver los datos directamente en el visualizador	**Si**	**No**

Importación manual de datos	**No**	**Si**
Interfaz para personalizar la apariencia del grafo de red	**Si**	**No**
Zoom, escala y panorámica del grafo	Si	Si
Extracción de nodos, relaciones y atributos del grafo	**Si**	**No**
Interfaz de visualizaciones nodos, relaciones y atributos del grafo	**Si**	**No**
Algoritmos de Detección de Comunidades	Si	Si
Algoritmos de Detección de Centralidad	Si	Si
Interfaz para personalizar los algoritmos a ejecutar sobre el grafo	Si	No
Escalable en la cantidad de algoritmos de grafos que puede aplicar al grafo	Si	No
Limitado en volumen de datos a procesar	No	No

Tabla VI Comparativo entre *TOG* y *Gephi*

2.7 Herramienta de Visualización de Grafos – *Cytoscape*

El proyecto *Cytoscape* ofrece una plataforma central para visualizar y analizar diferentes tipos de redes biológicas. *Cytoscape* se basa en principios y métodos de desarrollo de software que permiten la incorporación incremental y abierta de nuevas herramientas, funcionalidades de software o técnicas analíticas. Esto significa que *Cytoscape* se basa en una arquitectura de *plug-ins*. El sistema central de *Cytoscape* ofrece un conjunto diverso de herramientas para visualización gráfica, manipulación, edición y análisis topológico de diferentes tipos de redes biológicas. Permite la importación y exportación de archivos de datos de red en diferentes formatos, y se puede utilizar como interfaz para diferentes tareas integrales de análisis de datos, incluidas aquellas que requieren datos de expresión génica, análisis de anotaciones basadas en Ontología de Genes (*GO, Gene Ontology* por sus siglas en inglés) y minería de texto. También se encuentran disponibles complementos para la simulación cuantitativa de modelos de ruta bioquímica y agrupación en red.

Una de las características de funcionalidad más poderosas que ofrece *Cytoscape* es la integración interactiva de redes biológicas y datos de expresión genética, que permite al usuario cargar datos, establecer propiedades visuales de nodos y conexiones, y analizar patrones funcionales y estructurales[16].

Los formatos de entrada admitidos son archivos de texto delimitados, *MS Excel, SIF* (formato de interacción simple), *GO* (asociación de genes), etc. Permite la identificación de módulos activos en redes biológicas. *Cytoscape* también permite exportar estructuras de red como imágenes en diferentes formatos [17].

En [12] explica que *Cytoscape* es poderoso con respecto a los atributos visuales. Casi cualquier atributo visual se puede conectar (mapear) a los datos. Por ejemplo, más allá del tamaño y el color, *Cytoscape* ofrece color de borde de nodo, ancho de borde de nodo, transparencia de nodo, fuente de etiqueta, tipo de línea de borde, transparencia de borde,

visibilidad de borde, entre otros. Puede acceder a los atributos visuales en la pestaña *Estilo*. El panel muestra una lista de atributos visuales organizados por pestañas en la parte inferior para cambiar entre atributos de nodo y atributos de borde. Tenga en cuenta que los atributos visuales adicionales pueden estar disponibles en el menú desplegable *Propiedades*. Como ejemplo, en la figura 12, se puede observar el juego de colores para distinguir las relaciones.

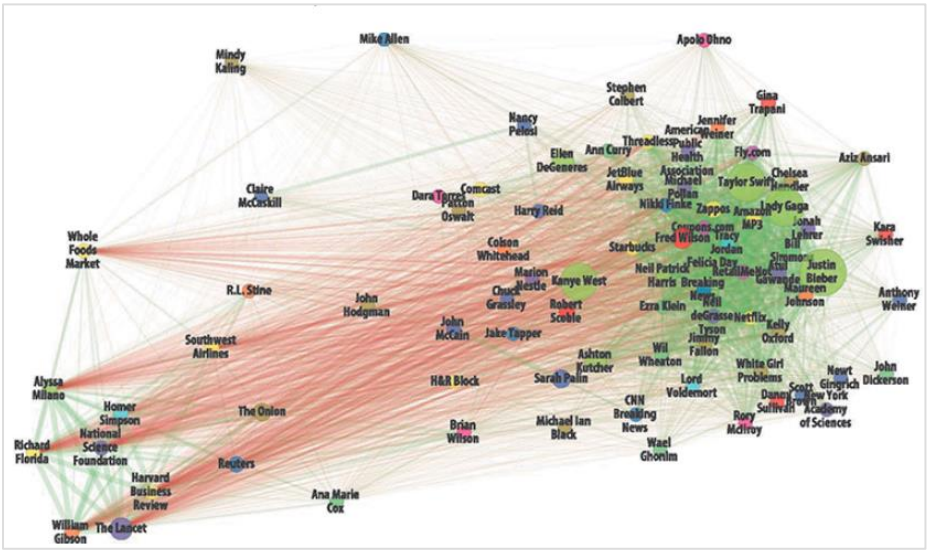


Figura 12 Los temas que tienden de manera similar están unidos por gruesas líneas verdes. Los temas que están inversamente correlacionados tienen gruesas líneas rojas entre ellos. Los temas sin correlación tienen líneas marrones finas transparentes [12]

A diferencia de *Cytoscape*, el proyecto aquí presentado es una aplicación completa y no de *plug-ins*, esta aplicación es escalable gracias al modelo *MVC*, además de ser multiusuario por la arquitectura *Web*, al igual que *Gephi*, *Cytoscape* está basado en librerías *JAVA* lo que hace su visualización menos vistosa y moderna que la implementada en esta investigación usando *D3* y el modelo orientado a documentos. Una ventaja de *Cytoscape* es la versatilidad en la manipulación de colores en los nodos y bordes de los grafos, esto será considerado como un área de mejora para la propuesta aquí presentada. Otra muy importante diferencia es la ejecución de algoritmos de grafos, como los de centralidad y detección de comunidades, *Cytoscape* no los tiene como parte de su producto principal, y es necesario agregar un *plug-in*, probarlo y configurarlo para obtener dicha integración, que se ofrece de modo transparente y configurable en esta investigación. La tabla VII muestra las similitudes y diferencias entre Cytoscape y este proyecto.

	TOG	***Cytoscape***
Interconexión directa con una base de datos de grafos	**Si**	**No**
Visualización en tiempo real	Si	Si
Filtrado dinámico	Si	Si
Exportar en múltiples formatos	**No**	**Si**
Importar y exportar a / desde diferentes fuentes	**No**	**Si**
Arquitectura *Web*	**Si**	**No**

Arquitectura *Plug-ins*	**No**	**Si**
Modelo *MVC*	**Si**	**No**
Visualización basada en *DOM*	**Si**	**No**
Puede escribir la consulta en *CYPHER* y ver los datos directamente en el visualizador	**Si**	**No**
Importación manual de datos	**No**	**Si**
Interfaz detallada para personalizar la apariencia del grafo de red	**Si**	**No**
Manipulación de colores de nodos, relaciones y atributos	**No**	**Si**
Zoom, escala y panorámica del grafo	Si	Si
Extracción de nodos, relaciones y atributos del grafo	**Si**	**No**
Interfaz de visualizaciones nodos, relaciones y atributos del grafo	**Si**	**No**
Algoritmos de Detección de Comunidades	Si	Si
Algoritmos de Detección de Centralidad	**Si**	**No**
Visualización enfocada en grafos genéticos	**No**	**Si**
Interfaz para personalizar los algoritmos a ejecutar sobre el grafo	**Si**	**No**
Escalable en la cantidad de algoritmos de grafos que puede aplicar al grafo	**Si**	**No**
Limitado en volumen de datos a procesar	No	No

Tabla VII Comparativo entre *TOG y Cytoscape*

2.8 Herramienta de Visualización de Grafos – *yWorks*

La herramienta de visualización de grafos *yWorks* es un producto con licencia que tiene fuertes capacidades para crear y editar grafos de forma amigable, tiene una biblioteca muy poderosa, *yFiles* que facilita extraer datos de bases de datos de grafos y crear visualizaciones de grafos con pocos *clics*.

Esta herramienta tiene mucho desarrollo, gran soporte y es muy estable, con *yFiles*, no necesita representar los datos en bruto en la base de datos, con las asignaciones definidas por software, puede mostrar abstracciones más útiles de los datos almacenados en *NEO4J*[18], algunas de las funcionalidades de yWorks:

- Crear visualizaciones de elementos perfectas, específicas de los datos que se mostrarán en la pantalla. El nivel de representación detallada y las pantallas reactivas personalizadas de los datos dan como resultado diagramas y visualizaciones fáciles de entender y seguir.

- Con el diseño correcto, se puede resaltar la estructura de los datos y sus dependencias y relaciones.

- Los algoritmos inteligentes de enrutamiento de relaciones y etiquetado aseguran que no se oculte información detrás de otros elementos. Esto elimina la necesidad de que el usuario desenrede manualmente el diagrama.

- La biblioteca *yFiles* viene con el conjunto más completo de diversos algoritmos de diseño automático para que pueda jugar y elegir.

- Las interacciones y las animaciones incorporadas ayudan al usuario a comprender y navegar por el diagrama más fácilmente y a aumentar la experiencia del usuario.

- Puede agregar interacciones de usuario y la capacidad de editar tanto la estructura como las propiedades del diagrama. Deje que las interacciones desencadenen actualizaciones en la base de datos *NEO4J* o en sistemas externos.

- Incruste las visualizaciones en aplicaciones de usuario final nuevas o existentes como una solución de etiqueta blanca. Cree aplicaciones independientes o integre la funcionalidad en paneles y herramientas más grandes.

Los modelos de visualización principales disponibles son:

- Diseño jerárquico
- Diseño orgánico
- Diseño ortogonal
- Diseño del árbol
- Diseño circular
- Diseño del globo
- Diseño radial
- Diseño paralelo en serie
- Enrutadores

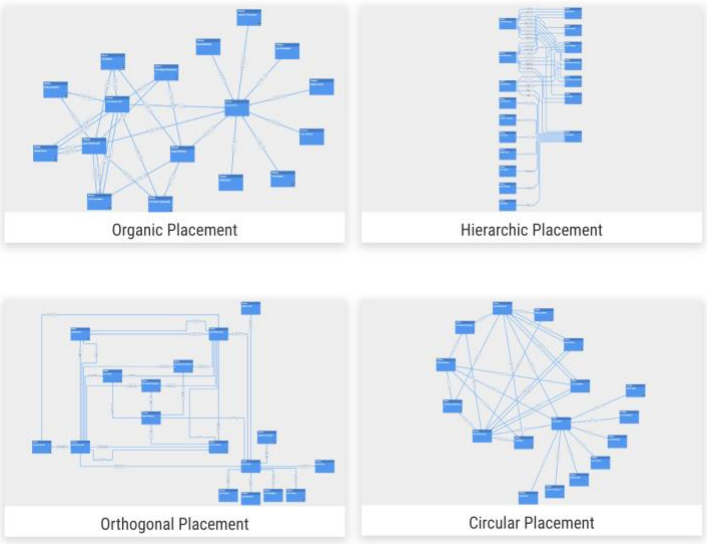


Figura 13 Cuatro Modelos de visualización disponibles en *yWorks*[18]

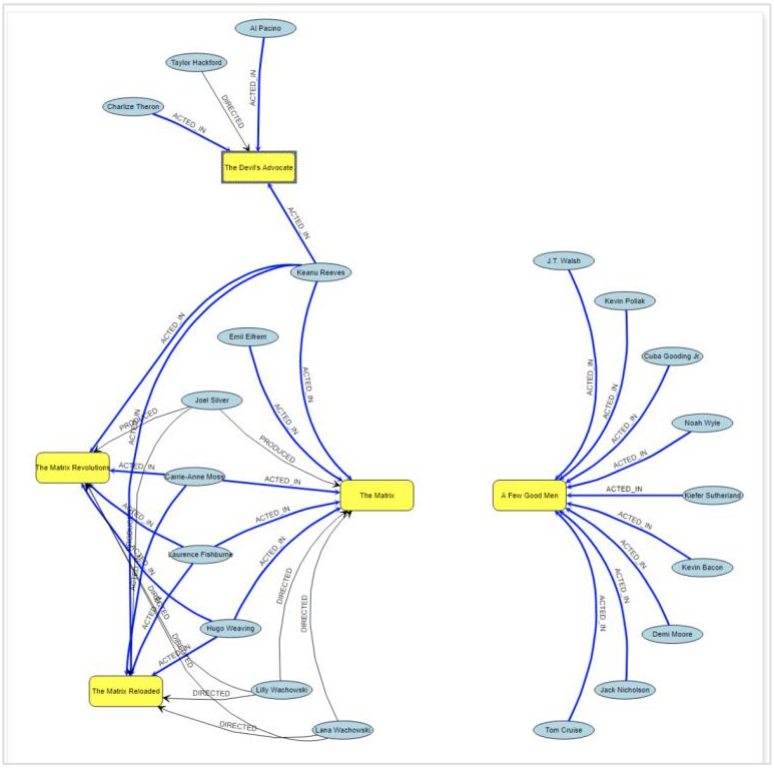


Figura 14 Gráfica automáticamente generada por *yWorks* con datos extraídos de la base de datos de grafos *NEO4J* [19]

yWorks es una gran herramienta de visualización de grafos, muy fácil de usar y con unos cuantos *click* se puedes modelar las gráficas robustas, como ejemplo las figuras 13 y 14, *yWorks* puede administrar también grandes volúmenes de datos sin fallar la aplicación, es innegable notar que *yWorks* y la investigación aquí presentada tienes varias similitudes en apariencia, aun así, parece importante destacar algunos diferenciadores entre *yWorks* y el proyecto aquí presentado, descritas a continuación:

- Es una aplicación con estructura *MVC*, por lo tanto, la conectividad con *NEO4J* permanece inmutable e independiente de la aplicación *Web*, lo que protege el acceso a los datos.

- Además, el middleware, funciona a través de *APIs REST* lo cual lo hace poderoso y con la posibilidad de ser consumidos por otras aplicaciones *Web* que pudieran desarrollarse después, por lo tanto, esta la capacidad hacer esta propuesta ampliamente escalable.

- Este proyecto tiene la posibilidad de ejecutar una amplia gama de algoritmos de grafos a petición del usuario, abre la posibilidad actualmente a ocho de ellos que son sencillamente incrementables, y solo se ejecutan si para el usuario es requerido, además de que estos algoritmos, agregar atributos al grafo de *NEO4J*, no solamente a los datos extraídos.

- Los modelos de visualización actualmente disponibles en este trabajo son menos que los disponibles en yWorks, sin embargo, la aplicación permite la sencilla implementación de más gráficas sin modificaciones profundas en el código.

La tabla VIII muestra las similitudes y diferencias entre *yWorks* y este proyecto.

	TOG	***yWorks***
Interconexión directa con una base de datos de grafos	Si	Si
Visualización en tiempo real	Si	Si
Filtrado dinámico	Si	Si
Exportar en múltiples formatos	**No**	**Si**
Importar y exportar a / desde diferentes fuentes	**No**	**Si**
Arquitectura *Web*	**Si**	**No**
Modelo *MVC*	**Si**	**No**
Visualización basada en *DOM*	**Si**	**No**
Puede escribir la consulta en *CYPHER* y ver los datos directamente en el visualizador	Si	Si
Importación manual de datos	**No**	**Si**
Interfaz detallada para personalizar la apariencia del grafo de red	**Si**	**Si**
Manipulación de colores de nodos, relaciones y atributos	No	No
Zoom, escala y panorámica del grafo	Si	Si
Extracción de nodos, relaciones y atributos del grafo	**Si**	**No**
Algoritmos de Detección de Comunidades	Si	Si
Algoritmos de Detección de Centralidad	**Si**	**No**
Interfaz para personalizar los algoritmos a ejecutar sobre el grafo	**Si**	**No**
Escalable en la cantidad de algoritmos de grafos que puede aplicar al grafo	**Si**	**No**
Limitado en volumen de datos a procesar	No	No

Tabla VIII Comparativo entre TOG e yWorks

Una característica para destacar entre todos los modelos descritos en este estado del arte es que ninguno tiene la posibilidad de hacer análisis de sentimientos en los atributos de grafo, y el *TOG* si lo tiene.

Hemos estudiado modelo MVC, los algoritmos de comunidades de NEO4J, herramientas de visualización de grafos como EXCEL, NODEXL, GEPHI, CYTOSCAPE a YWORKS, un total de ocho distintas soluciones disponibles hoy en día, que comparten y aportan distintitos matices de la solución que en este trabajo se propone. En los siguientes capítulos revisaremos el marco conceptual, así como el desarrollo del marco de análisis y visualización de grafos que impacta a esta solución.

3. MARCO TEÓRICO

Ahora revisaremos los componentes conceptuales y tecnológicos que componen este trabajo, resaltando los antecedentes de bases de datos, NEO4J, algoritmos de grafos y servicios Web y de visualización de datos.

3.1. Tendencia del crecimiento de la información

Históricamente las empresas han almacenado datos de sus clientes para ayudar a mejorar sus operaciones, sin embargo, la cantidad de información que se reunía antes era muy poca, esto debido a que estaba limitada por el medio de recolección, por ejemplo encuestas personales y/o telefónicas, censos, compras, las mismas que requerían invertir mucho tiempo en la aplicación del método, encuestadores y personas dispuestas a contestar dicho métodos, así como su captura y procesamiento para posterior interpretación.

Debido a la expansión de la tecnología y la digitalización de información, se ha incrementado significativamente la cantidad de datos que fluye a través de la economía, en muchos casos, los clientes proveen información mientras que compran, visitan sitios de internet, pagan sus cuentas de modo digital, se conectan con su familia y amigos por medio de redes sociales, usando diferentes aplicaciones móviles, desde rastreadores fitness o televisiones inteligentes, ahora las empresas también reúnen información sobre sus preferencias, experiencias y hasta sobre las características individuales de los consumidores.

Este escenario, que es nuestra actual realidad, nos trae nuevos temas y retos para almacenar, administrar y analizar cantidades masivas de información, ahora es una necesitad considerar el volumen, velocidad y variedad, ver figura 15, de los datos debido a que el crecimiento de la misma se aumenta en una velocidad exponencial, así como la imperativa necesidad de tener herramientas y metodologías muy eficientes para el análisis de esta información y poder realmente aprovechar que la tenemos ahora disponible y así lograr un proceso de mejora continua en el servicio al cliente y las operación de las empresas.

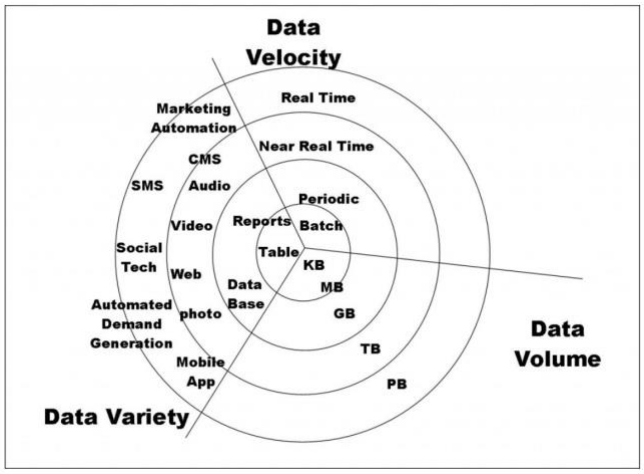


Figura 15 Las 3 primeras "V" de *big data* [20].

El buen análisis y manejo de los datos con lo que ahora contamos puede producir grandes beneficios para la sociedad, como mejoras en la medicina, educación, salud y transporte; así como también generar mejoras en la economía permitiendo a las empresas ofrecer mejores productos, así como bienes y servicios personalizados con las preferencias de los consumidores, todo esto solo se puede alcanzar con el correcto análisis de la información [1].

Según lo estudiado en [21], el volumen de datos operados por las aplicaciones modernas está creciendo a un ritmo muy acelerado, los repositorios de datos para tales aplicaciones superan los *exabytes* y están aumentando rápidamente de tamaño, lo que trae consigo desafíos interesantes para las plataformas informáticas distribuidas y paralelas. Estos retos van desde la construcción de sistemas de almacenamiento que pueden acomodar estos grandes conjuntos de datos hasta la recopilación de datos de fuentes ampliamente distribuidas geográficamente en sistemas de almacenamiento y la ejecución de un conjunto diverso de cálculos sobre datos. Así como también las tareas de análisis a menudo tienen plazos estrictos, y la calidad de los datos es una preocupación importante en otras aplicaciones.

Los datos pueden ser estructurados, por ejemplo: registros médicos electrónicos, financieros, estadísticas gubernamentales, semiestructurados, por ejemplo: texto, tweets, correos electrónicos, no estructurados, por ejemplo: audio y video y en tiempo real, por ejemplo: trazas de red, genéricos registros de monitoreo. Todas estas aplicaciones comparten el potencial de proporcionar ideas invaluables, si se organizan y analizan adecuadamente.

Las aplicaciones que requieren análisis efectivos de grandes conjuntos de datos son ampliamente reconocidas hoy en día, estas incluyen análisis de atención médica, optimización de procesos comerciales y recomendaciones basadas en redes sociales.

Los beneficios socioeconómicos subyacentes del análisis de *big data* y la diversidad de las características de las aplicaciones plantean desafíos. En su publicación [21] destaca la escala y el alcance de los problemas de análisis de datos, las plataformas de hardware de uso común para ejecutar aplicaciones de análisis es de suma importancia poner en perspectiva los recursos necesarios para el procesamiento, como la memoria, almacenamiento,

procesamiento, los recursos de red, así como todas las consideraciones de consumo de energía, que nos permitirán ejecutar dicho análisis.

En [21] se resaltan de manera muy clara los alcances de la aplicabilidad del análisis de *big data*, y sus áreas de impacto, tales como:

- En empresas y comercios el papel de la analítica de *big data* es bien reconocido para mejorar la eficiencia, principalmente en guiar los procesos de decisión.

- En salud y bienestar humano, ofrecen un enorme potencial para intervenciones de pronóstico, terapias novedosas que contribuyen a moldear el estilo de vida y el comportamiento. El análisis de *big data* también es clave para la rentabilidad y la sostenibilidad de la infraestructura sanitaria.

En entornos interactivos u orientados al cliente, el análisis de *big data* guía la interfaz del sistema con los clientes. Es bien conocido que las empresas que preparan su entorno operativo logran optimizar la experiencia del cliente (y los resultados comerciales). Ahora están surgiendo nuevos entornos, como las casas inteligentes, que optimizan simultáneamente los espacios habitables, así como la huella energética y el costo, refiriéndonos específicamente al internet de las cosas (*IoT* por sus siglas en inglés).

3.1.1. Información Estructurada

Los datos estructurados son aquellos que pueden ser formateados para facilitar el almacenamiento, uso y generación de información. Se aplica un formato con base en el tipo de procesamientos que se pretenda ejecutar en los datos. Algunos datos podrían no estar listos, es decir son *no estructurados* para algunos tipos de procesamiento, pero sí podrían estar listos, por ende, *estructurado*s para otros. Por ejemplo, el valor de datos 37,890 podría referirse a un código postal, un valor de ventas o un código de producto. Si este valor representa un código e producto se guarda como texto, no se pueden hacer cálculos matemáticos con ese número, por otra parte, si este valor representa una transacción de ventas, es necesario darle formato numérico[3].

Según [22] podrimos concluir que la información estructurada es aquella que se encuentra almacenada, catalogada e indexada, lo cual permite una fácil identificación y acceso a la misma y, por lo general, está registrada en una base de datos o catálogo estructurado de datos.

Podemos citar una algunas características importantes de las diversas fuentes de información estructurada como

- Los datos se almacenan con una estructura bien definida y que aplica unas normas muy estrictas
- La información se almacena en tablas y se definen relaciones entre dichas tablas
- Las tablas se componen de filas (*tuplas*) y columnas (campos o atributos)

- Por lo general, los almacenes de datos (*Data Warehouses*) usan esta tecnología como almacenamiento subyacente
- Suelen contener metadatos: información sobre los propios datos que ayuda en su interpretación (ej. descripciones, unidades de medida usadas.

3.1.2. Información Semiestructurada

Los datos semiestructurados se refieren a datos que tienen tanto los elementos de un esquema organizacional como aspectos que son arbitrarios. Un diario de teléfono personal con columnas para el nombre, la dirección, el número de teléfono y las notas podría considerarse un conjunto de datos semiestructurados. El usuario puede no estar al tanto de las direcciones de todas las personas y, por lo tanto, algunas de las entradas pueden tener solo un número de teléfono y viceversa.

Del mismo modo, la columna para notas puede contener información descriptiva adicional, es un campo arbitrario que permite al usuario agregar información complementaria. Por lo tanto, las columnas de nombre, dirección y número de teléfono pueden considerarse estructuradas en el sentido de que pueden presentarse en un formato tabular, mientras que la sección de notas no está estructurada en el sentido de que puede contener un conjunto arbitrario de información descriptiva que no puede ser representado en las otras columnas en el diario.

En informática, los datos semiestructurados suelen estar representados por formatos, como JSON, que pueden encapsular asociaciones tanto estructuradas como sin esquemas o arbitrarias, generalmente utilizando pares clave-valor. Un ejemplo más común podrían ser los mensajes de correo electrónico, que tienen una parte estructurada, como el nombre del remitente, la hora en que se recibió el mensaje, etc., que es común a todos los mensajes de correo electrónico y una parte no estructurada representada por el cuerpo o contenido del correo electrónico[23]. Refiera a la figura 16 para revisar ejemplos de tipos de datos estructurados, semiestructurados y datos no estructurados.

Tabla 2. Tipos de datos en el paradigma *big data*

Datos estructurados	Datos semiestructurados	Datos no estructurados
Fichas de clientes Fecha de nacimiento Nombre Dirección Transacciones en un mes Puntos de compra	**Correos electrónicos** Parte estructurada: destinatario, receptores, tema Parte no estructurada: cuerpo del mensaje	**Persona a persona** Comunicaciones en las redes sociales **Persona a máquina** Dispositivos médicos Comercio electrónico Ordenadores, móviles **Máquina a máquina** Sensores, dispositivos GPS Cámaras de seguridad

Figura 16 Ejemplos de tipos de datos, estructurados, semi-estructurados y no estructurados [20].

3.2. Diferentes fuentes de información masiva

Según [23] la tecnología actual nos permite recopilar datos a una velocidad asombrosa, tanto en términos de volumen como de variedad. Hay varias fuentes que generan datos, pero en el contexto del *big data*, las fuentes principales son las siguientes:

- **Redes sociales.** Podría decirse que la fuente principal de todos los *big data* que conocemos hoy en día son las redes sociales que han proliferado en los últimos 5-10 años. En general, se trata de datos no estructurados que están representados por millones de publicaciones en redes sociales y otros datos que se generan en segundos a través de las interacciones del usuario en la Web en todo el mundo. El aumento en el acceso a Internet en todo el mundo ha sido un acto de auto-cumplimiento para el crecimiento de los datos en las redes sociales.

- **Medios de comunicación.** En gran parte como resultado del crecimiento de las redes sociales, los medios representan los millones, sino miles de millones, de cargas de audio y visuales que tienen lugar diariamente. Los videos subidos en *YOUTUBE*, las grabaciones de música en *SOUNDCLOUD* y las imágenes publicadas en *INSTAGRAM* son ejemplos principales de medios de comunicación, cuyo volumen continúa creciendo de manera desenfrenada.

- ***Data warehouses***. Las empresas han invertido durante mucho tiempo en instalaciones especializadas de almacenamiento de datos conocidas comúnmente como *data-warehouses*. Un *DW* es esencialmente colecciones de datos históricos que las empresas desean mantener y catalogar para su fácil recuperación, ya sea para uso interno o con fines reglamentarios. A medida que las industrias cambian gradualmente hacia la práctica de almacenar datos en plataformas como *HADOOP* y *NOSQL*, cada vez más empresas están trasladando datos de sus almacenes de datos preexistentes a algunas de las tecnologías más nuevas. Los correos electrónicos de la compañía, los registros contables, las bases de datos y los documentos internos son algunos ejemplos de datos en *data-warehouse* que ahora se descargan en *HADOOP* o plataformas similares a *HADOOP* que aprovechan múltiples nodos para proporcionar una plataforma altamente disponible y tolerante a fallas.

- **Sensores.** Un fenómeno más reciente en el espacio de los grandes datos ha sido la recopilación de datos de los dispositivos sensores. Aunque los sensores siempre han existido y las industrias como el petróleo y el gas han usado sensores de perforación para mediciones en plataformas petrolíferas durante muchas décadas, la llegada de dispositivos portátiles, también conocidos como *Internet Of Things* como *FITBIT* y *APPLE WATCH*, significaba que cada uno el individuo podría transmitir datos a la misma velocidad a la que solían hacer unas pocas plataformas petroleras hace 10 años.

Los dispositivos portátiles pueden recolectar cientos de mediciones de un individuo en cualquier momento dado. Aunque aún no es un gran problema de datos como tal, ya que la industria sigue evolucionando, es probable que los datos relacionados con sensores se

asemejen más al tipo de datos espontáneos que se generan en la Web a través de las actividades de las redes sociales.

3.3. Grafos

Los analistas de redes sociales utilizan grafos y matrices para representar y analizar información sobre los patrones de relación entre los actores sociales (usuarios, amigos y "amigos de amigos"). En el análisis de red, los datos generalmente se modelan como un grafo o conjunto de grafos. Un grafo es una estructura de datos que tiene un conjunto finito de nodos, llamados vértices, junto con un conjunto finito de líneas, llamadas relaciones, que unen algunos o todos estos nodos. Antes de definir algunas métricas usando el recuento de triángulos, necesitamos definir una tríada y un triángulo. Deje T = (a, b, c) ser un conjunto de tres nodos distintos en un grafo (identificado por G). T es una tríada si dos de esos nodos están conectados({(a,b), (a,c)}) y es un triángulo si los tres nodos están conectados ({(a,b), (a,c), (b,c)})[24].

3.3.1. Aplicaciones de grafos

Para ilustrar la diversidad de aplicaciones que involucran el procesamiento de grafos, comenzamos nuestra exploración de algoritmos en esta área fértil mediante la introducción de varios ejemplos

- **Mapas.** Una persona que está planeando un viaje puede necesitar responder preguntas tales como "¿Cuál es la ruta más corta desde *Providence* a *Princeton*?". Un viajero experimentado que ha experimentado retrasos en el tráfico en la ruta más corta puede formular la pregunta "¿Cuál es la forma más rápida de obtener ¿De *Providence* a *Princeton*? "Para responder a estas preguntas, procesamos información sobre conexiones (carreteras) entre artículos (intersecciones).

- **Contenido *Web*.** Cuando navegamos por la *Web*, encontramos páginas que contienen referencias (enlaces) a otras páginas y nos movemos de una página a otra haciendo clic en los enlaces. Toda la *Web* es un grafo, donde los elementos son páginas y las conexiones son enlaces. Los algoritmos de procesamiento de grafo son componentes esenciales de los motores de búsqueda que nos ayudan a ubicar la información en la Web.

- **Horarios.** Un proceso de fabricación requiere la realización de una variedad de trabajos, bajo un conjunto de restricciones que especifican que ciertos trabajos no se pueden iniciar hasta que se hayan completado ciertos trabajos. ¿Cómo programamos los trabajos de modo que ambos respetemos las restricciones dadas y completemos todo el proceso en el menor tiempo posible?

- **Comercio.** Los minoristas y las instituciones financieras rastrean las órdenes de compra / venta en un mercado. Una conexión en esta situación representa la

transferencia de efectivo y bienes entre una institución y un cliente. El conocimiento de la naturaleza de la estructura de conexión en este caso puede mejorar nuestra comprensión de la naturaleza del mercado.

- **Red de computadoras.** Una red informática consta de sitios interconectados que envían, reenvían y reciben mensajes de varios tipos. Estamos interesados en conocer la naturaleza de la estructura de interconexión porque queremos establecer cables y construir *switches* que puedan manejar el tráfico de manera eficiente.

- **Redes sociales.** Cuando utilizas una red social, construyes conexiones explícitas con tus amigos. Los artículos corresponden a personas; las conexiones son a amigos o seguidores. Comprender las propiedades de estas redes es una aplicación moderna de procesamiento de grafos de gran interés, no solo para las empresas que las respaldan, sino también para la política, la diplomacia, el entretenimiento, la educación, el *marketing* y muchos otros dominios.

En aplicaciones prácticas, es común que el volumen de datos involucrados sea realmente enorme, de modo que los algoritmos eficientes hacen la diferencia entre si una solución es o no factible [23]. Ver figura 17 para encontrar ejemplos de aplicaciones típicas de grafos.

aplicaciones	*item*	*conexión*
mapas	intersección	camino
contenido web	página	enlace
circuitos	dispositivo	cables
comercio	cliente	transacción
relaciones	estudiante	aplicación
redes computacionales	lugar	conexión
programas	métodos	llamada
redes sociales	personas	amistad

Figura 17 Aplicaciones típicas de grafos[25].

3.3.2. Tipos de grafos

Estudiaremos dos modelos de grafos: el primero de ellos grafos no dirigidos o bien con conexiones simples y a continuación dígrafos, donde la dirección de cada conexión es significativa

3.3.2.1. Grafos no dirigidos

Un grafo es un conjunto de vértices y una colección de relaciones que conectan cada uno un par de vértices.

Según [25] los nombres de los vértices no son importantes para la definición, pero necesitamos una forma de referirnos a los vértices. Por convención, usamos los nombres 0 a V-1 para los vértices en un grafo V-vértice. La razón principal por la que elegimos este sistema es facilitar la escritura de código que acceda de manera eficiente a la información correspondiente a cada vértice, utilizando la indexación de matrices. No es difícil de usar una tabla de símbolos para establecer una cartografía 1-1 asociar V nombres de vértice arbitrarios con los números enteros V entre 0 y V-1, por lo que la conveniencia de utilizar índices como nombres de vértice viene sin pérdida de generalidad (y sin mucha pérdida de eficiencia). Usamos la notación v-w para referirnos a una relación que conecta v y w; la notación w-v es una forma alternativa de referirse a la misma relación, ver figura 18 con ejemplo a dos diferentes maneras de representar el mismo grafo.

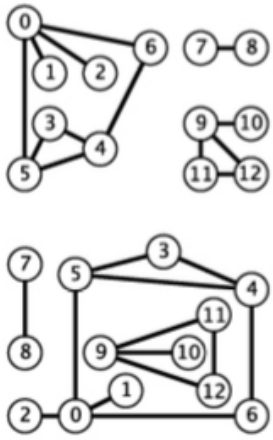


Figura 18 Dos dibujos del mismo grafo[25].

Dibujamos un grafo con círculos para los vértices y líneas que los conectan a sus relaciones. Un dibujo nos da intuición sobre la estructura del grafo; pero esta intuición puede ser engañosa, porque el grafo se define independientemente del dibujo. Por ejemplo, los dos dibujos de la izquierda representan el mismo grafo, debido a que el grafo es nada más que su conjunto (no ordenado) de vértices y su colección (no ordenada) de las relaciones
(los pares de puntos).

3.3.2.2. Grafos dirigidos

En los grafos dirigidos, las relaciones son unidireccionales: el par de vértices que define cada relación es un par ordenado que especifica una adyacencia unidireccional. Muchas aplicaciones (por ejemplo, grafos que representan la *Web*, limitaciones de programación o llamadas telefónicas) se expresan naturalmente en términos de grafos dirigidos. La restricción de un solo sentido es natural, fácil de aplicar en nuestras implementaciones, y parece inocuo; pero implica una estructura combinatoria adicional que tiene profundas implicaciones para nuestros algoritmos y hace que trabajar con grafos dirigidos sea bastante diferente de trabajar con grafos no dirigidos[25].

aplicaciones	*Vértice*	*borde*
red alimentaria	especias	depredador-presa
contenido web	página	Hipervínculo
programas	módulo	referencia externa
celulares	teléfono	llamada
becas escolares	papel	citación
financieras	acciones	transacción
internet	máquinas	conexión

Figura 19 Aplicaciones típicas de un dígrafo [25].

Decimos que una relación dirigido apunta desde el primer vértice en el par y apunta al segundo vértice en el par. La diferencia de un vértice en un dígrafo es el número de relaciones que apuntan desde él, ver figura 19 con algunas referencias a aplicaciones típicas de un digrafo; el grado de un vértice es el número de relaciones que lo apuntan. Soltamos el modificador dirigido al referirnos a las relaciones en los dígrafos cuando la distinción es obvia en el contexto, ver figura 20, donde podemos observar los diferentes componentes de un grafo.

El primer vértice en una relación dirigida se llama su cola; el segundo vértice se llama su cabeza. Dibujamos las relaciones dirigidas como flechas que apuntan de la cola a la cabeza. Usamos la notación v-> w para referirnos a una relación que apunta de v a w en un dígrafo. Al igual que con los grafos no dirigidos, nuestro código maneja relaciones paralelas y bucles automáticos, pero no están presentes en los ejemplos y generalmente los ignoramos en el texto. Ignorando las anomalías, hay cuatro formas diferentes en que dos vértices pueden estar relacionados en un dígrafo: sin relaciones, una relación v-> w de v a w; una relación w-> v de w a v; o dos relaciones v-> w y w-> v, que indican conexiones en ambas direcciones [25].

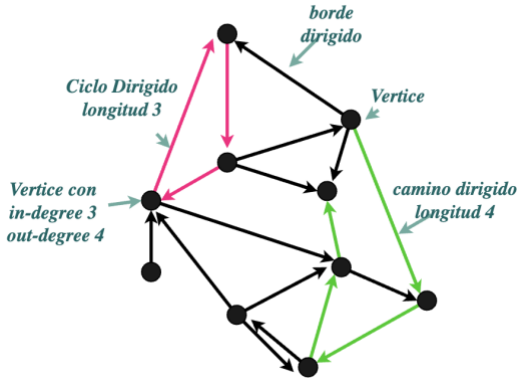


Figura 20 Anatomía de un dígrafo [25].

3.4. Bases de Datos

Actualmente hay muchos debates y discusiones con respecto al manejo y tratamiento de la información digital, y a su almacenamiento, el cual los arquitectos, diseñadores y desarrolladores necesitan acceder para su recuperación y análisis, en este momento, las opciones disponibles para lograrlo son los tradicionales sistemas de gestión de bases de datos relacionales y por otro los sistemas de bases de datos no relacionales y distribuidos conocidos como *NOSQL* (*Not Only SQL*)[26].

3.4.1. Bases de datos relacionales

Una buena definición es la de [27] que dice que una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Estas bases de datos son presentadas a los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo. Los sistemas de bases de datos relacionales pueden presentar un inconveniente asociado al proceso de normalización que los caracteriza.

Sin embargo, hay problemas muy comunes dentro de las bases de datos jerárquicas, el principal es el hecho de que los sistemas gestores bases de datos no implementan ningún control sobre los datos, sino que queda en manos de los desarrolladores garantizar que se cumplan las condiciones invariantes que se requieran para su buen funcionamiento. Es una realidad que en el ejercicio todas las aplicaciones están sujetas a errores y fallos por lo que es casi imposible que no se produzcan inconsistencias a la hora de almacenar los datos en la base de datos [27], a su vez esta publicacion describe los problemas típicos que se producen dentro de este tipo de bases de datos son:

- **Duplicidad de registros.** No se garantiza la inexistencia de registros duplicados. Es decir, no se garantiza que dos registros cualesquiera tengan diferentes valores en un subconjunto concreto de campos.

- **Integridad referencial.** No existe garantía de que un registro hijo esté relacionado con un registro padre valido.

- **Desnormalización**. No tienen controles que impidan la desnormalización de una base de datos. No existe el concepto de campos clave o campos únicos.

3.4.2. Bases de datos no relacionales

Los tiempos han cambiado y la información ha evolucionado a tal punto que los sistemas de información se tienen que encontrar con la manipulación de grandes cantidades de datos, lo cual ha resultado difícil y se han implementado otras estrategias para manejar la información.

Una de esas estrategias es un concepto alternativo para la gestión de bases de datos conocido como *NOSQL*, el cual no solo utilizan el lenguaje *SQL*, este es un sistema de gestión enfocado a solucionar los problemas de escalabilidad y rendimiento de las bases de datos relacionales, en contextos donde el volumen de datos es alto y los sistemas sean bastante concurridos por los usuarios. Dependiendo de la forma en qué se almacena la información en un sistema de gestión *NOSQL*[26].

NOSQL (*Not Only SQL*) realmente es una categoría muy amplia para un grupo de soluciones de persistencia que no siguen el modelo de datos relacional, y que no utilizan *SQL* como lenguaje de consulta; pero, en resumen, las bases de datos *NOSQL* pueden clasificarse en función de su modelo de datos en las siguientes cuatro categorías:

- Orientadas a clave-valor (*Key-Value stores*)
- Orientadas a columnas (*Wide Column stores)*
- Orientadas a documentos (*Document stores*)
- Orientadas a grafos (*Graph databases*)

3.5. Bases de Datos de Grafos

Habiendo estudiado lo anterior sabemos de la creciente necesidad de organizar la información, según [7], una representación de estos datos es en forma de grafos, la cual se presta bien para datos tanto estructurados como para los que tienen un esquema dinámico, donde explica que la teoría de los grafos se traduce en el trabajo actual en biología computacional y grafos sociales con consultas de ruta más cortas, agrupación, detección de comunidades y otros algoritmos de grafos. La optimización de estas consultas separa las bases de datos de grafos del resto.

La tendencia reciente, siguiendo el movimiento *NOSQL*, se ha alejado de las bases de datos relacionales a las más adecuadas para una aplicación determinada, de acuerdo a [7] también establece que, si bien gran parte de este movimiento se centra en la escalabilidad horizontal de los datos con organización por columnas y organización por valores, el modelo de datos de grafos proporciona un mayor nivel de complejidad de datos en comparación. La figura 21, muestra una categorización de modelos de datos *NOSQL*, comparando la complejidad de los datos versus el tamaño de los datos. Los modelos de datos de grafos proporcionan un mayor nivel de complejidad de datos a cambio de poder manejar menos datos.

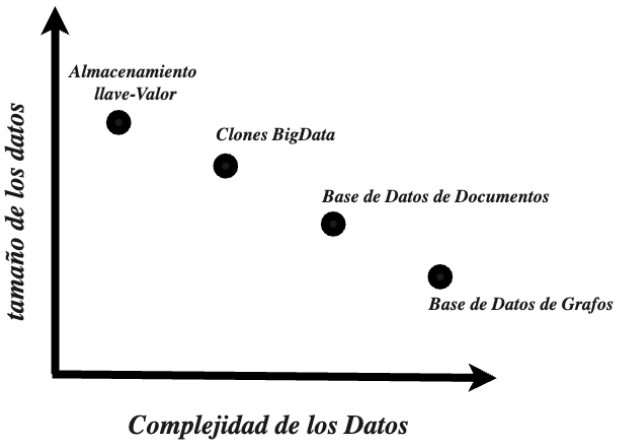


Figura 21 Comparativo en tamaño y complejidad de modelo de datos NOSQL [24]

El uso de grafos para almacenar datos no solo permite un esquema dinámico, sino que también proporciona representaciones de datos que anteriormente no eran posibles. La capacidad de superponer diferentes grafos (por ejemplo, social, temporal y espacial) en los datos amplía la funcionalidad de la consulta de datos [7].

Las redes sociales son un tema muy popular no solo en la sociedad, sino también en la investigación de grafos, las redes sociales no solo introducen un volumen enorme de datos, sino que también presentan grandes problemas a la comunidad investigadora durante el análisis de grafos debido a su gran tamaño. Estos grafos, no solo almacenan nodos con información de las personas, sino que también vinculan nodos de multimedia, relaciones y mensajes.

Existen diferentes modelos de bases de datos de grafos dependiendo de la organización de los datos, a continuación, mencionadas:

- Bases de datos de grafos
- Bases de datos de grafos distribuidas
- Bases de datos de grafos por valor
- Bases de datos de grafos de documentos
- Bases de datos de grafos de *SQL*
- Bases de datos de grafos *Map/Reduced*.

En [7], se explica que se han realizado múltiples estudios comparando el rendimiento de las bases de datos de grafos y las relacionales. Las bases de datos de grafos como *NEO4J* [28] se optimizan para consultas de adyacencia y recorrido de grafos. Si bien algunas operaciones en *NEO4J* pueden no ser tan rápidas como la indexación proporcionada en una base de datos *SQL*, el rendimiento mejorará mucho al realizar consultas tipo grafo, es decir creando consultas que consideren muchas relaciones, muchas características, tener características de árbol o requerir cambios frecuentes de esquema.

En una comparación de *NEO4J* y *MySQL* [29], los autores encontraron que las bases de datos de grafo funcionaron mejor que el modelo relacional en las consultas objetivas.

También se encontró, que cuando las comparaciones se centran en datos estructurados con grafos que son bastante densos, el rendimiento de indexación relacional con combinaciones ya no puede seguir el ritmo de la representación de datos vinculados en las bases de datos de grafo.

Las bases de datos de grafos *NEO4J* constituye una de las cuatro principales tecnologías de bases de datos conocidas colectivamente como *NOSQL*, y según [10] tiene las siguientes propiedades:

- Propiedad 1. *NEO4J* no tiene esquema.
- Propiedad 2. *NEO4J* cumple con los requisitos de una base de datos orientada a grafos.
- Propiedad 3. El modelo de grafos de propiedad es el modelo de datos conceptual primario de *NEO4J*.
- Propiedad 4. *NEO4J* admite *SPARQL*, un lenguaje de consulta *RDF W3C*, y *GREMLIN*, sin embargo, las consultas a un sistema *NEO4J* se envían principalmente en *CYPHER*, un lenguaje *ASCII* de arte, basado en patrones y declarativo.

La base de datos en grafos, nos permite la detección de comunidades, por ejemplo, encontrar el diámetro o el número de caminos más cortos que conectan dos vértices dados, según [30] uno de los algoritmos de grafos más utilizado es el de *Louvain* o algoritmo multinivel [31] donde se describe como un algoritmo de agrupamiento jerárquico que opera en gráficos ponderados. El objetivo es crear comunidades donde la densidad de relaciones es alta, mientras que la densidad intercomunitaria sigue siendo baja, el algoritmo de *Louvain* expresa la noción intuitiva de densidad de la relación con modularidad, se puede aplicar a grafos no ponderados, el resultado es siempre un grafo ponderado donde los pesos son proporcionales a la densidad de la relación local, si partimos de un grafo no ponderado se trata como un grafo ponderado con pesos iniciales iguales a uno.

De acuerdo con [32], el análisis de grafos grandes juega un papel destacado en varios campos de investigación y es relevante en muchas áreas de aplicación importantes. El análisis visual eficaz de los grafos requiere presentaciones visuales apropiadas en combinación con las respectivas facilidades de interacción del usuario y los métodos de análisis algorítmico de grafos.

Algunos gestores de bases de datos de grafos disponibles al momento de escribir este documento son:

- *NEO4J*
- *INFINITE GRAPH*
- *INFOGRID*
- *HYPERGRAPHDB*
- *DEX*
- *GRAPHBASE*
- *TRINITY*

3.6. NEO4J

NEO4J pertenece al mundo de las bases de datos *NOSQL* y está basado en el modelo de datos de la tienda de grafos. Es muy útil si quieres visualizar tus datos. La visualización de datos en grafos a menudo conduce a la identificación de diferentes patrones que se pasan por alto cuando vemos datos en formatos tabulares o textuales. Es un software de código abierto y el código fuente completo está disponible en *GITHUB*[14].

La plataforma de grafos de *NEO4J* se basa en la base de datos de grafos nativos**,** ver figura 22 para observar la integración de datos con este producto.

- La base de datos de grafos nativos de *NEO4J* es compatible con aplicaciones transaccionales y análisis de grafos.
- El análisis de grafos ayuda a los científicos de datos a obtener nuevas perspectivas sobre los datos.
- La integración de datos agiliza la destilación de datos tabulares y grandes datos en grafos.
- El lenguaje de consulta de grafos *CYPHER* es el puente hacia las herramientas analíticas de *big data.*
- La visualización y el descubrimiento de grafos ayudan a comunicar los beneficios de la tecnología de grafos en toda la organización.
- La arquitectura empresarial subyace y es compatible con datos grafos masivos.

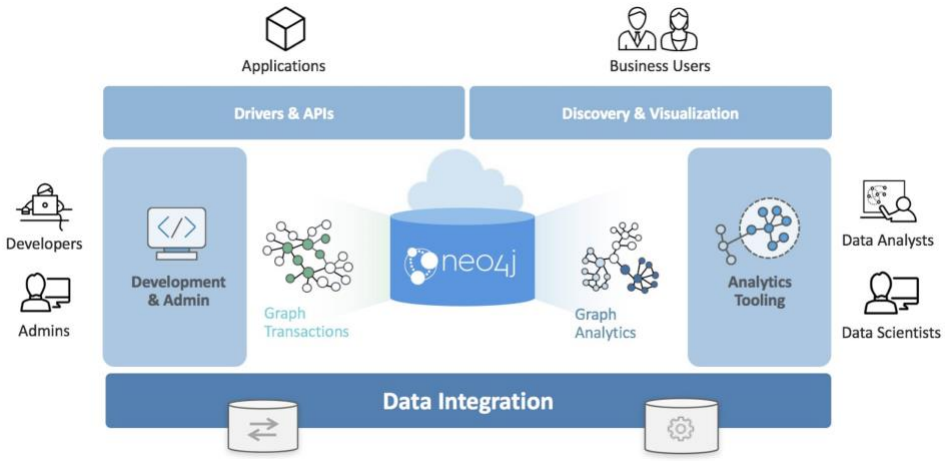


Figura 22 Integración de datos NEO4J [28].

Encontramos claramente definido en [28] que la plataforma de grafos de *NEO4J* está específicamente optimizada para mapear, analizar, almacenar y recorrer redes de datos conectados para revelar contextos invisibles y relaciones ocultas. Al mapear de forma intuitiva los puntos de datos y las conexiones entre ellos, *NEO4J* alimenta aplicaciones inteligentes en tiempo real que abordan los retos empresariales más difíciles de hoy en día, que incluyen:

- Inteligencia Artificial (*AI*) y aprendizaje automático
- Internet de las cosas (*IoT*)

- Recomendaciones en tiempo real y personalización
- *Master Data Management* (*MDM*)
- Detección de fraude
- Operaciones de red e *IT*
- Identidad y gestión de acceso

3.6.1. El lenguaje de consulta *CYPHER*

El lenguaje de consulta *CYPHER* es similar a *SQL*, pero específicamente para generar resultados de grafos como salida. Fue desarrollado por la empresa *Neo Technology* para *NEO4J*, pero luego se lanzó como un proyecto de código abierto llamado *openCYPHER* para que otras bases de datos de grafos también puedan utilizarlo en sus proyectos. Traduce consultas tradicionales muy complicadas en consultas simples de grafos *CQL(Common Query Language*).

En [14] se describe que hay tres componentes principales en un grafo, a saber:

- **Nodo.** Este es el objeto primario para representar cualquier dato en forma de grafo. Puede considerarlo como un elemento principal que hace a otros objetos. Representa entidades tales como persona, evento, lugar, cosa, etc. Si toma un ejemplo de un grafo de redes sociales, nodo es una representación de usted y sus amigos conectados entre sí.

- **Relación.** Es la representación de cómo se conectan los nodos y en qué tipo de conexión se encuentran. Si ampliamos el ejemplo anterior, donde usted y sus amigos están conectados entre sí, la relación define qué tipo de conexión es, si el conectado uno es amigo, colega, socio comercial, etc.

- **Propiedades.** Cada nodo y relación puede tener algunas propiedades. Por ejemplo, sus nodos en el ejemplo anterior pueden contener cierta información relacionada con usted, como su nombre, dirección, número de teléfono, etc. Están en forma de pares clave-valor.

Ver figura 23, donde podemos ver un ejemplo un grafo etiquetado y la figura 24, con los componentes de un grafo, aquí puede ver los nombres de los componentes del grafo en un ejemplo.

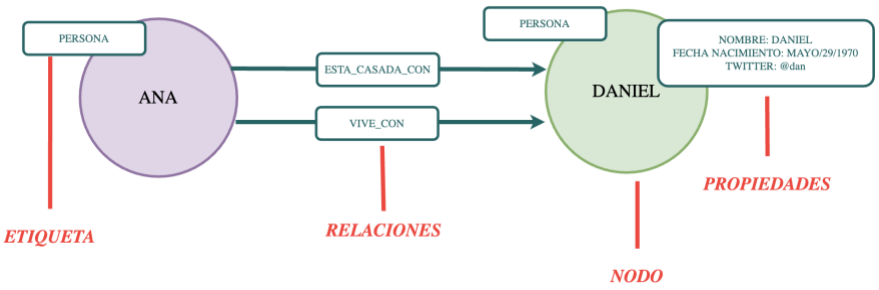


Figura 23 El modelo de grafo de propiedades etiquetadas [28].

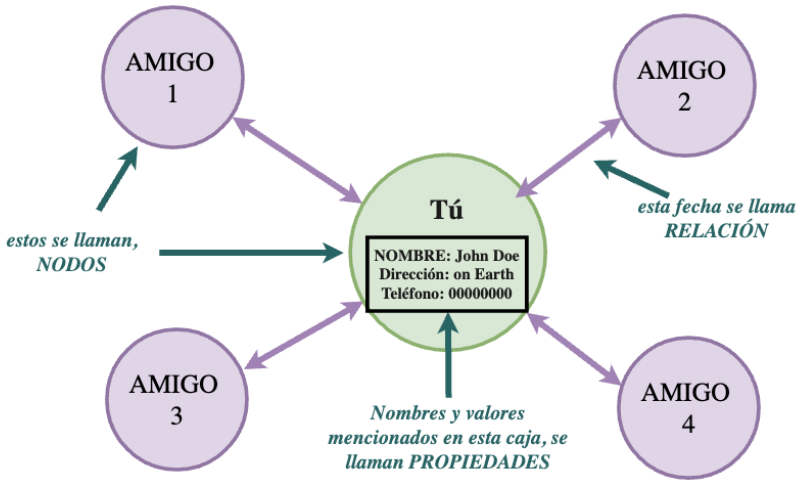


Figura 24 Componentes de un grafo [14].

3.7. Algoritmos de Grafos

Los algoritmos de grafos son un subconjunto de herramientas para análisis de grafos, análisis computacional y usos de la ciencia de datos, hay dos áreas de análisis que están en el corazón de los algoritmos grafos. Estas categorías corresponden a los algoritmos de cálculo de centralidad y detección de la comunidad [33].

3.7.1. Centralidad

La centralidad se trata de comprender qué nodos son más importantes en una red. Existen diferentes tipos de algoritmos de centralidad creados para medir diferentes cosas, como la capacidad de difundir rápidamente la información frente a unir distintos grupos. Los algoritmos de centralidad se utilizan para comprender los roles de nodos particulares en un grafo y su impacto en esa red. Son útiles porque identifican los nodos más importantes y nos ayudan a comprender la dinámica del grupo, como la credibilidad, la accesibilidad, la velocidad a la que se extienden las cosas y los puentes entre los grupos. Aunque muchos de

estos algoritmos fueron inventados para el análisis de redes sociales, desde entonces han encontrado usos en una variedad de industrias y campos[33].

Algunos algoritmos de centralidad, ver figura 26, son:

- Grado de centralidad (*Degree* por sus siglas en inglés), como medida de conexión básica
- Cercanía Centralidad (*Closeness* por sus siglas en inglés), para medir qué tan central es un nodo para el grupo.
- Centralidad de intermediación (*Betwenness* por sus siglas en inglés), para encontrar puntos de control.
- *PageRank* para comprender la influencia general.

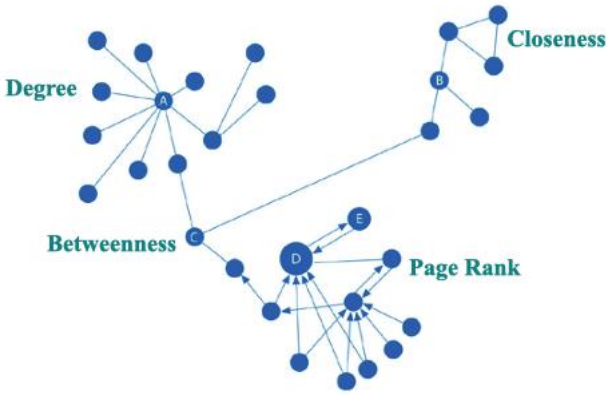


Figura 25 Algoritmos de centralidad representativos[30].

3.7.2. Detección de comunidades

La conectividad es un concepto central de la teoría de grafos que permite un análisis de red sofisticado, como la búsqueda de comunidades. La mayoría de las redes del mundo real exhiben subestructuras de subgrafos más o menos independientes. La conectividad se utiliza para encontrar comunidades y cuantificar la calidad de las agrupaciones. La evaluación de diferentes tipos de comunidades dentro de un grafo puede descubrir estructuras, como centros y jerarquías, y tendencias de grupos para atraer o repeler a otros. Estas técnicas se utilizan para estudiar fenómenos emergentes como los que conducen a cámaras de eco y efectos de burbujas de filtro.

La formación de comunidades es común en todo tipo de redes, y su identificación es esencial para evaluar el comportamiento del grupo y los fenómenos emergentes. El principio general para encontrar comunidades es que sus miembros tendrán más relaciones dentro del grupo que con nodos fuera de su grupo. La identificación de estos conjuntos relacionados revela grupos de nodos, grupos aislados y estructura de red. Esta información ayuda a inferir comportamientos o preferencias similares de los grupos de pares, estimar la capacidad de recuperación, encontrar relaciones anidadas y preparar datos para otros análisis. Los

algoritmos de detección de la comunidad también se usan comúnmente para producir la visualización de la red para la inspección general[33].

Los algoritmos de detección de la comunidad más representativos:

- **Componentes fuertemente conectados y componentes conectados** para encontrar clústeres conectados, ver figura 27- Propagación de etiquetas sobre multiples iteraciones [33].

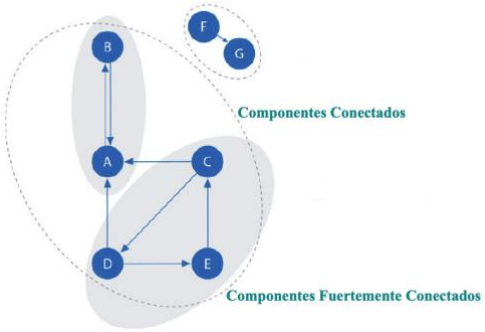


Figura 26 grafo mostrando Componentes fuertemente conectados y componentes conectados [30].

- **Propagación de etiquetas** para inferir rápidamente grupos basados en etiquetas de nodo, ver figura 28.

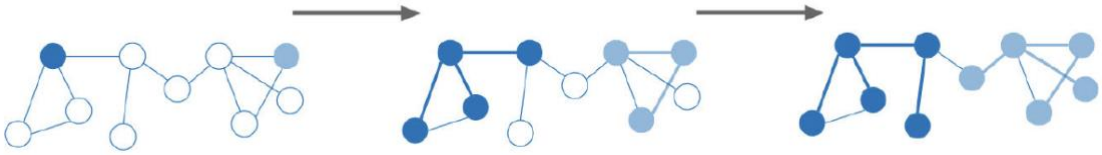


Figura 27 Propagación de etiquetas sobre múltiples iteraciones [30].

- **Modularidad de *Louvain*** para observar la calidad y las jerarquías de agrupación, ver figura 29.

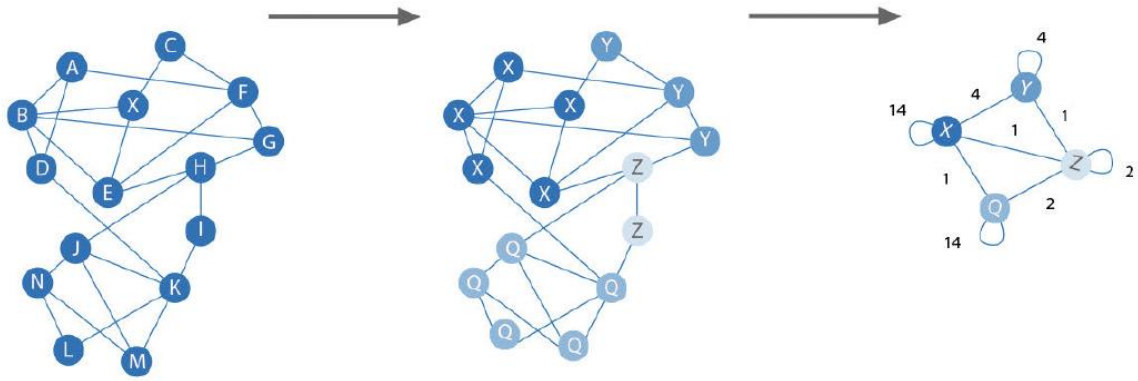


Figura 28 Modularidad de *Louvain* sobre múltiples iteraciones[30].

Hoy en día el desarrollo de sitios *Web* ha recorrido un largo camino desde el comienzo de la *World Wide Web (WWW*), muchas tecnologías de programación y lenguajes ahora se están utilizando para crear aplicaciones *Web*. Una de las tecnologías para crear aplicaciones *Web* es *MVC,* es un método para crear *Web* separando la capa del modelo, la capa del controlador y la capa de visualización para que sea más fácil y rápido.

MVC ha mostrado sus beneficios para las aplicaciones *Web* interactivas que permiten múltiples representaciones de la misma información, promueven la reutilización del código y ayudan a los desarrolladores a centrarse en ciertas características de la aplicación. El *framework MVC* se ha convertido ampliamente en un estándar en el desarrollo de software moderno[5].

3.8. Modelo *MVC* usando *PYTHON* para el desarrollo de *framework AIOHTTP*

En el modelo *MVC* se pueden separar los datos, mostrar, controlar el software y también puede alcanzar la separación de la capa de lógica de negocios y la capa de presentación. Esto permite que se pueda desarrollar una solución de software de manera efectiva utilizando este *framework MVC*, también es estable, eficiente y capaz de desarrollar aplicaciones de alta calidad[34].

En [35] se establece que un modelo para desarrollar aplicaciones *Web* rápidas basadas en la arquitectura Modelo-Vista-Controlador tiene varios componentes útiles como seguridad, validación y creación de formularios, así como acceso y enrutamiento de bases de datos. El modelo *MVC* propuesto en este *TOG* utilizamos el lenguaje de programación *JAVASCRIPT*.

Como parte de esta investigación, se diseñará un *MVC* en el *framework* de *AIOHTTP* utilizando el lenguaje *PYTHON*, esta combinación pretende ayudar a los desarrolladores a mejorar la velocidad y la calidad del trabajo, y proporcionar un *framework* de *AIOHTTP* que sea mejor tanto para usuarios novatos como experimentados.

3.8.1. Descripción general del *MVC*

En [6] se define que este método se divide en tres partes interconectadas, a saber, el modelo, la vista y el controlador. La Figura 30 muestra el flujo del modelo-vista-controlador.

a. **Modelo.** Tratar directamente con bases de datos para manipular datos (insertar, actualizar, eliminar, buscar), manejar la validación desde las partes del controlador, pero no puede tratar directamente con la sección de vista.

b. **Vista.** La parte que maneja la lógica de presentación. En una aplicación *Web*, esta sección suele ser un archivo de plantilla *HTML*, que establece el

controlador. Ver funciones para recibir y representar datos para el usuario. Esta sección no tiene acceso directo a la sección del modelo.

c. **Controlador.** La parte que regula la relación entre la parte del modelo y la parte de la vista, el controlador funciona para recibir solicitudes y datos del usuario y luego determinar qué procesará la aplicación.

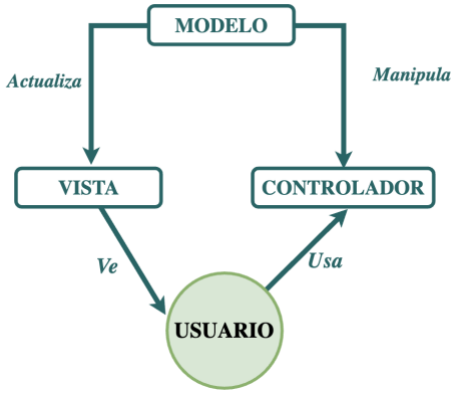


Figura 29 El patrón Modelo-Vista-Controlador [6] .

3.9. Servicios *Web*

Un servicio *Web* es un componente de software que se basa en estándares abiertos y es independiente de cualquier hardware, plataforma, lenguaje de desarrollo o sistema operativo. Puede variar de simple a complejo en función de las operaciones realizadas por el propio servicio. También se considera un componente de auto descripción, ya que no requiere ningún tipo de esquema para describir sus datos [36]. Y, en última instancia, los servicios prestados se otorgan con la ayuda de una secuencia de mensajes de solicitud respuesta[37].

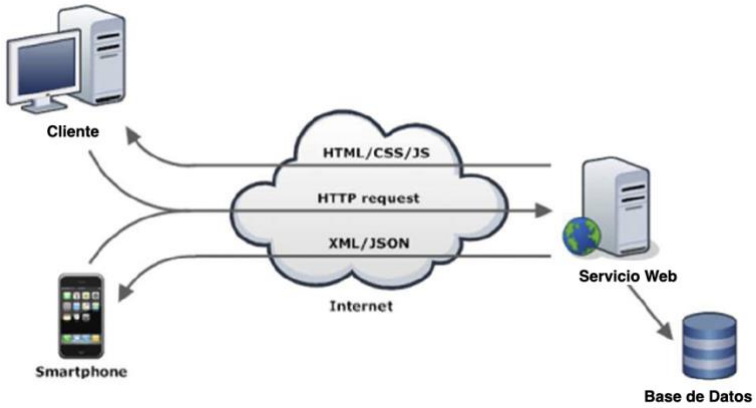
Según[38], los desarrolladores seguramente pueden beneficiarse de los servicios *Web*, en muchos aspectos diferentes, por ejemplo, un servicio *Web* proporciona una solución única de su tipo para integrar varias aplicaciones juntas sin la necesidad de tener un *middleware* para cada integración de aplicación a aplicación. Esto se debe a los estándares abiertos que se utilizan principalmente en los servicios Web, lo que lo hace comprensible para todas las aplicaciones. Por lo tanto, las aplicaciones ahora pueden comunicarse entre sí sin tener en cuenta los sistemas operativos en los que se ejecutan, el entorno de desarrollo utilizado para desarrollar esas aplicaciones o el *hardware* utilizado [39].

En términos de reducción de la complejidad, la actualización de un servicio específico no tendrá un impacto significativo en ningún otro servicio, siempre que estén separados. Esto también se aplica a la capacidad de mantenimiento, ya que dividir un sistema completo en varios servicios reduciría la complejidad de encontrar errores e intentaría solucionarlos.

Debido al hecho de que los servicios *Web* se basan en estándares abiertos, cualquier aplicación con menos consideración a sus especificaciones puede interactuar con el servicio (es decir, enviar y recibir datos) [36]. Esto implica que los servicios *Web* se pueden usar una y otra vez, sin tener que cambiar ningún código único, y como las aplicaciones ahora actuarán como nada más que consumidores, y, por lo tanto, la reutilización sería otro beneficio de los servicios *Web*.

3.9.1. Arquitectura de *REST*

REST se basa únicamente en la denominada arquitectura cliente/servidor, ver figura 31. El cliente solicita información del servidor y el servidor responde al servidor esa solicitud particular. Además, algunas solicitudes enviadas desde el cliente deben incluir algunos parámetros que el servidor podría utilizar como entrada para producir una salida correcta. *REST* se basa principalmente en el protocolo *HTTP* estándar para el transporte de mensajes de solicitud / respuesta. Además, todos los comandos *HTTP* como *GET, PUT, POST y DELETE* son compatibles con *REST*. La Figura 3 muestra la interacción entre diferentes clientes y un servidor *Web REST* [38].



FLASK Figura 30 Clientes que se comunican con un servidor REST [40].

Como se puede ver en la figura anterior, las solicitudes provenientes de diferentes dispositivos se comunicarán con el servidor *REST* a través del protocolo *HTTP*. Y el servidor manejará esas solicitudes por separado simplemente consultando todos los datos necesarios, ordenándolos y presentándolos al cliente en un mensaje de respuesta.

Igualmente [34] explica que *REST*, tiene tres elementos principales, que son: recurso, *URI* y presentación:

- **Recurso**. un recurso puede ser de cualquier tipo, como una página *Web*, una imagen, un archivo o una secuencia de datos.

- ***URI***. que puede considerarse como una *URL* que apuntará a un recurso particular (ubicación del host).

- **Presentación**. un aspecto importante de la arquitectura *REST* es la presentación de datos. Con el servidor *REST*, un cliente puede especificar qué estándar de presentación se utilizará para ordenar la salida, a fin de facilitar que el cliente interprete los datos recibidos. Un estándar de presentación puede ser, por ejemplo, los formatos: *HTML, XML, JSON, CSS, JAVASCRIPT*, etc.

En el diseño de aplicaciones *Web*, y para el consumo eficiente de servicios *Web*, como el servicio de *REST* anteriormente descrito, en este proyecto, usaremos el lenguaje de programación *PYTHON* y la librería de *AIOHTTP* para su implementación.

3.10. PYTHON

El atractivo de *PYTHON* radica en su simplicidad y belleza, así como en la conveniencia del gran ecosistema de herramientas específicas de dominio que se han construido sobre él, fue concebido a fines de la década de 1980 como un lenguaje de enseñanza y *scripting*, *PYTHON* se ha convertido desde entonces en una herramienta esencial para muchos programadores, ingenieros, investigadores y científicos de datos de la academia y la industria. Como astrónomo enfocado en construir y promover las herramientas abiertas gratuitas para la ciencia intensiva en datos.

En [41], se describe que *PYTHON* es útil para una amplia gama de tareas:

- **Datos.** PYTHON es muy bueno para recopilar datos, luego limpiar, ajustar, analizar y predecir con esos datos. Puede usarlo con herramientas como *PANDAS, MATPLOTLIB, JUPYTER, SCIKIT-LEARN y TENSORFLOW*.

- ***Operations.*** PYTHON es muy adecuado para automatizar y mantener la infraestructura en funcionamiento. Puede usarlo con herramientas como *SALTSTACK, ANSIBLE, BOTO3* y muchas bibliotecas de terceros.

- **Desarrollo Web.** PYTHON se usa para construir y mantener sitios Web; Los ejemplos incluyen *YOUTUBE, QUOTA, INSTAGRAN, PINTEREST y DROPBOX*. Puede usar herramientas como *DJANGO*, *FLASK* y *AWS LAMBDA*.

Hablando específicamente de las capacidades de *PYTHON*, en el desarrollo Web, que es una de las áreas de aplicación de este trabajo, *PYTHON* es capaz de solicitar información de un servidor *Web*, realiza un manejo básico de la respuesta del servidor e interactuar con los sitios de manera automatizada[42].

Este *TOG*, hace uso de la práctica *Web Scrapping* para recopilar datos a través de cualquier otro medio que no sea un programa que interactúa con una *API* (o, obviamente, a través de un humano usando un navegador *Web*). Esto se logra más comúnmente escribiendo un programa automatizado que consulta un servidor Web, solicita datos (generalmente en forma de *HTML* y otros archivos que componen las páginas *Web*) y luego analiza esos datos para

extraer la información necesaria, el *Web Scrapping* es un tema relativamente disparejo, con prácticas que requieren el uso de bases de datos, servidores *Web*, *HTTP*, *HTML*, seguridad de Internet, procesamiento de imágenes, ciencia de datos y otras herramientas[42].

3.11. Librería AIOHTTP

AIOHTTP es un micro *framework* Cliente/servidor *HTTP* asíncrono que pueden simplificar enormemente el proceso de escribir servidores cuando se utiliza un enfoque basado en eventos, esta librería fue escrita en el lenguaje de programación *PYTHON*, fue creado mayormente por *Nikolay Kim* and *Andrew Svetlov*. *AIOHTTP* está diseñado para hacer aplicaciones Web de manera rápida y fácil, con la capacidad de mejorar aplicaciones complejas.

La herramienta principal para realizar solicitudes es el módulo de *requests*. El principal problema con las solicitudes es que el hilo está bloqueado hasta que obtengamos una respuesta. Por defecto, las operaciones de solicitud están bloqueando. Cuando el hilo llama a un método como *GET* o *POST*, se detiene hasta que se completa la operación. Para descargar múltiples recursos a la vez, necesitamos muchos hilos. En este punto, *AIOHTTP* nos permite hacer solicitudes de forma asincrónica [43].

ClientSession es la interfaz principal recomendada para que *AIOHTTP* realice solicitudes. Esta interfaz le permite almacenar *cookies* entre solicitudes y mantiene objetos que son comunes para todas las solicitudes (bucle de eventos, conexión y recursos de acceso). Después de abrir una sesión de cliente, puede usarla para realizar solicitudes. El administrador de contexto con declaración asegura que se cerrará correctamente en todos los casos [43].

3.12. Visualización de Datos

En [44] se resalta que la investigación de visualización de datos se centra en la exploración y análisis de datos, sin embargo, la gran mayoría de las visualizaciones que las personas ven fueron creadas para un propósito diferente: presentación. Ya sea que estemos hablando de gráficos que muestran datos para ayudar a hacer una presentación del presentador, visuales de datos creados para acompañar una noticia, o las infografías, muchas más personas consumen gráficos de los que los crean.

Las técnicas utilizadas para presentar los datos son principalmente las utilizadas en el análisis: gráficos de barras, gráficos de líneas, etc. Aunque los entendemos bien, esa comprensión se basa en su papel en el análisis.

Por supuesto[44], también explica que es importante contar con técnicas de análisis que no se conviertan en un lío ilegible cuando aplique un filtro o cambie el mapeo en uno de sus ejes, lo que interrumpiría el flujo del análisis, es aquí donde en este proyecto implementamos el uso de la librería de visualización *D3* para generar gráficas altamente configurables y

personalizables para desplegar los datos de diferentes maneras e impulsar su sencilla interpretación.

3.13. Visualización de Grafos

La visualización de grafos se concentra en el desarrollo de diseños de grafos efectivos y mapeos visuales. La visualización de grandes grafos se acompaña de técnicas de interacción efectivas, en particular, en los casos en que todo el grafo es demasiado complejo o grande para ser visualizado en una vista estática, en ocasiones, la interacción por sí sola puede no ser suficiente para lograr ciertas tareas analíticas.

Según [32] existen tres elementos que forman la base para sistemas efectivos de análisis de grafos visuales y están estrechamente relacionados, esto son: representación visual, interacción del usuario y análisis algorítmico.

En la visualización de grafos el pre-procesamiento algorítmico a menudo incluye la simplificación de gráficos para reducir el tamaño, mientras se mantiene la estructura principal del gráfico. También se puede utilizar el pre-procesamiento de las propiedades del gráfico para la visualización del grafo (en algoritmos para posicionamiento de nodos y aristas) o para resaltar partes interesantes del gráfico. El sub-grafo se usa entonces para una inspección visual más fácil, ya que los grafos grandes y complejos son difíciles de entender incluso utilizando algoritmos avanzados de posicionamiento de relaciones y nodos (diseños).

Las técnicas para mostrar gráficos generales se pueden dividir en tres grupos principales: basados en enlaces de nodo, basados en matriz e híbridos.

En [45] presentan una comparación de técnicas de enlace de nodo y matriz de adyacencia. Según el estudio, las ventajas de los diagramas de enlace de nodo son su intuición, compacidad y una mejor idoneidad para las tareas de seguimiento de ruta, son más efectivos para grafos más pequeños y dispersos. La representación de matriz de adyacencia inherentemente no tiene cruces de relaciones y problemas de superposición de nodos, y por lo tanto es adecuada también para grafos densos. Cuando se utiliza el orden de nodos apropiado, pueden revelar fácilmente subestructuras densas en el grafo. Sin embargo, también sufren de escalabilidad en espacios de visualización limitados, especialmente para grafos muy grandes.

En el análisis gráfico visual, el diseño gráfico y el orden de la matriz influyen en la efectividad de estas representaciones. Estos problemas están, por lo tanto, en el núcleo de la investigación de visualización de grafos [32].

3.14. Librería *D3*

D3 le permite vincular datos arbitrarios a un Modelo de Objetos de Documento (*DOM*) y luego aplicar transformaciones basadas en datos al documento. Por ejemplo, puede usar *D3* para generar una tabla *HTML* a partir de una matriz de números [46], *D3.js* es una biblioteca

de *JAVASCRIPT* que le permite crear gráficos y visualizaciones de datos en el navegador con *HTML, SVG(Scalable Vector Graphics*) y *CSS(*Cascading Style Sheet*)*[47].

D3 no es un marco monolítico que busca proporcionar todas las características imaginables. En cambio, *D3* resuelve el gran problema: la manipulación eficiente de documentos basados en datos. Esto evita la representación patentada y ofrece una flexibilidad extraordinaria, exponiendo todas las capacidades de los estándares *Web* como *HTML, SVG y CSS*. Con una sobrecarga mínima, *D3* es extremadamente rápido y admite grandes conjuntos de datos y comportamientos dinámicos para la interacción y la animación. El estilo funcional de *D3* permite la reutilización de código a través de una colección diversa de módulos oficiales y desarrollados por la comunidad [46].

Este también simplifica algunas de las tareas más comunes, así como algunas de las más complejas, que un desarrollador puede ejecutar al crear visualizaciones basadas en el navegador. En esencia, asigna fácilmente las propiedades de imagen *SVG* a valores de datos. A medida que cambian los valores de los datos, debido a las interacciones del usuario, también cambian las imágenes, ver figura 32. *D3* es una biblioteca masiva, llena de millones de opciones, pero sus conceptos básicos son fáciles de aprender. No necesita conocer todos los detalles de la biblioteca para convertirse en un desarrollador funcional de *D3*[47].

Figura 31 Demostración de algunos modelos gráficos creados usando la librería *D3*[48].

En este capítulo estudiamos los conceptos técnicos de todos los componentes de esta solución y entendemos la razón por la cual aportan de modo significativo a la misma, en el siguiente capitulo se describe la solución a nivel intermedio, entenderemos las tecnologías implementadas, la versión usada, y podrá ver el diagrama de arquitectura explicando la integración de tecnologías y su aplicación.

4. *FRAMEWORK* DE ANÁLISIS Y VISUALIZACIÓN DE GRAFOS

En este capítulo se describe de modo detallado la implementación técnica de las tecnologías consumidas, el diagrama de arquitectura de todo lo que esta solución integra así como una serie de imágenes de la interfaz grafica del usuario donde se pueden representar las capacidades y características disponibles en la pagina principal del usuario, y en la interfaz para manipulación de modelos de visualización.

4.1. Descripción de la Solución

La solución informática propuesta en este proyecto hace aplicación de múltiples tecnologías y *frameworks*, está distribuida en tres capas, *back-end, middleware y front-end*. A continuación, se describen.

4.1.1.1. Back-end

Consta de una base de datos de grafos. La instalada es NEO4J usa *CYPHER* como lenguaje de consultas y una de las características más destacables de *NEO4J* que son ampliamente explotadas en esta solución, es la disponibilidad de ejecutar algoritmos de base de datos que permiten un profundo análisis del grafo. En la tabla IX se muestran las versiones del software utilizados para desarrollar el proyecto. Refiera a la tabla X para ver las características generales de *NEO4J* que son relevantes para este proyecto.

Producto	**Versión**
NEO4J Enterprise	3.5.17
The Graph Algorithms library	3.5.4.0
PYTHON	3.8.2
PYTHON Modules:	
AIOHTTP	3.6.2
NEO4J	1.7.6

networkx	2.4
google.cloud	0.34.0
google-cloud-language	1.3.0
BOOTSTRAP	3.0

Tabla IX Nombre de Productos y librerías implementadas en esta solución.

Neo4J es una base de datos de grafos que usa CYPHER como lenguaje de consulta para interaccion con los datos, este disponible a través del servicio BOLT para su implementación dentro de servicios y/o programas, NEO4J permite la ejecución de poderos algoritmos de grafos, a continuación, los que implementaremos en este trabajo:

- *PageRank*
- *Betweenness Centrality*
- *Louvain*
- *Label Propagation*
- *Connected Components*
- *Degree*
- *In-degree*
- *Out-degree*

4.1.1.2. Middleware

Es un servicio Web escrito en el lenguaje de programación de PYTHON que junto con las librerías de AIOHTTTP permiten que funcione como tal, cuenta con 3 APIs disponibles por medio de llamadas REST desde cualquier navegador Web, sirven datos de la base de datos de grafos NEO4J y regresan archivos en formato JSONP a la aplicación Web que los llame .Además, mediante archivos JSON es posible definir variables de ambiente y seguridad. Refiera a la tabla X para ver un resumen de las características más destacables del middleware.

Características	**Librerías**

• Escrito en lenguaje *PYTHON3*
• Algoritmos para ejecutar configurables en un archivo tipo *JSON*
• 4 servicios disponibles:
  • ejecución de consultas
  • ejecución de algoritmos
  • análisis de sentimientos
  • obtención de esquema de la base de datos
• Autenticación configurable
	• *NEO4J*
	• *AIOHTTP*
	• *GOOGLE CLOUD*
	• *NETWORKX*
	• *PYSIMPLEGUI*

Tabla X Características del *Middleware*

4.1.1.3. Front-End

Una interfaz escrita en *JQUERY/JAVASCRIPT*, que es fácil de usar y amigable con el usuario que permite personalizar los algoritmos de consulta *CYPHER*, agregar atributos de análisis de sentimientos por medio de la *API* de *Google Sentiment*, así como ejecutar consultas *CYPHER* de forma libre, una vez ejecutados la aplicación *Web* permite al usuario a personalizar la visualización de dichos datos de acuerdo con los atributos extraídos, combinándolos en cualquier forma que el usuario prefiera, estos modelos están desarrollados en las librerías *D3*. Refiera la tabla XI para el resumen de las características del *front-end*.

Características	Utilerías/Librerías
• Una página *Web* con la aplicación que se conecta al servicio de *middleware* y extrae datos. • Una página *Web* para la visualización de datos del grafo. • Cada modelo de visualización se despliega en una nueva página *Web*. • Los datos se transfieren entre *middleware* y *front-end* a través de archivos *JSONP*. • Los datos entre las dos páginas *de front-end* se transfieren en archivos tipo *JSON*. • Integración de múltiples tecnologías en una misma solución.	• *JAVASCRIPT* • *BOOTSTRAP* • *D3* • *HTML* • *CSS* • *PHP*

Tabla XI Características del *Front-End.*

4.2. Diagrama de Arquitectura

La Figura 33, representa el diagrama de arquitectura de la solución general de este trabajo de obtención de grado, como punto central esta la interfaz web donde el usuario accede a los datos y posteriormente(lado derecho de la figura) los visualiza en diferentes modelos desarrollados usando D3 que es un es una librería JavaScript para manipular y visualizar documentos basados en datos, del lado izquierdo, tenemos la arquitectura del servicio web desarrollado en PYTHON que se conecta a la base de datos de grafos NEO4J para extraer datos y/o ejecutar algoritmos de grafos, y también a la API de Google para hacer análisis de sentimientos.

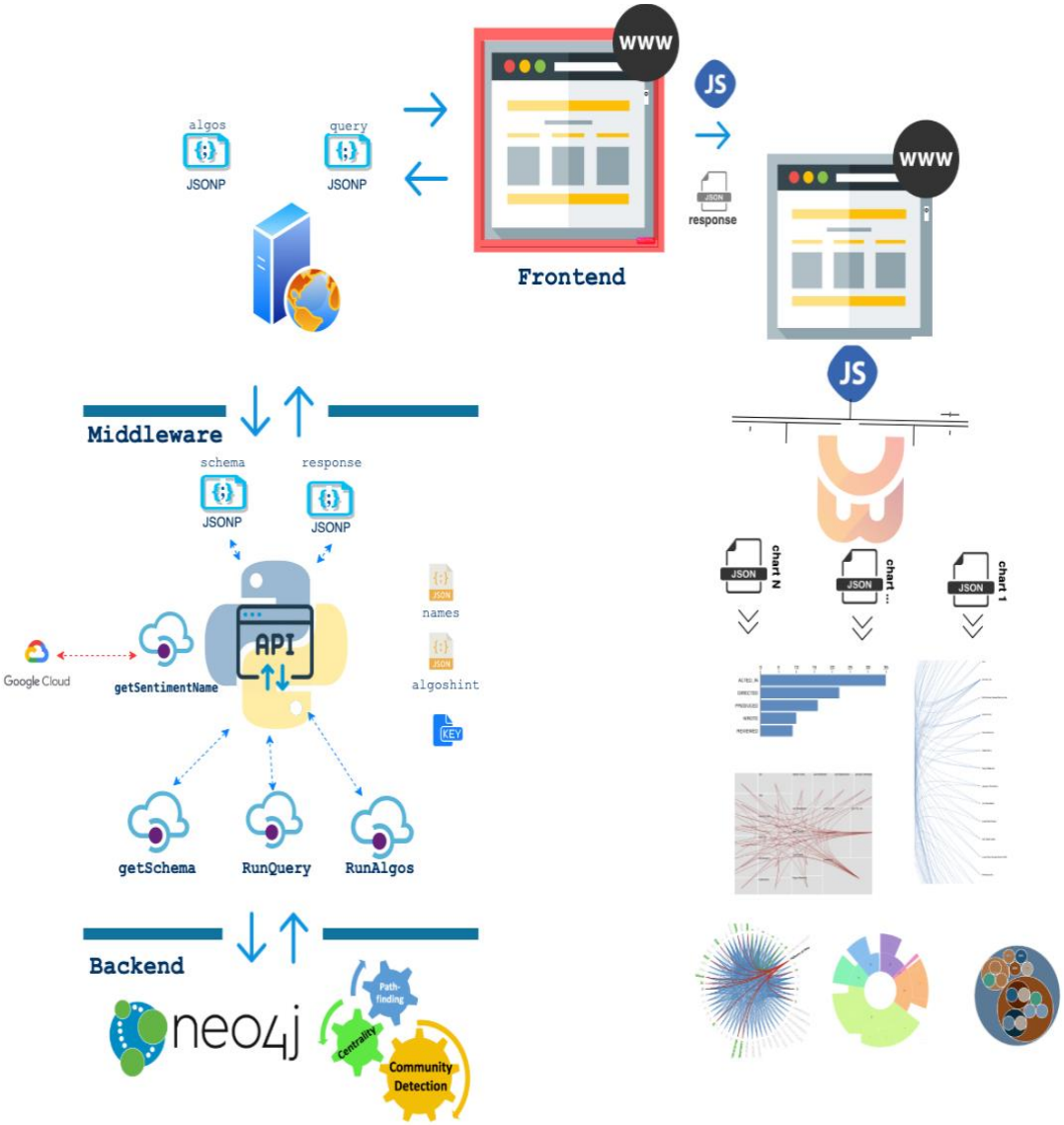


Figura 32 Diagrama de Arquitectura.

4.3. Diseño del Sistema

En esta sección se describe las funcionalidades de la solución.

4.3.1. Página principal de la aplicación

La interfaz *Web* consta de cuatro secciones en donde el usuario podrá visualizar el esquema de la base de datos que esta disponible en *NEO4J*, otra sección, donde se encuentran los algoritmos de grafos disponibles a ser ejecutados, así como una sección para que el usuario haga extracción de datos a su criterio y por último, la sección donde podemos ejecutar análisis de sentimientos sobre los atributos deseados por el usuario.

4.3.1.1. Extraer el esquema de la base de datos de grafos cargada en *NEO4J*

La figura 34 muestra el ejemplo de esquema de la base de datos *Movies*, extraído desde la aplicación, la información disponible es:

- Nodos
- Atributos
- Relaciones
- Atributos de las Relaciones

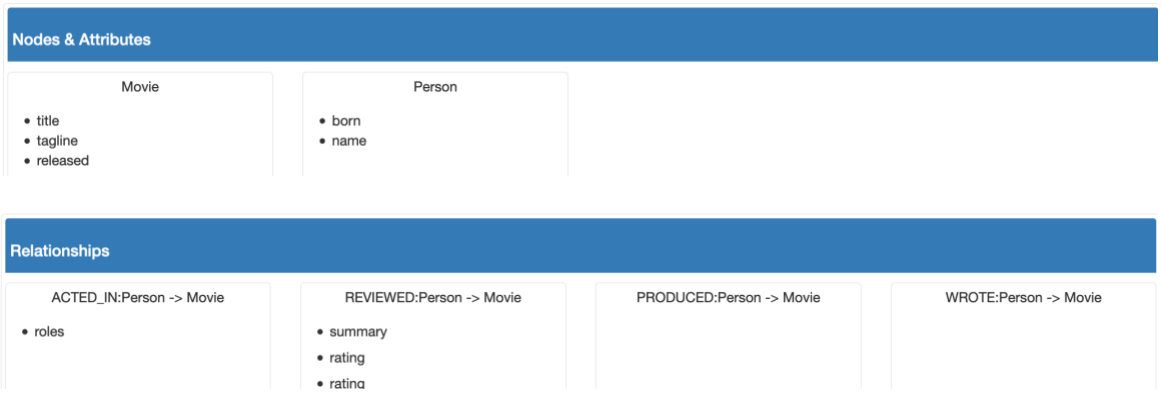


Figura 33 Captura de Pantalla aplicación donde muestra el esquema de base de datos: *Movies*.

4.3.1.2. Genera una propuesta de ejecución en CYPHER

La aplicación genera una propuesta de ejecución en *CYPHER* para cada uno de los algoritmos de grafos disponibles, permite al usuario ajustar libremente cada algoritmo a ejecutar, una vez ejecutado permite hacer uso del nuevo atributo de forma inmediata. La figura 35 muestra la sugerencia generada automáticamente de ejecución de algoritmos en base a la base de datos disponible en *NEO4J*.

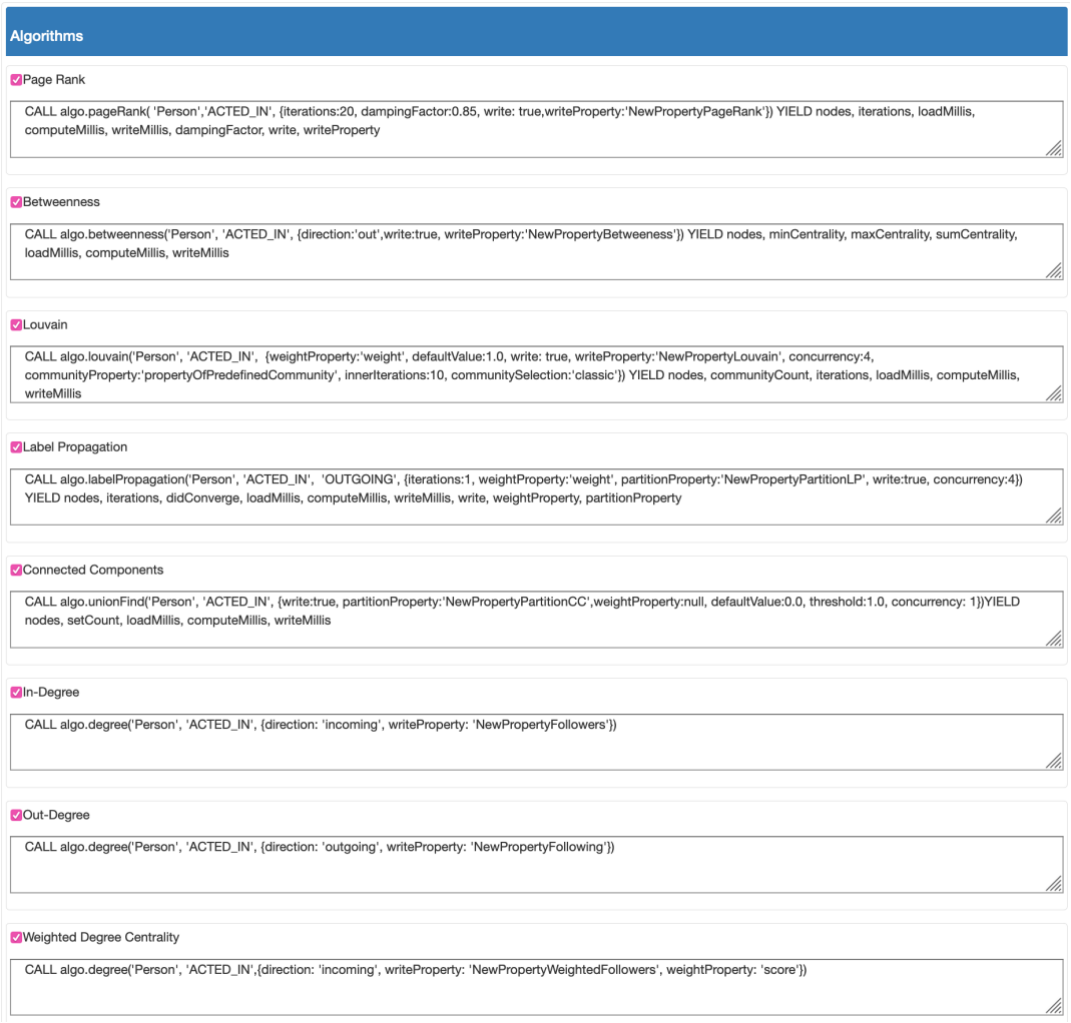


Figura 34 Captura de Pantalla de sugerencia de algoritmos.

4.3.1.3. Área de Análisis de Sentimiento sobre Atributos

Permite hacer análisis de sentimientos sobre cualquier atributo de la *BDG*, y crea un atributo nuevo después del análisis, igualmente, este nuevo atributo puede ser usado inmediatamente. La figura 36 muestra un extracto de la aplicación donde se observan los atributos disponibles para análisis de sentimientos.

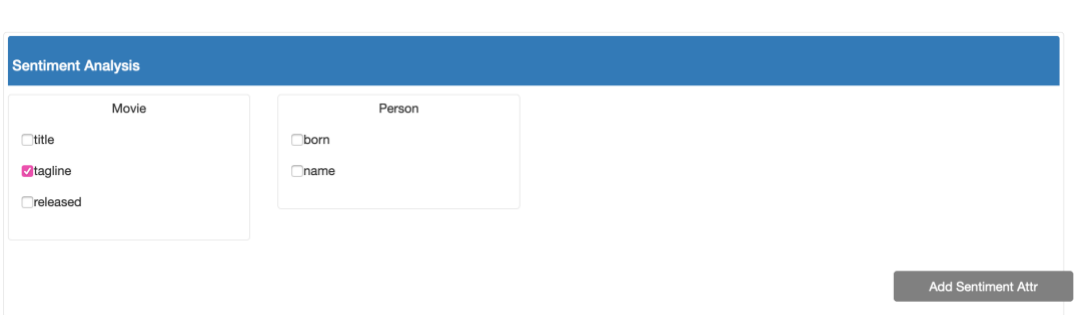


Figura 35 Captura de Pantalla para análisis de sentimientos.

4.3.1.4. Ejecución de consultas de forma libre

El usuario puede diseñar cualquier consulta en lenguaje *CYPHER* sin restricción, una vez ejecutada esta consulta, el usuario será dirigido a una nueva página para la visualización de dichos datos. La figura 37 muestra un extracto de la aplicación donde el usuario puede escribir una consulta en *CYPHER* y ejecutara contra *NEO4J*, una vez obtenidos los datos, el usuario es redirigido a otra pantalla para visualizar dicha extracción de datos.

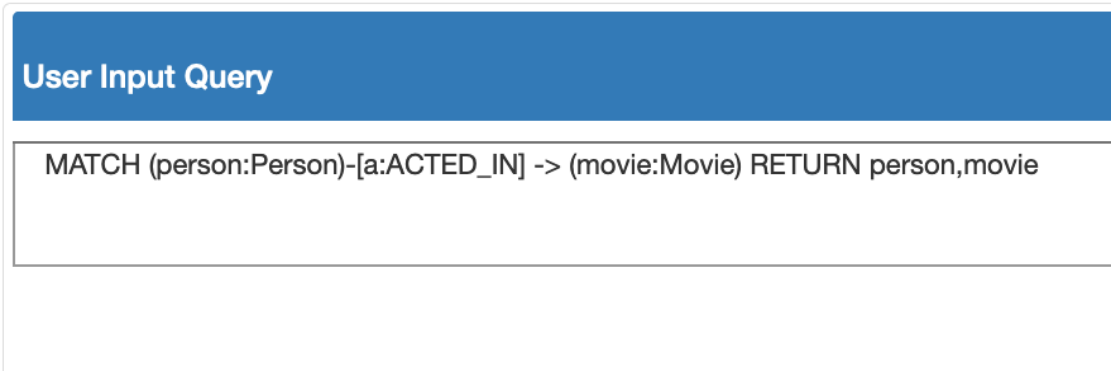


Figura 36 Captura de Pantalla donde el usuario puede ejecutar cualquier consulta.

4.3.2. Página de Visualización de datos extraídos de la base de datos de grafos.

En esta aplicación permite elegir entre siete modelos de visualización para analizar los datos, refiera la tabla XIII, pueden ser todos al mismo tiempo también, el usuario podrá seleccionar entre todos los atributos extraídos en la consulta para cada modelo, revise la figura 38 donde puede ver una captura de pantalla donde el usuario puede seleccionar los atributos a cargar en la gráfica(s) que desee analizar.

Modelos de Visualización
1. *Knowledge Map* 2. *Zoomable Sunburst* 3. *Hierarchical Bar* 4. *Hierarchical Edge Bundle* 5. *Hierarchical Edge Tree* 6. *Hierarchical Edge Vertical* 7. Table Graph

Tabla XII Modelos de Visualización disponibles.

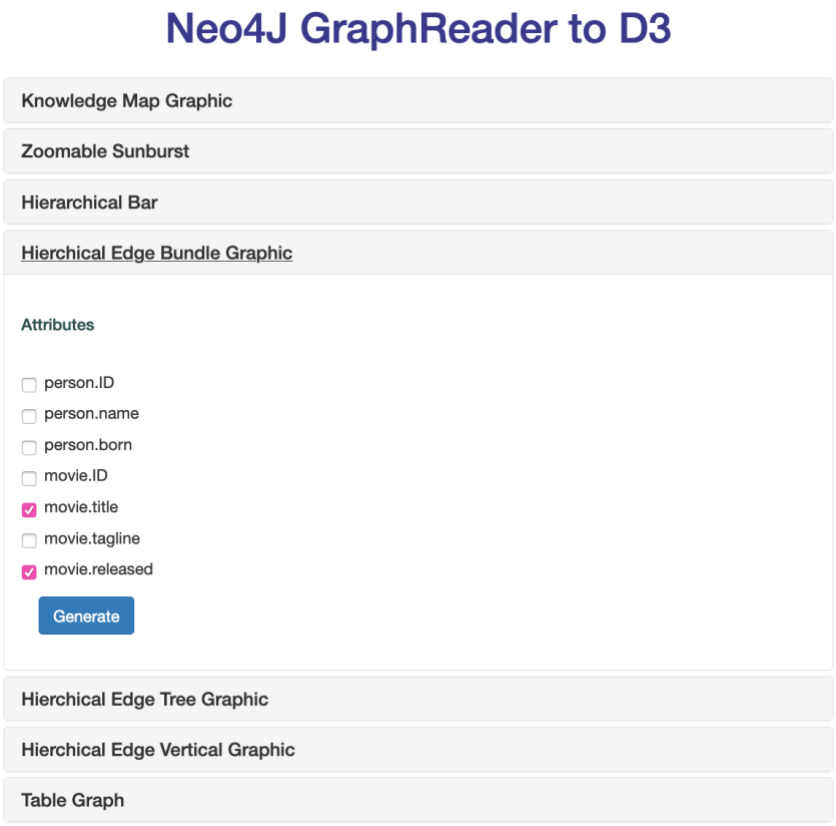


Figura 37 Captura de Pantalla de Visualizador, extendido el modelo *Hierarchical Edge Bundle* donde puede ver que todos los atributos del grafo extraído están disponibles.

4.3.2.1. Modelo *Knowledge Map*

Permite la agrupación de valores y los organiza en niveles, tantos como atributos y relaciones haya disponibles, la figura 39 representa la agrupación en este modelo de gráfica, y está acompañado de una tabla de referencia, a la cual se puede acceder directamente desde el visualizador *Table Graph.*

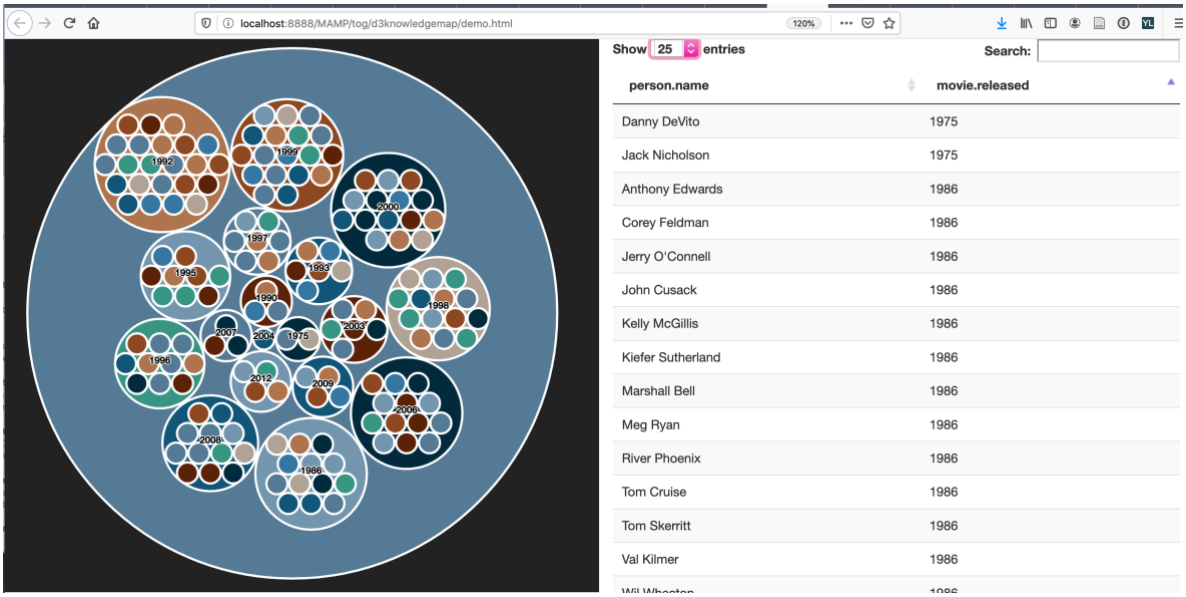


Figura 38 Gráfico *Knowledge Map.*

La figura 40, muestra el valor interior de una de las burbujas de este gráfico, demostrando aquí un nivel de interacción de la gráfica.

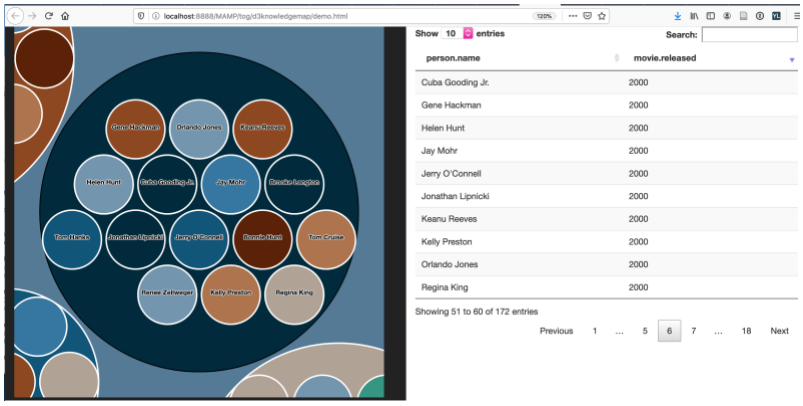


Figura 39 Contenido de la burbuja con valor 2000, del Modelo *Knowledge Map*

4.3.2.2. Modelo *Zoomable Sunburst*

Esta visualización radial de relleno de espacio creada por John Stasko muestra los valores acumulativos de los subárboles. Se usa comúnmente para visualizar paquetes de software y los sistemas de archivos de forma anidada El modelo resplandor solar, *Zoomable Sunburst*, es similar a un mapa de árbol, excepto que utiliza un diseño radial. El nodo raíz del árbol está en el centro, con hojas en la circunferencia. El área de cada arco corresponde a su valor. La figura 41 representa la agrupación en este modelo de gráfico, acompañado de la tabla referencial *Table Graph* que resulta de utilidad para el análisis de los datos.

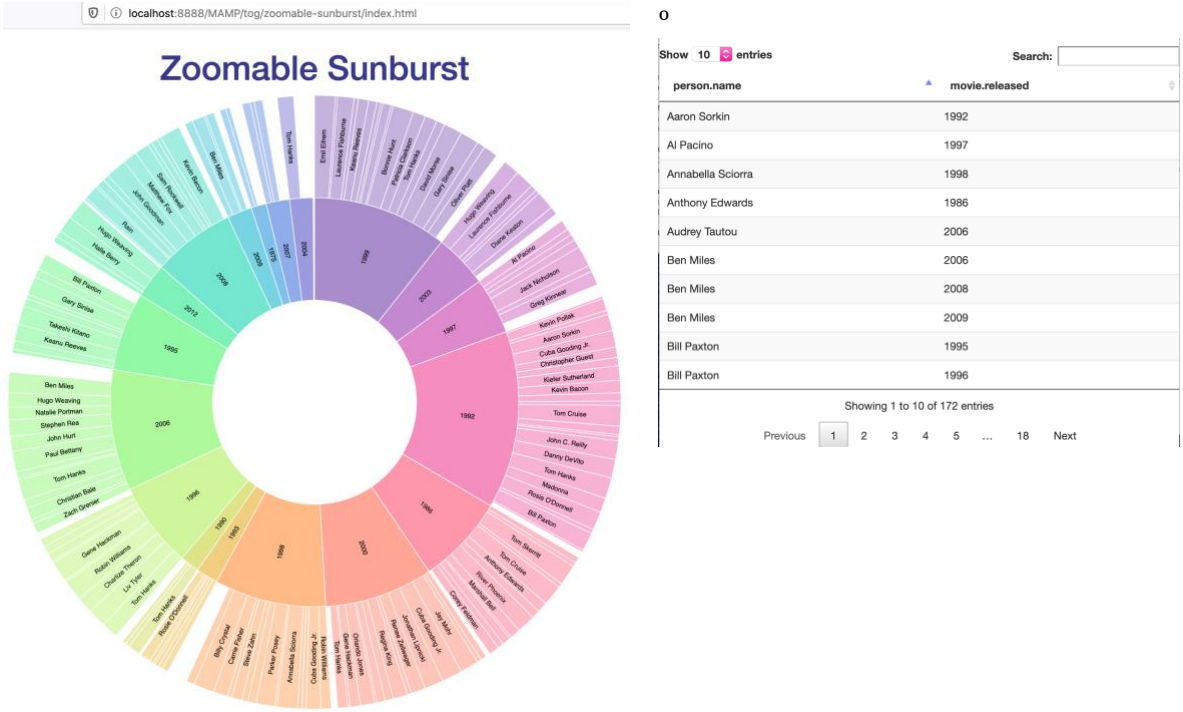


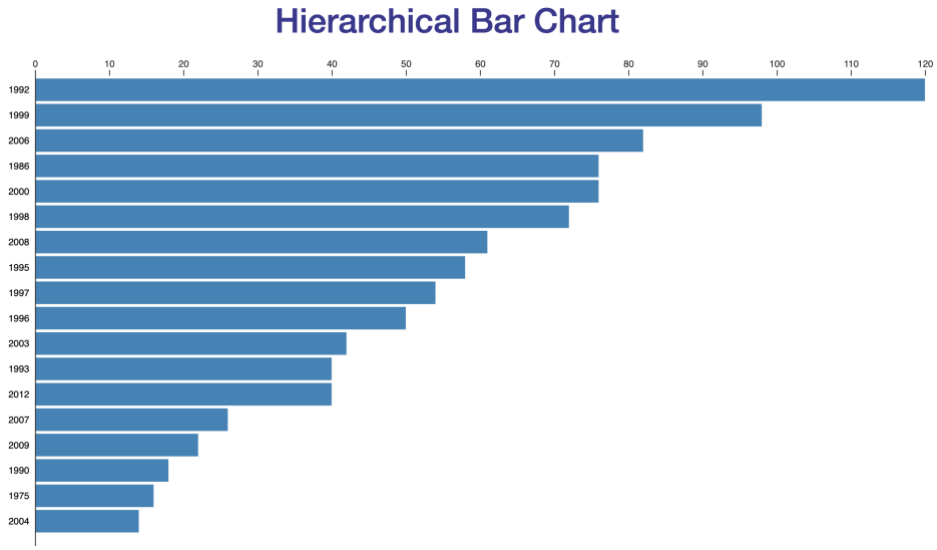
Figura 40 Gráfico *Zoomable Sunburst.*

La figura 42, muestra el valor interior de uno de los niveles de este gráfico, demostrando aquí un nivel de interacción del gráfico *Zoomable Sunburst*, donde vemos que el nivel 2000, contiene el nombre de los 15 actores que participaron en películas en dicho año.

Figura 41 Nivel interior del Gráfico *Zoomable Sunburst.*

4.3.2.3. Modelo *Hierarchical Bar Chart*

Un gráfico de barras es una forma de resumir un conjunto de datos categóricos. El gráfico de barras muestra datos utilizando varias barras, cada una representando una categoría particular. El largo de cada barra es proporcional a una agregación específica (por ejemplo, la suma de los valores en la categoría que representa). Las categorías podrían ser algo así como un grupo de edad o una ubicación geográfica, así como también permite ver la contribución de diferentes categorías a cada barra o grupo de barras en el gráfico de barras. La figura 43 representa la agrupación en este modelo de gráfica, observe que la parte inferior del modelo agrega una tabla referencial, *Table Graph.*



Show 10 entries Search:

person.name	movie.released
Danny DeVito	1975
Jack Nicholson	1975
Anthony Edwards	1986
Corey Feldman	1986
Jerry O'Connell	1986
John Cusack	1986
Kelly McGillis	1986
Kiefer Sutherland	1986
Marshall Bell	1986
Meg Ryan	1986

Showing 1 to 10 of 172 entries Previous 1 2 3 4 5 … 18 Next

Figura 42 Gráfico *Hierarchical Bar Chart.*

La figura 44, muestra el valor interior de uno de los niveles de este gráfico, demostrando aquí un nivel de interacción del gráfico *Hierarchical Bar Chart*, donde vemos que el nivel 2000, contiene el nombre de los 15 actores que participaron en películas en dicho año.

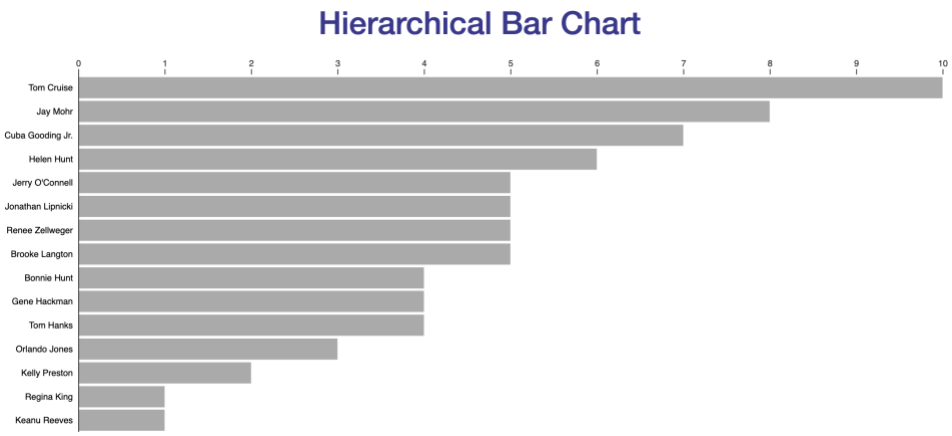
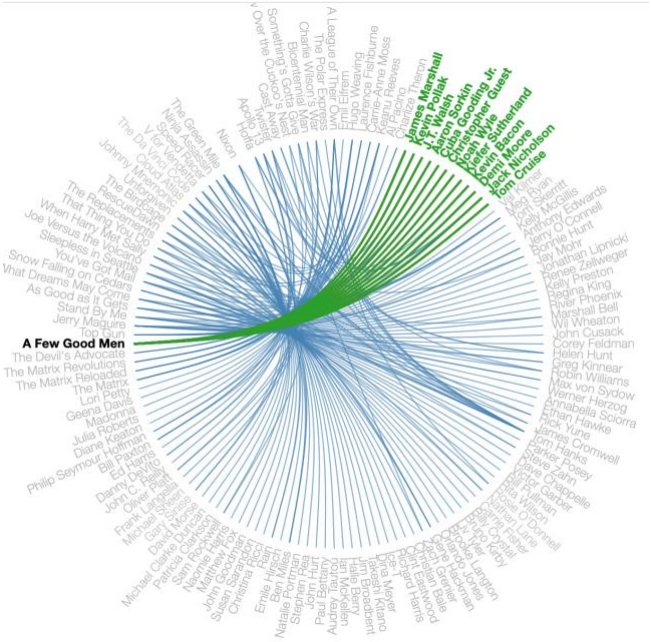


Figura 43 *Hierarchical Bar Chart*, nivel interior del valor 2000 de la gráfica 43.

4.3.2.4. Modelo *Hierarchical Edge Bundle*

Permite visualizar relaciones de adyacencia entre entidades organizadas en una jerarquía. Los elementos de su biblioteca se organizan en varias carpetas, como consulta, datos, escala, etc. Cada carpeta se subdivide en subcarpetas y así sucesivamente. La agrupación jerárquica de bordes utiliza una curva que sigue el enlace de jerarquía entre los 2 elementos. La figura 45, de modo similar a los modelos anteriores, observe que debajo de la gráfica, se despliega una tabla referencial llamada *Table Graph.*



Show 10 entries Search:

person.name	movie.title
Aaron Sorkin	A Few Good Men
Al Pacino	The Devil's Advocate
Annabella Sciorra	What Dreams May Come
Anthony Edwards	Top Gun
Audrey Tautou	The Da Vinci Code
Ben Miles	V for Vendetta
Ben Miles	Speed Racer
Ben Miles	Ninja Assassin
Bill Paxton	Apollo 13
Bill Paxton	Twister

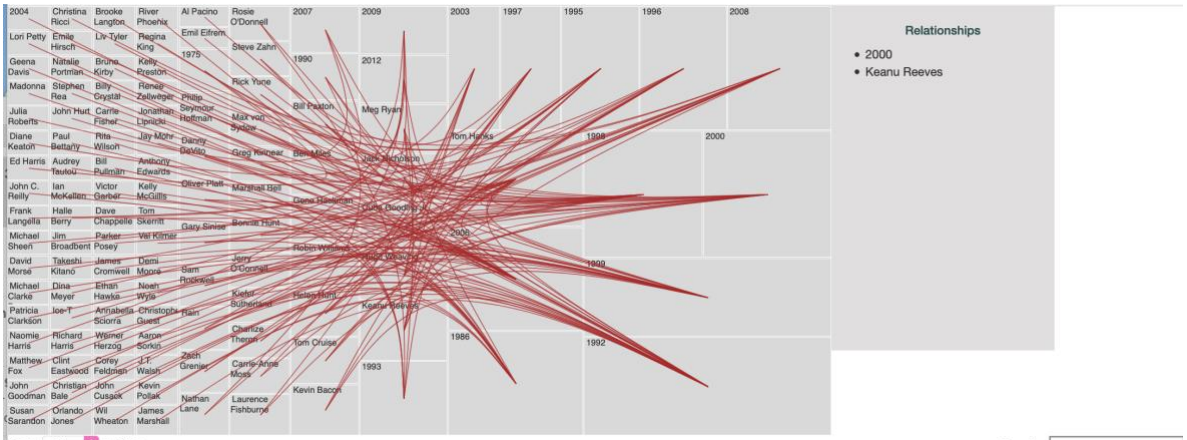
Showing 1 to 10 of 172 entries

Previous 1 2 3 4 5 … 18 Next

Figura 44 Gráfico *Hierarchical Edge Bundle*

4.3.2.5. Modelo *Hierarchical Edge Tree*

Es una agrupación similar a la del gráfico *Hierarchical Edge Bundle*, solo que, en lugar de tener las relaciones en liana de derecha a izquierda, están toda representadas en rectángulos, es una implementación del algoritmo jerárquico de agrupación de bordes de *Danny Holten* en *D3*, y muestra las dependencias entre clases en una jerarquía de clases de software. Las dependencias se agrupan de acuerdo con los paquetes principales. Este ejemplo utiliza dos diseños: un *d3.layout.treemap* radial para colocar los nodos del árbol y *d3.layout.bundle* para agrupar las dependencias en paquetes *spline*. La figura 46 representa la agrupación en este modelo de gráfica.



Show 10 entries Search:

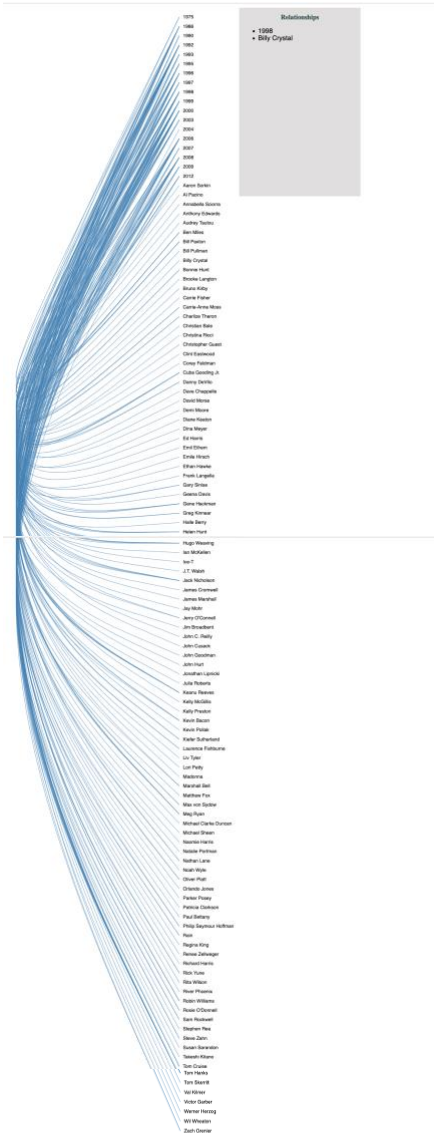
person.name	movie.released
Aaron Sorkin	1992
Al Pacino	1997
Annabella Sciorra	1998
Anthony Edwards	1986
Audrey Tautou	2006
Ben Miles	2006
Ben Miles	2008
Ben Miles	2009
Bill Paxton	1995
Bill Paxton	1996

Showing 1 to 10 of 172 entries Previous 1 2 3 4 5 … 18 Next

Figura 45 Gráfico *Hierarchical Edge Tree.*

4.3.2.6. Modelo *Hierarchical Edge Vertical*

Un diseño plano de agrupamiento jerárquico de bordes, en oposición a la versión radial. La figura 47 representa la agrupación en este modelo de gráfica, acompañado de una *Table Graph* con los datos explícitos contenidos en el modelo de visualización.



Show 10 entries Search:

person.name	movie.released
Aaron Sorkin	1992
Al Pacino	1997
Annabella Sciorra	1998
Anthony Edwards	1986
Audrey Tautou	2006
Ben Miles	2006
Ben Miles	2008
Ben Miles	2009
Bill Paxton	1995
Bill Paxton	1996

Showing 1 to 10 of 172 entries

Previous 1 2 3 4 5 … 18 Next

Figura 46 Gráfico *Hierarchical Edge Vertical.*

4.3.2.7. Modelo *Table Graph*

Este modelo permite visualizar las columnas que el usuario desee, ordenar por todas las columnas y hacer búsqueda de datos a través de la tabla. La figura 48 muestra un ejemplo de dos atributos del grafo.

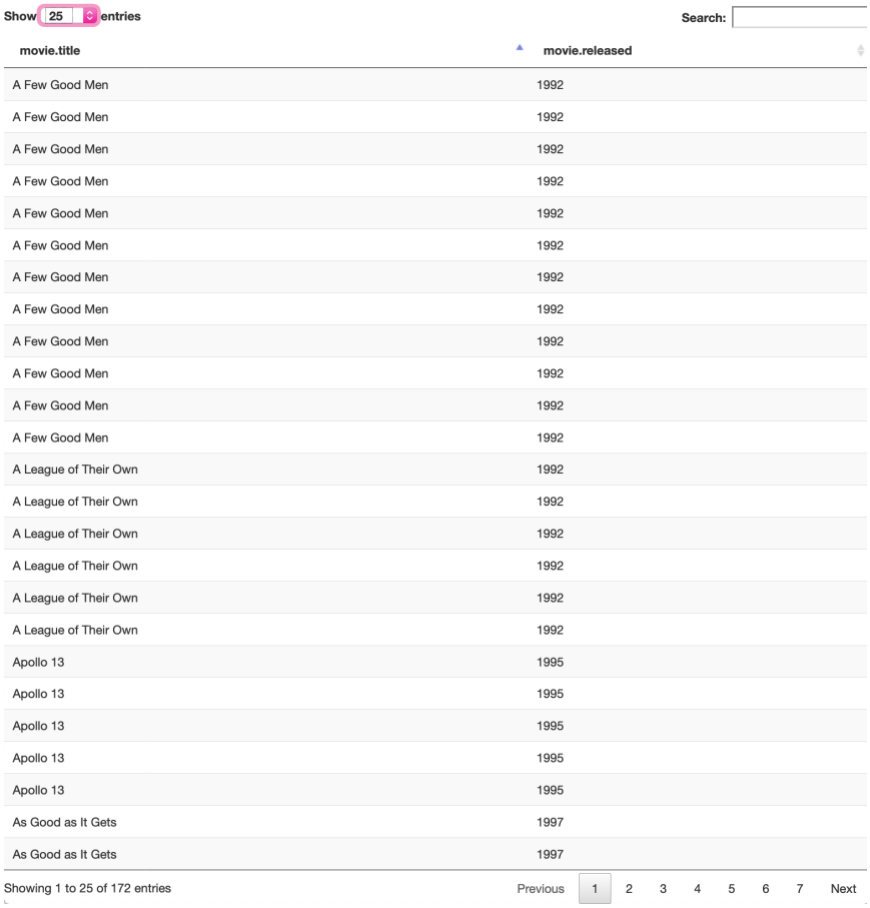


Figura 47 Gráfico *Table Graph*.

Ahora que conocemos a detalle la arquitectura y interfaz del usuario, en el siguiente capítulo revisaremos un caso practico de aplicación de esta solución.

5. RESULTADOS

En esta sección analizaremos los resultados de este trabajo de obtención de grado por medio del estudio de un caso de uso donde aplicaremos todos los algoritmos de grafos y todas las visualizaciones disponibles a través de seis distintos escenarios aplicados.

5.1. Resultados

Este proyecto ha logrado materializar el concepto de manejar, administrar y visualizar una base de datos de grafos desde una aplicación *Web* amigable, que a su vez nos permite hacer análisis de los datos de grafos a través de la ejecución de poderosos algoritmos de base de datos provistos por la librería de *NEO4J*, así como también, nos provee de siete modelos de visualización para su análisis.

Permite extraer datos libremente de la base de datos de grafos, así como hacer modificaciones al grafo cuando el usuario lo necesite, como, por ejemplo, al ejecutar los algoritmos de base de datos, con los parámetros como: nodos, relaciones, direcciones y limites que el usuario elija. Este proyecto esta principalmente enfocados en la ejecución de algoritmos de centralidad y detección de comunidades.

Dentro de esta sección, realizaremos el análisis de un caso de uso donde exploraremos un grafo, agregaremos la ejecución de cada uno de los algoritmos disponibles, así como mostrar su aplicación en los modelos de visualización disponibles.

Exploraremos la base de datos: *Estudiantes.db*, que contiene información general de estudiantes, su relación con otros alumnos, el lugar de origen, género, el talento en al que se destaca y el nombre de su padre o tutor. En la figura 49 se puede ver el esquema de la base de datos en *NEO4J*.

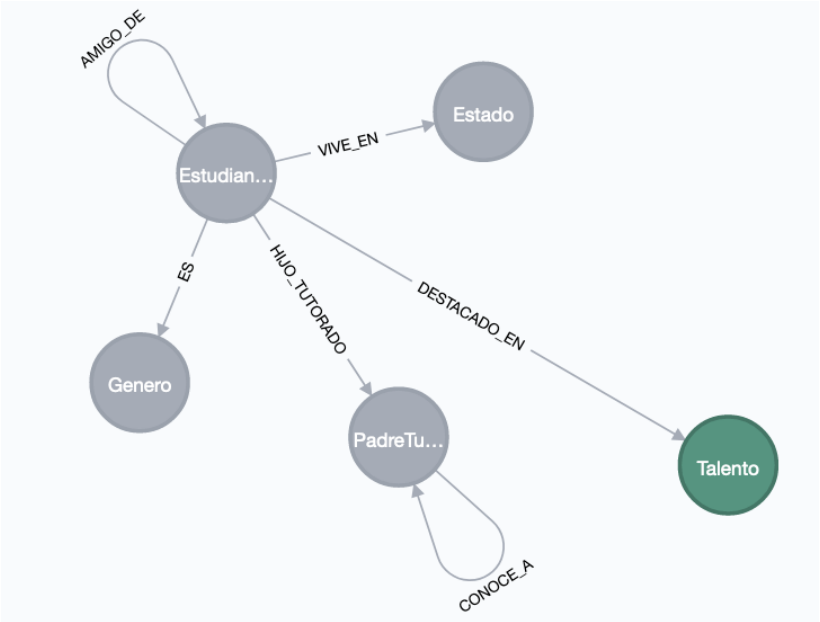


Figura 48 Esquema de base de datos visto desde NEO4J. El nodo en color verde contiene el talento en el que los estudiantes son desatacados.

Para fines de demostración, analizaremos a relación *AMIGO_DE* entre *Estudiantes*, y exploraremos si esta relación nos muestra conexiones fuertes de centralidad y detección de comunidades. En la figura 50 que muestran y las relaciones, donde a simple vista, sería complejo determinar dichas conexiones. Regularmente el análisis de grafos será aplicado a grafos de alto volumen y complejidad, sin embargo, para este documento se eligió un grafo de tamaño pequeño para hacer más clara la explicación y su aplicabilidad.

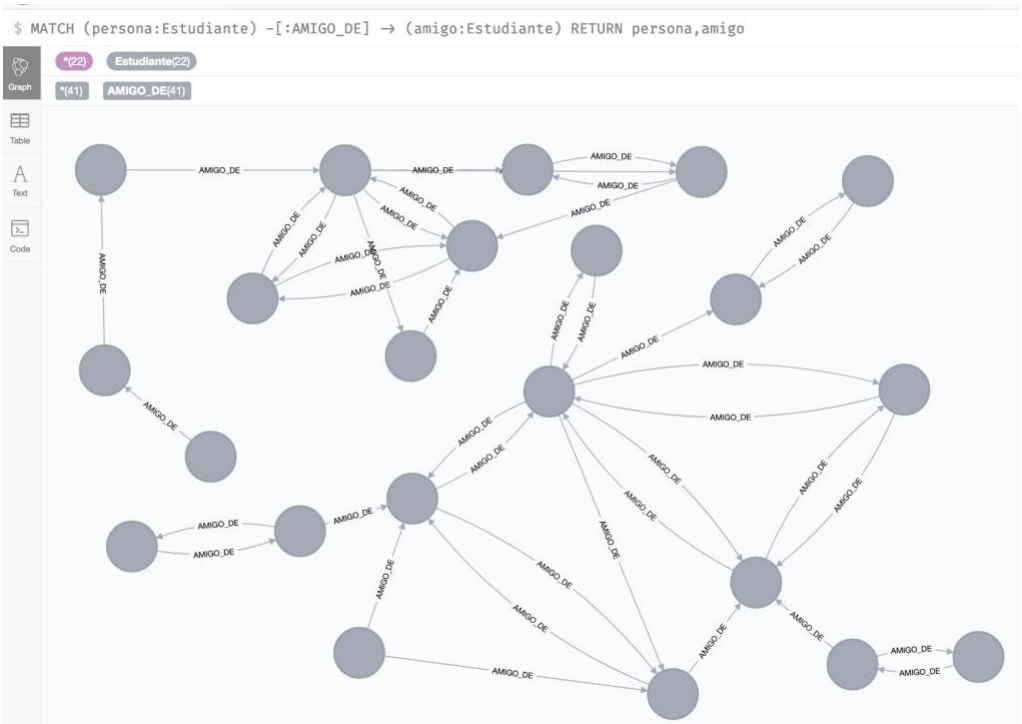


Figura 49 Grafo Estudiantes desde NEO4J. Se analizan los nodos Estudiante, y la relación AMIGO_DE.

Ahora, realicemos dicho análisis desde el proyecto que esta investigación genera, La figura 51 nos muestra los nodos y atributos contenidos en la base de datos *Estudiantes.db*, a su vez la figura 52 despliega las relaciones que existen entre nodos y los atributos de estas en caso de existir.

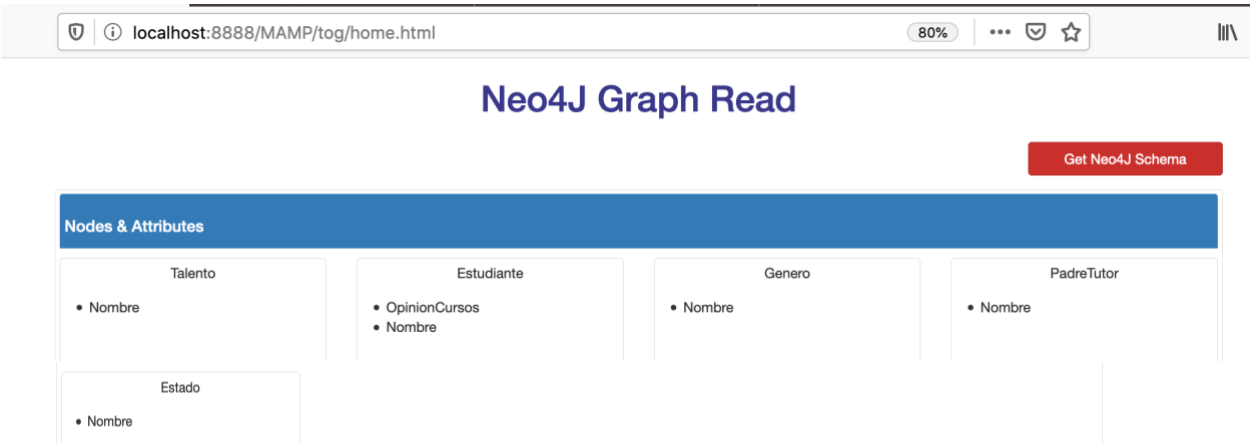


Figura 50 Nodos y atributos de los nodos contenidos en la base de datos *Estudiantes.db*.

Figura 51 Relaciones y atributos de estas contenidas en la base de datos *Estudiantes.db*.

Una de las secciones más importantes de este trabajo se refiere a la ejecución de algoritmos de grafos, en la figura 53, se muestra el código en *CYPHER* para aplicar todos los algoritmos disponibles a la base de datos. Nótese, que son cajas de texto editables donde el usuario puede ajustar sentencias sugeridas usando el lenguaje de consultas *CYPHER* y apegado a la sintaxis propia de cada algoritmo de *NEO4J*.

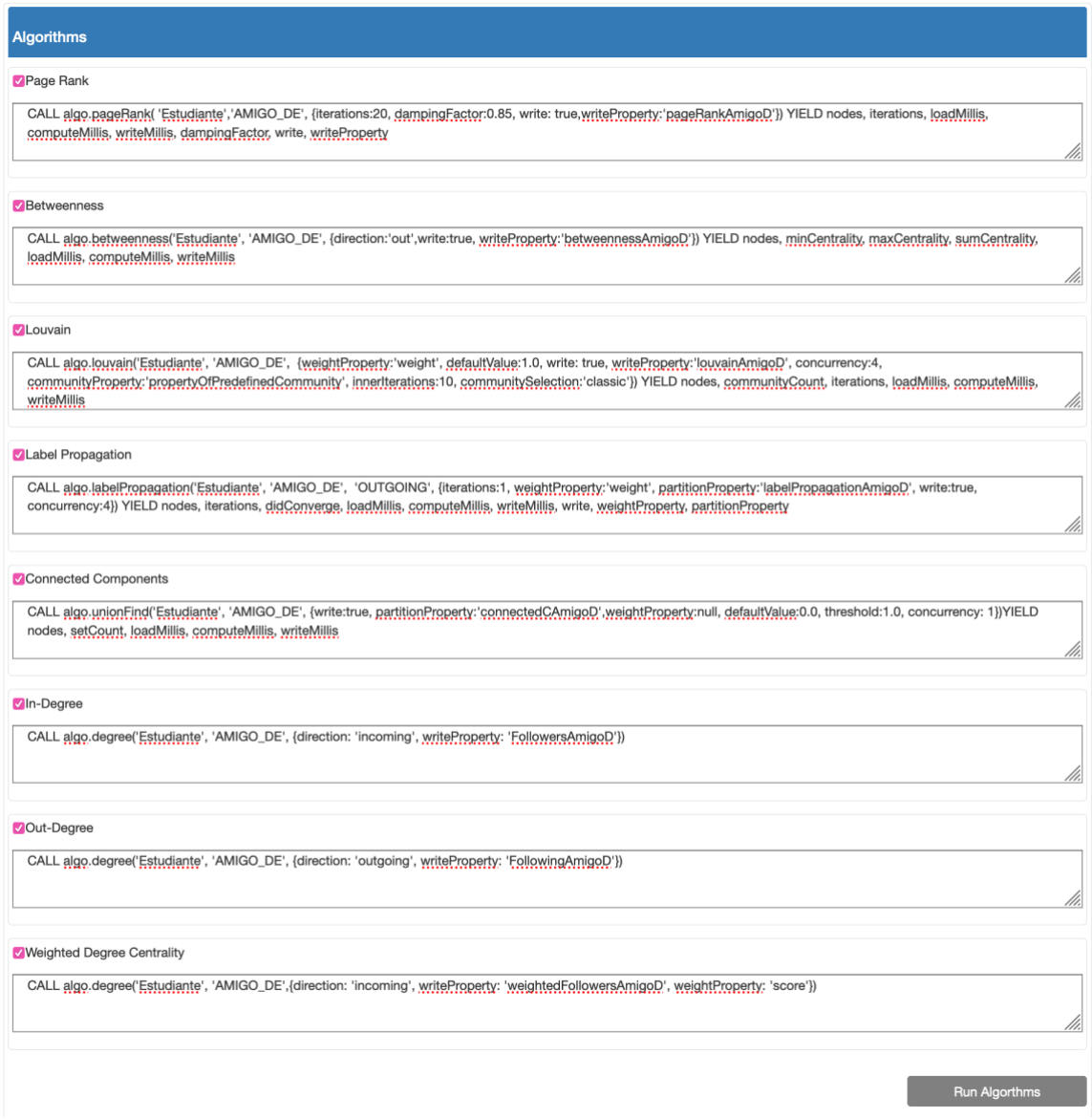


Figura 52 Algoritmos disponibles para ejecución, con sugerencia de construcción con base a las relaciones y atributos de la base de datos *Estudiantes.db*.

Además de lo anterior, el usuario puede hacer una extracción de datos a su libre conveniencia. En la figura 54, se observa la sección donde el usuario puede escribir la consulta en *CYPHER* y extraer cualquier cantidad deseada de nodos y relaciones para analizar en la herramienta visual.

Figura 53 - Sección de libre ejecución de consultas a la base de datos de grafos

Otra importante habilidad de esta solución es la evaluación sentimientos de atributos, donde estos pueden ser catalogados como *Positivo, Negativo y Neutro* según la *API* de *Google Cloud* que es consumida desde el *middleware*, y genera un nuevo atributo con el valor del sentimiento correspondiente, el cual puede ser usado por el usuario en la sección de ejecución de consultas (*User Input Query).* La figura 55, muestra esta sección, el usuario deberá seleccionar los atributos que desee analizar.

Figura 54 Sección que permite la ejecución de Análisis de Sentimientos para uno o más atributos de la base de dato de grafos.

5.2. Caso de Uso

Ahora, ejecutamos todos los algoritmos de base de datos disponibles en la aplicación, e inmediatamente despues podemos ver que se generó una nueva propiedad al nodo Estudiante, estos nuevos atributos se identifican porque preceden el nombre del algoritmo tal como se muestra en la figura 56.

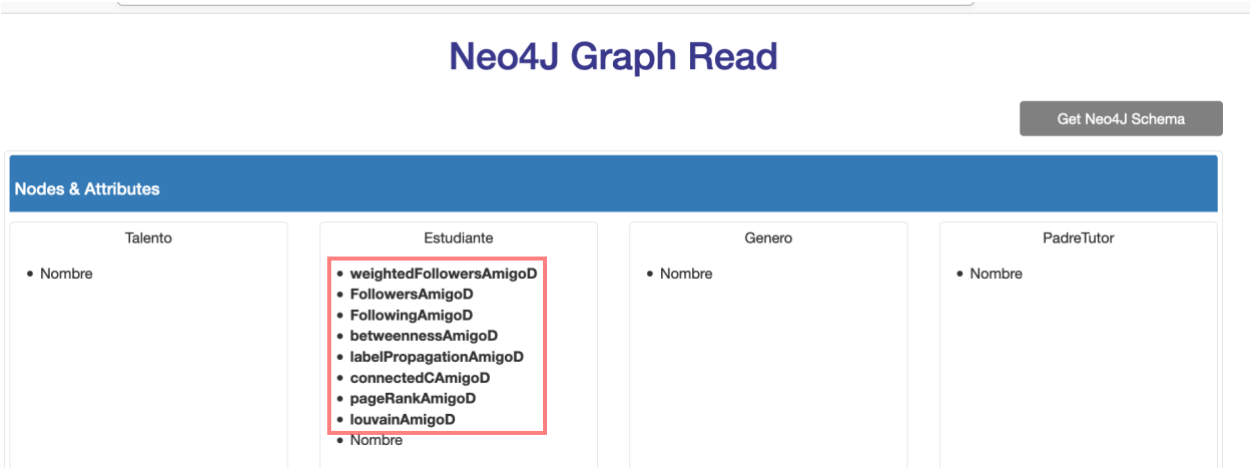


Figura 55 En texto resaltado se observan los atributos recientemente añadidos, *ver texto dentro de marco rojo como referencia*.

Vayamos al análisis de esta información, usaremos la consulta en CYPHER

```
MATCH (persona:Estudiante) -[:AMIGO_DE]-> (amigo:Estudiante)
RETURN persona, amigo
```

Con esa consulta analizaremos las relaciones y su contexto. Usamos la sección *User Input Query*, que se muestra en la figura 54, para ejecutar esta consulta, lo que inmediatamente nos redirige a la página de análisis, donde podemos analizar el grafo usando los atributos agregados por la ejecución de los algoritmos de base de datos.

Ahora planteamos posibles escenarios de análisis.

5.2.1. Escenario 1, Algoritmo de Centralidad, *PageRank*

Supongamos que el usuario final, está haciendo un análisis del grafo en *NEO4J*, y busca conocer cuál o cuales son los Estudiantes con mayor influencia, aquellos que están más fuertemente conectados con otros Estudiantes. Para esto decide explorarlo a través de la propiedad generada por el algoritmo de centralidad *PageRank*, en la figura 57 vemos la selección de atributos.

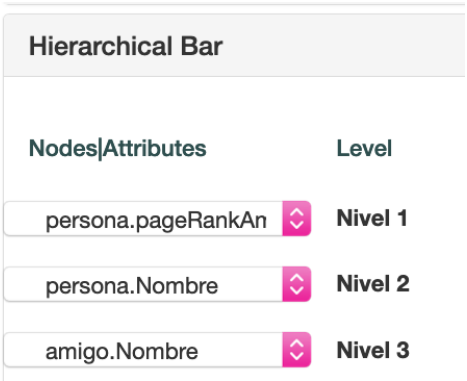


Figura 56 Selección de Atributos en el Modelo *Hierarchical Bar*.

En la figura 58 el modelo de visualización muestra de forma ordenada que el Estudiante: *Jesús*, es el más influyente, ya que su valor del atributo *PageRank* es el más alto en el grafo. Observe también, que cada modelo viene acompañado con una tabla que muestra los atributos seleccionados y permite ordenación y búsqueda, esto para complementar la información proporcionada por el modelo de visualización principal.

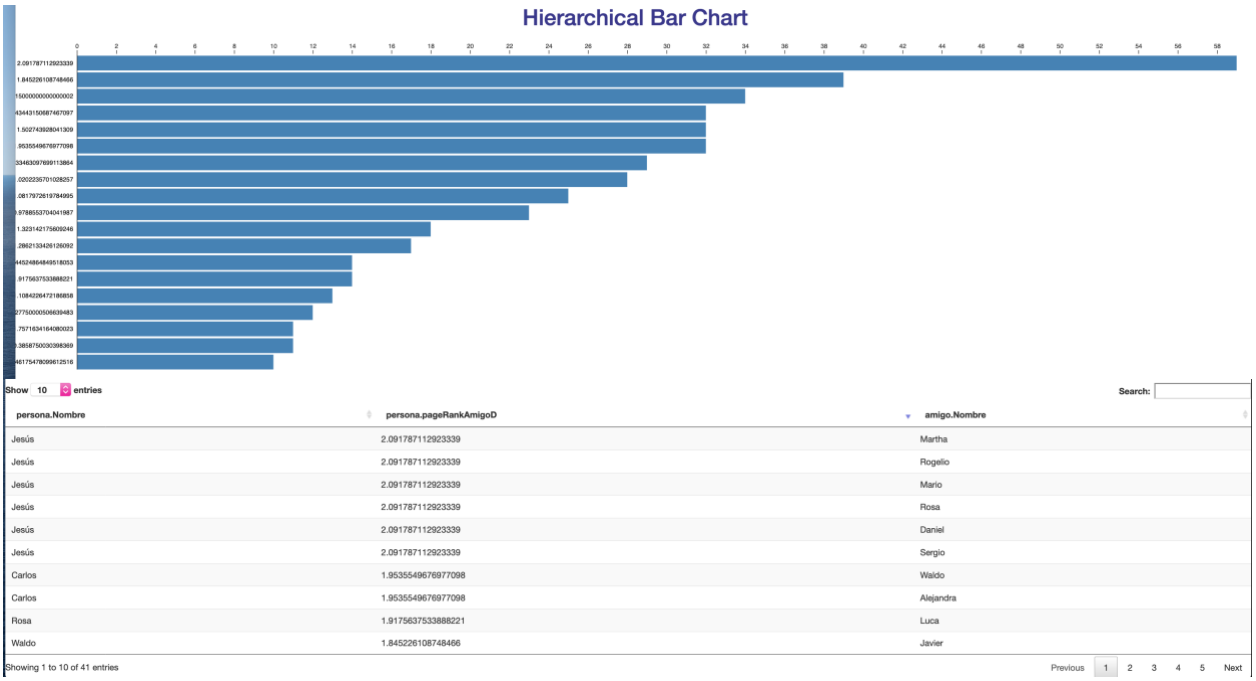


Figura 57 Modelos *Hierarchical Bar* y *Table Graph*.

5.2.2. Escenario 2, Algoritmo de Centralidad, *Betweeness*

El analista de este grafo ahora quiere saber qué estudiantes tienen más conexiones con otros estudiantes, necesita identificar a las personas que son un puente con otros grupos de personas, para esto, explotará el algoritmo de centralidad *Betweeness*. *L*a figura 59 muestra la selección de atributos en el modelo.

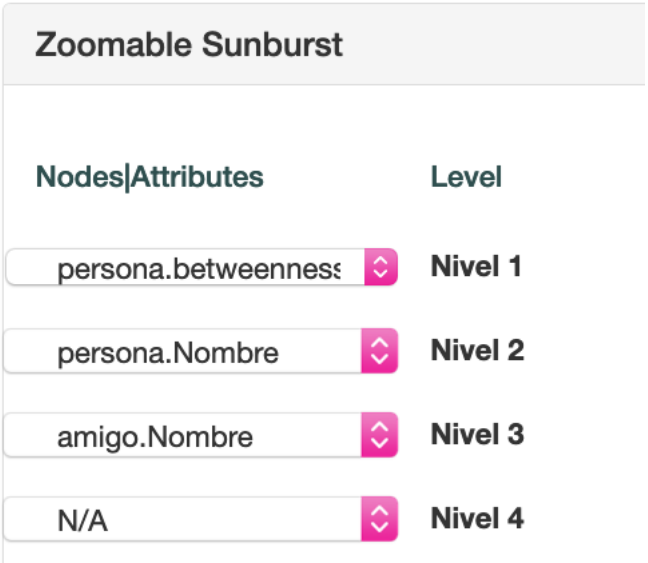


Figura 58 Selección de atributos en el Modelo *Zoomable Sunburst*.

En la figura 60 observamos que valores más altos en el atributo *betweennessAmigoD*, son *46, 25 y 20.5* que corresponden a *Jesús, Waldo y Rogelio.* La figura 60 es un subgrafo del mismo modelo y nos provee de información más detallada, tal como podemos ver en la figura 61.

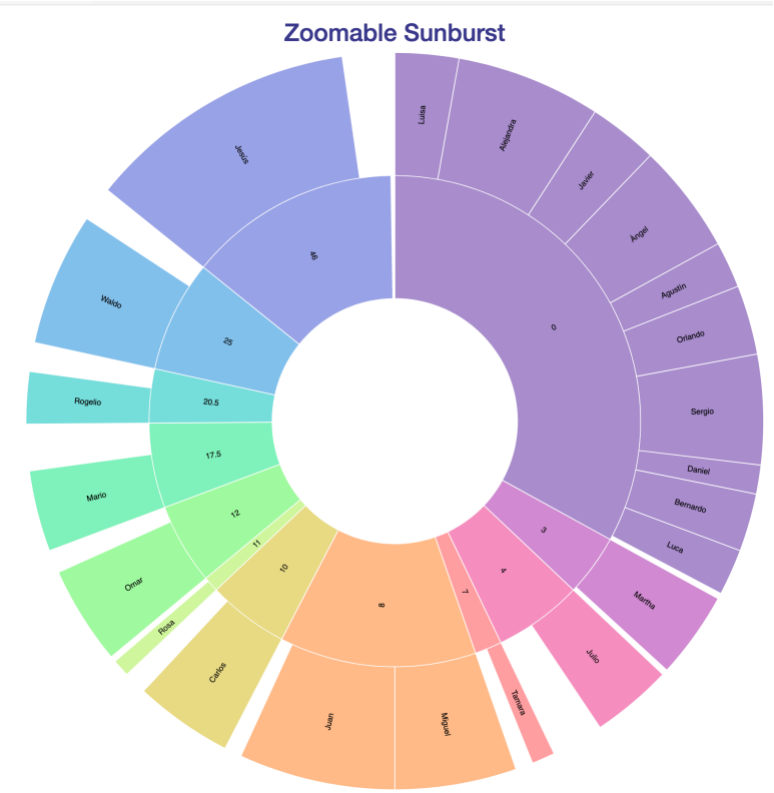


Figura 59 Modelo *Zoomable Sunburst* mostrando el atributo *Betweeness* recién agregado.

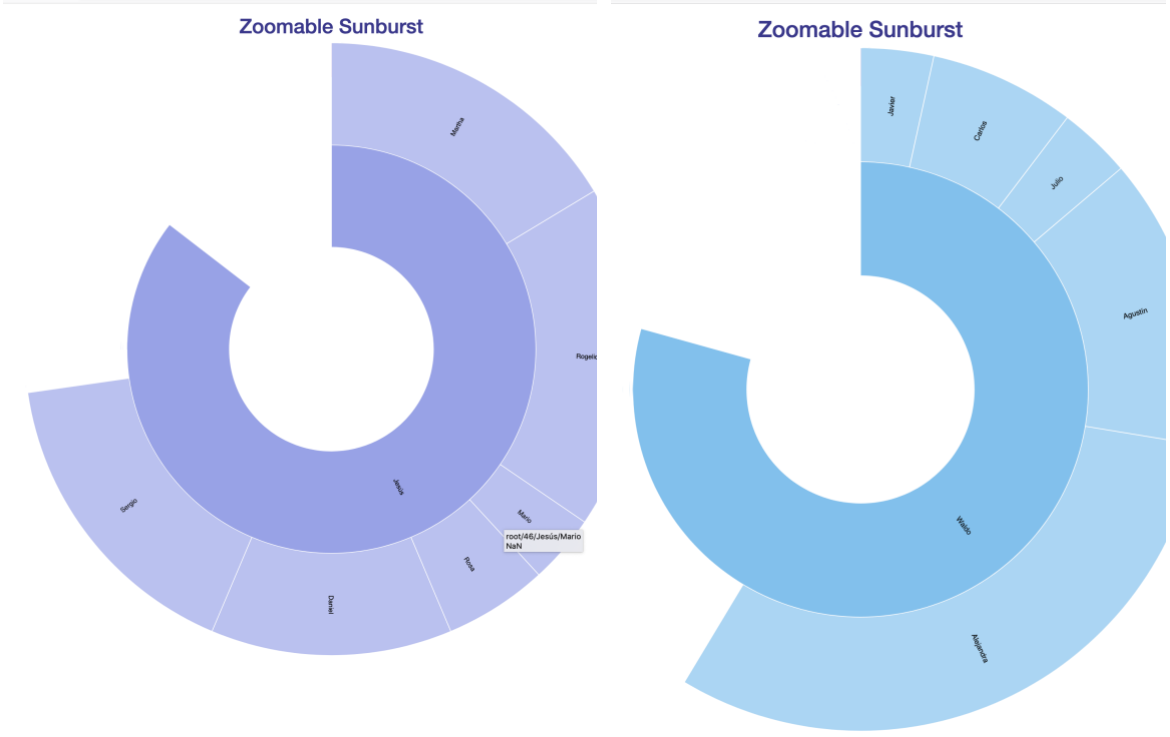


Figura 60 Modelo *Zoomable Sunburst*, un nivel adentro de los dos valores más altos, *Jesús y Waldo* para así observar los estudiantes con quienes están conectados.

5.2.3. Escenario 3: Detección de Comunidades, *Louvain*

Ahora, queremos detectar las comunidades de estudiantes que estén fuertemente conectadas entre sí, pero poco conectadas con otras comunidades, para eso, podemos usar la propiedad generada por el algoritmo de detección de comunidades *Louvain*, ya que hace un de agrupamiento jerárquico para es crear comunidades donde la densidad de la relación es alta, mientras que la densidad intercomunitaria sigue siendo baja.

La figura 62 muestra la selección de atributos, donde *louvainAmigoD* es el atributo guía en este propósito, y usamos el Modelo *Knowledge Map Graphic.*

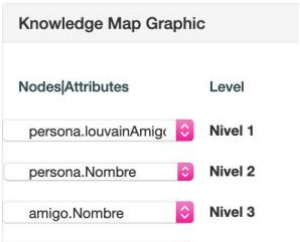
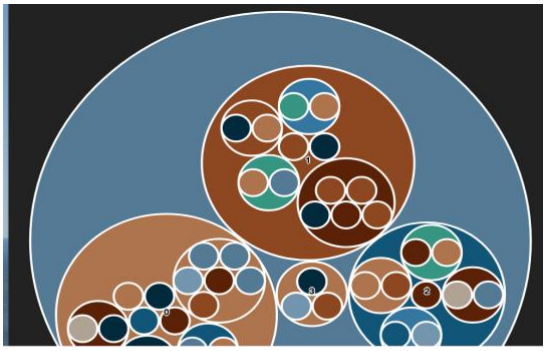


Figura 61 Selección de Atributos en el Modelo *Knowledge Map Graphic*.

La figura 63 muestra un total de 4 comunidades, donde la relación entre ellas es fuerte, ya que es un algoritmo jerárquico, nos genera comunidades con identificadores del *0* a *n*, donde *0* representa a la comunidad más fuerte y grande.



Show 10 entries Search:

persona.Nombre	persona.louvainAmigoD	amigo.Nombre
Omar	3	Waldo
Orlando	3	Tamara
Tamara	3	Omar
Bernardo	2	Miguel
Martha	2	Rogelio
Martha	2	Mario
Miguel	2	Rogelio
Miguel	2	Bernardo
Rogelio	2	Martha
Rogelio	2	Jesús

Showing 1 to 10 of 41 entries Previous 1 2 3 4 5 Next

Figura 62 Modelos *Knowledge Map Graphic y Table Graph*.

Si expandimos las comunidades 0 y 1, podemos ver los estudiantes que se muestran en cada una. La figura 64 muestra a la izquierda los integrantes de la comunidad 0 y la derecha los integrantes de la 1.

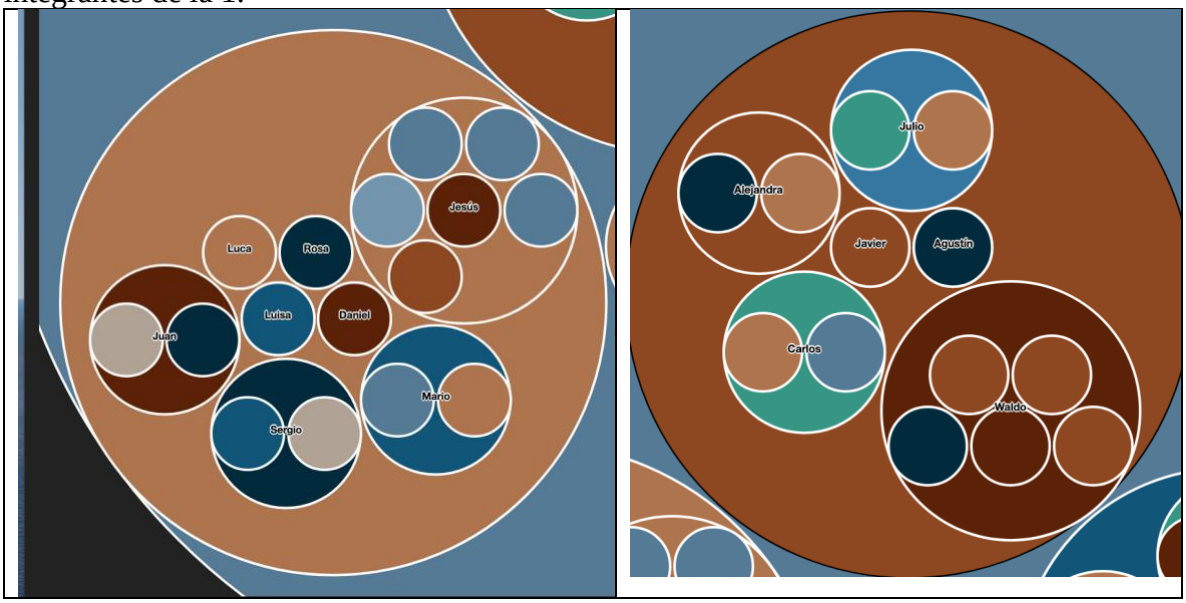
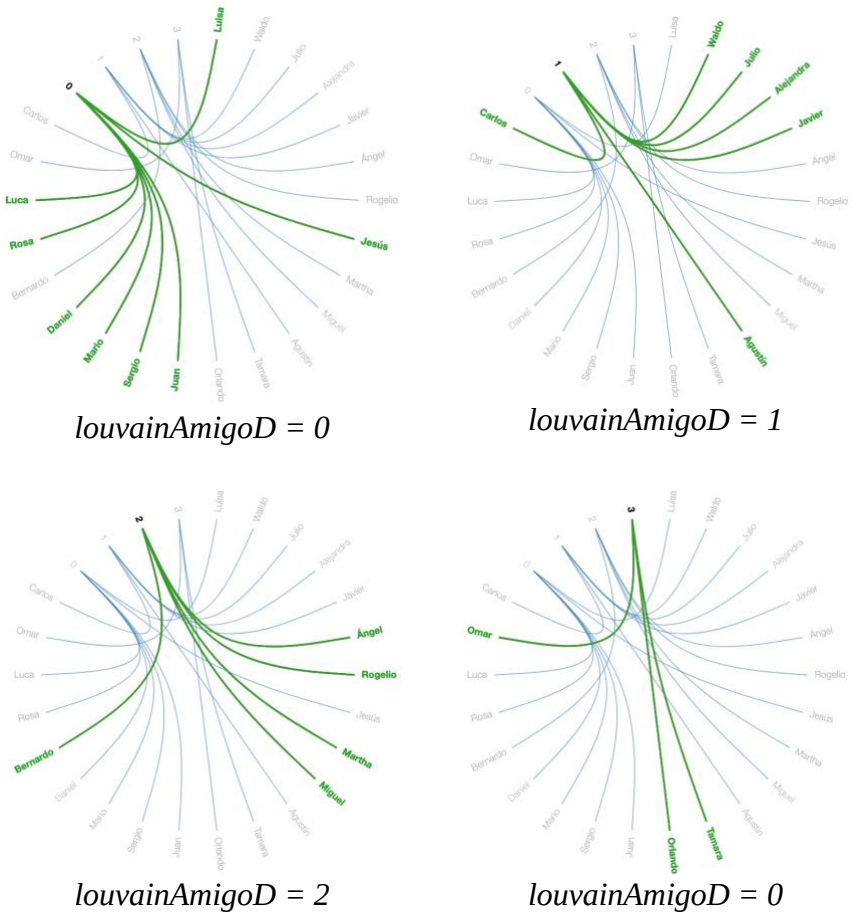


Figura 63 Comunidad *Louvain*, 0 y 1 para el nodo Estudiantes.

Tal como se aprecia en la figura 65, otra manera de visualizar esta relación puede ser, a través del modelo *Hierchical Edge Bundle Graphic*, bajo la misma selección de atributos que el gráfico anterior, solo seleccionando un modelo distinto, donde igualmente podemos ver las relaciones de las comunidades 0 y 1.



louvainAmigoD = 0 *louvainAmigoD = 1*

louvainAmigoD = 2 *louvainAmigoD = 0*

Figura 64 Modelo *Hierchical Edge Bundle* mostrando 4 comunidades generadas por el algoritmo *Louvain*.

5.2.4. Escenario 4: Detección de Comunidades, *Label Propagation.*

Busquemos ahora, detectar comunidades más pequeñas con modularidad menor, para esto, utilizaremos el atributo generado por el algoritmo de detección de comunidades *Label Propagation*, tal como se muestra en la figura 66, a su vez, en la figura 67 observaremos comunidades más cerradas y más fuertemente conectadas comparado con las que detecta *Louvain*

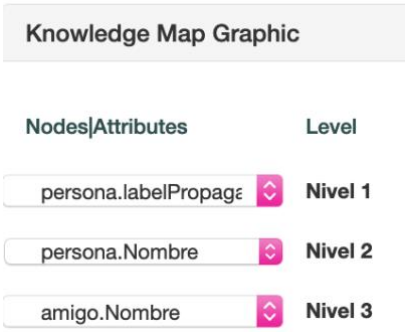


Figura 65 Selección de Atributos, *labelPropagationAmigoD* en el Modelo *Knowledge Map Graphic*.

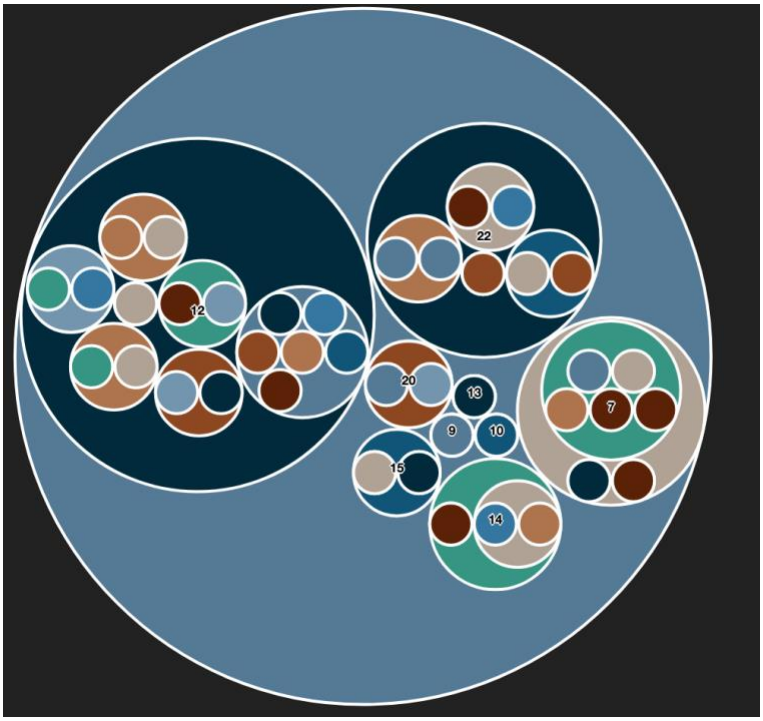


Figura 66 Modelo *Knowledge Map Graphic* mostrandos 9 comunidades generadas por el algoritmo *Label Propagation.*

5.2.5. Escenario 5: Detección de Comunidades, *Weakly Connected Components*

Queremos encontrar en este grafo aquellas comunidades que estén débilmente conectadas, identificar los nodos que sean potencialmente inalcanzables por otros nodos o bien que el alcanzarlos sea complejo, para este propósito, sirve el algoritmo de detección de comunidades *Connected Componentes.* La figura 68, refiere la selección de atributos para desplegar el modelo de visualización.

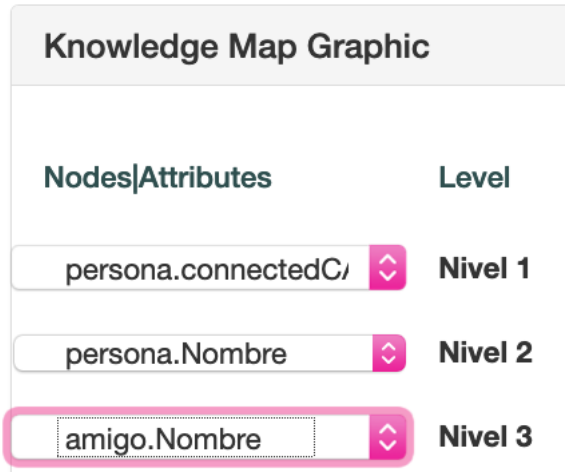


Figura 67 Selección de atributos, *connectedCAmigoD* en Modelo *Knowledge Map Graphic*.

En la figura 69, observamos que los nodos más débilmente conectados son aquellos que están con sin contenido en el modelo *Knowledge Map,* y así en orden hasta los más grandes. En este ejemplo, las comunidades *1,6,8,9,10,11,1516,17 y 21* son las más débiles.

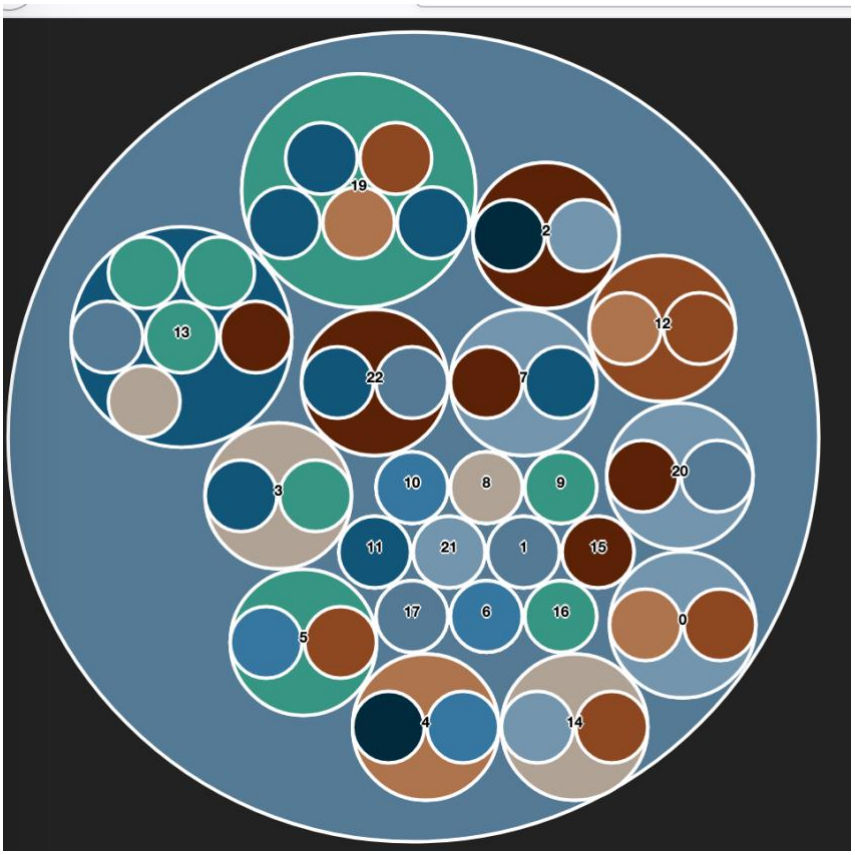


Figura 68 Modelo *Knowledge Map Graphic* muestra 22 comunidades generadas por el algoritmo *Connected Components*.

En la figura 70 podemos corroborar que esas comunidades son simples, de dos elementos solamente, y, por lo tanto, están débilmente conectadas.

persona.Nombre	persona.connectedCAmigoD	amigo.Nombre
Juan	0	Mario
Juan	0	Luisa
Javier	1	Carlos
Bernardo	6	Miguel
Agustín	8	Julio
Omar	9	Waldo
Tamara	10	Omar
Orlando	11	Tamara
Luca	15	Rosa
Rosa	16	Luca
Daniel	17	Jesús

Figura 69 Modelo *Table Graph*, que representa una tabla anexa a *Modelo Knowledge Map Graphic*.

Hasta aquí hemos revisado los cinco algoritmos de base de datos disponibles y aplicados a la base de datos de estudiantes. Es importante remarcar que también están disponible los algoritmos de centralidad de *Degre*, *In-degree y OutDegree* que nos ayudan a analizar la conectividad de los nodos para identificar cuáles son más influyentes.

Usando *Degre*, *In-dregree* y *Out-Degree* podemos identificar a cuantos nodos está siguiendo un X nodo, por lo tanto, podemos saber que tan relacionado esta y podría influir dependiendo del contexto de análisis, por ejemplo, en una red de interconexión de computadoras.

5.2.6. Escenario 6: Análisis de Sentimiento en Atributos

El usuario, requiere conocer cuál es la opinión generalizada de los estudiantes con respecto a la escuela, para esto, podemos hacer uso de la capacidad de esta solución de evaluar la opinión como *Positiva, Negativa o Neutra* gracias al uso de la *API* de *Google Sentiment.* Dicho lo anterior, el grafo que estamos revisando en esta sección, contiene un atributo llamado *OpinionCursos* que contiene un texto donde cada estudiante ha expresado su retroalimentación con respecto a los cursos de la universidad en donde están inscritos. En la figura 71 el usuario selecciona el atributo a evaluar.

Figura 70 Muestra todos los atributos del grafo y permite seleccionar aquellos a evaluar el *Sentiment*.

Una vez terminado de evaluar, se agrega un atributo nuevo al grafo con nombre del atributo evaluado, más la palabra *Sentiment*, para este ejemplo *OpiniónCursosSentiment* tal como se aprecia en la figura 72.

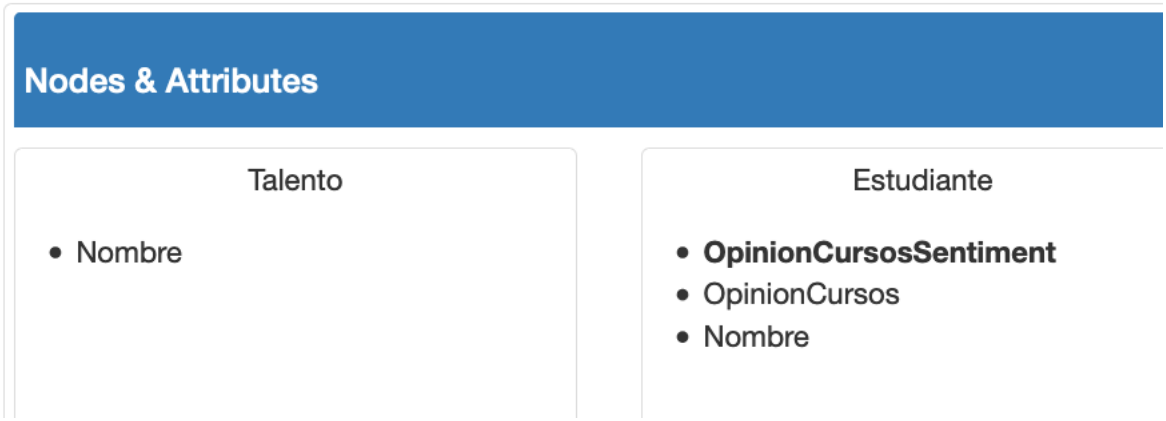
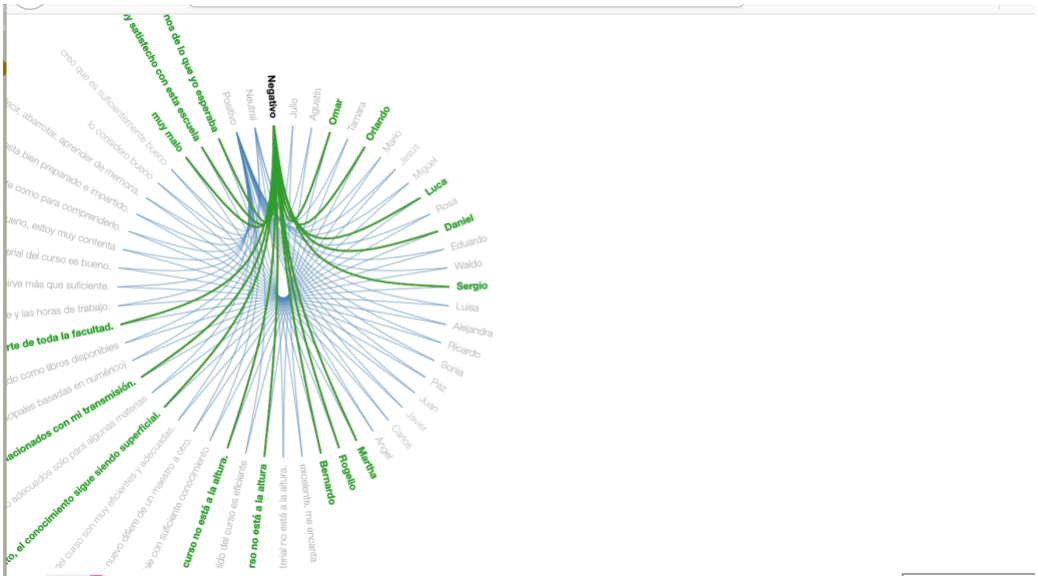


Figura 71 Muestra en texto resaltado el nuevo atributo resultado del análisis de Sentimiento.

Usaremos los modelos de visualización *Hierchical Edge Bundle Graphic*. En la figura 73 *BDG* podemos apreciar los estudiantes que están relacionados con opiniones negativas.



Show 10 entries Search:

persona.Nombre	persona.OpinionCursos	persona.OpinionCursosSentiment
Agustín	El conocimiento y la profundidad del curso son buenos, pero al llegar al material no está a la altura.	Neutral
Alejandra	el contenido es muy eficiente y nos brinda un gran conocimiento de cada tema Según yo, todo va bien de acuerdo con la interacción. Se sirve más que suficiente.	Positivo
Bernardo	es menos de lo que yo esperaba	Negativo
Carlos	lo considero bueno	Positivo
Daniel	algunos cursos son irrelevantes ya que no están relacionados con mi transmisión.	Negativo
Eduardo	Hay muchos cursos que son irrelevantes para la rama. Algunos temas no están cubiertos. No hay material adecuado para ningún curso. promedio (en un término debe haber solo dos materias principales basadas en numérico)	Neutral
Javier	El contenido, por supuesto, está perfectamente en línea con la filosofía de enseñanza que se perpetúa aquí, es decir, abarrotar, aprender de memoria.	Positivo
Jesús	Los materiales o enlaces proporcionados para el estudio son buenos, la claridad en profundidad de los cursos, pero de nuevo difiere de un maestro a otro.	Positivo
Juan	me parece que esta bien preparado e impartido.	Positivo
Julio	excelente, me encanta	Positivo

Showing 1 to 10 of 26 entries Previous 1 2 3 Next

Figura 72 Modelo *Hierchical Edge Bundle Graphic*, resalta en verde la relación del atributo "*Negativo*" con los otros nodos del grafo, y adjunto su correspondiente *Table Graph*.

Podemos observar que *Luca, Sergio, Daniel, Orlando, Martha, Rogelio, Bernando y Omar* tienen una opinión *Negativa*, así como también, podríamos leer los textos en detalle ya sea en la gráfica o en la tabla que se muestran en la figura 73.

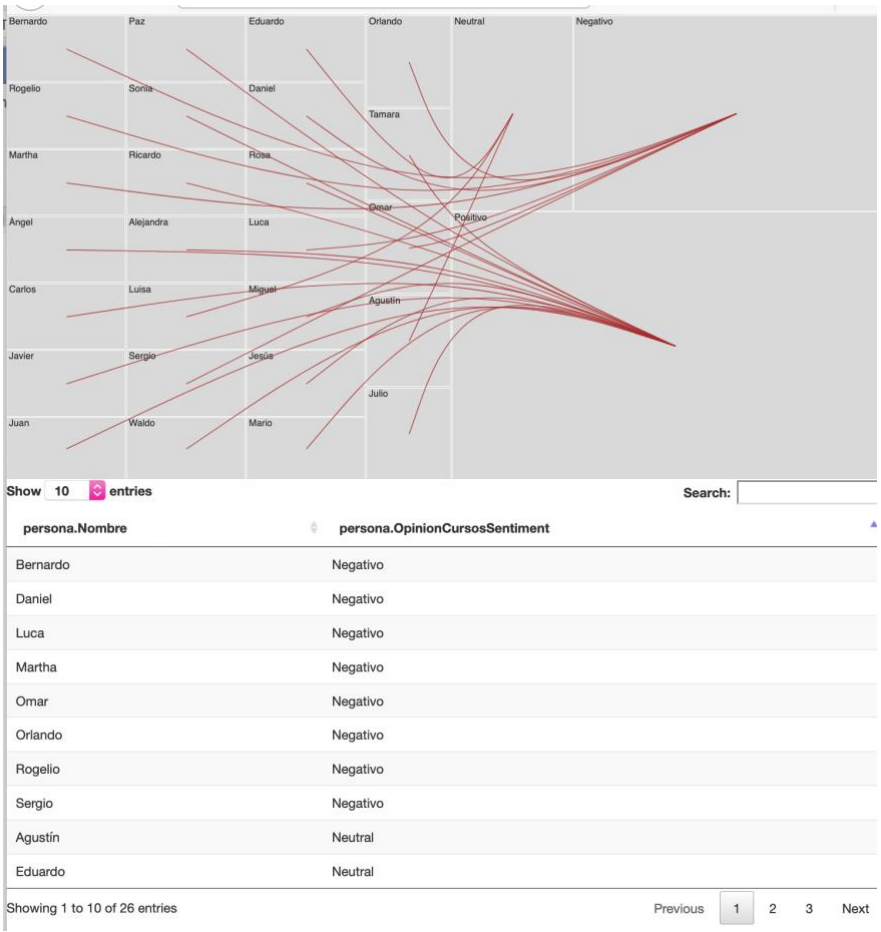


Figura 73 Modelo *Hierchical Edge Tree Graphic*, resalta en rojo las relaciones entre los otros nodos del grafo para los atributos seleccionados, y adjunta su correspondiente *Table Graph*.

En la figura 74, vemos un modelo en forma de rectángulos donde el tamaño de cada uno representa el peso de la relación en base al grafo, por ejemplo, sencillamente podremos ver que el cuadro *Positivo* es el más grande en comparación al resto, por lo tanto, podríamos concluir que la mayor parte de las opiniones son de este tipo.

El analista tiene una aplicación *Web* fácil de usar para conectarse directamente a una base de datos de grafos a través de la ejecución de algoritmos de grafos que dan como resultado atributos recientemente agregados a la *BDG* existente. Estos atributos están correlacionados con las áreas principales de análisis como el cálculo de centralidad y la detección comunitaria. Estos atributos tienen un significado muy significativo para apoyar el análisis de datos[49]

Ahora, el *BDG* tendría atributos significativos para los algoritmos de grafos ejecutados, lo que apoya a una mejor interpretación de los datos de gráficos, lo que ayuda a abordar la solución de problemas en una amplia variedad de áreas, como la reducción de costos en diferentes áreas de actividades comerciales como se explica en [50], de manera similar, también es compatible con el abordaje de problemas completos de polinomios *no NP* (*PN*) conocidos como *k-coloring*, *Hamilton Path*, entre otros [51]. Además, el usuario puede leer

los datos en siete modelos de visualización diferentes, así como una variedad de combinaciones de atributos para su visualización.

Esta solución permite también analizar y comprender las opiniones y comentarios de los usuarios para identificar nuevos requisitos o servicios utilizando una *API* de terceros para el análisis de sentimientos y así, obtener y agregar atributos a las instancias de nodos en función de los datos existentes.

6. CONCLUSIONES

6.1. Conclusiones

NEO4J tiene una capacidad potente para ejecutar un amplio conjunto de algoritmos integrados que son fáciles de implementar, rápidos de ejecutar y proporcionan excelentes resultados [32]. Este proyecto permite ejecutar sin esfuerzo algoritmos de grafos que permiten al usuario desarrollar soluciones inteligentes basadas en los datos consultados, así como conectarse con una *API* de terceros y obtener la evaluación de sentimientos sobre atributos a libre criterio del usuario. También ayuda a los usuarios a interpretar fácilmente los datos extraídos mediante la creación de siete modelos visuales.

Ahora hemos proporcionado una solución altamente innovadora y eficiente para que los científicos de datos realicen análisis de datos de grafos. Los servicios *Web* han sido una parte esencial de la gestión de los extractos de datos de ida y vuelta según las decisiones del usuario. La arquitectura de modelo vista y controlador implementada nos permite adaptar fácilmente las próximas funciones para este proyecto, que refleja los flujos de trabajo, la lógica empresarial y, por lo tanto, la interacción del usuario [52]. Junto con un conjunto de tecnologías modernas elegidas, *NEO4J, PYTHON y JAVASCRIPT*, lo que significa que, con el paso del tiempo y los nuevos requisitos, se adaptarían fácilmente.

Los objetivos descritos en la sección 1.4 de este trabajo de obtención de grado están satisfactoriamente completados tanto de modo general como de modo particular, lo presentado en la sección Framework de análisis y visualización de grafos y resultados lo demuestran.

6.2. Trabajo Futuro

Se identifican algunas áreas de oportunidad para la próxima versión de este proyecto, por ejemplo, agregar más modelos de visualización, opciones personalizables para colorear y guardar grafos para su posterior comparación y análisis sobre todos los modelos de visualización. Crear una interfaz con una mejor experiencia de navegación para usuario donde pueda configurar de forma visual la conectividad con la base de datos, por último, agregar algoritmos de ejecución desde un menú en el *front-end* en lugar de hacerlo en el *middleware*. Así como también, un análisis de *performance* más profundo para el manejo de grafos de un tamaño mayor a 1 millos de nodos y relaciones con la intención de tener un parámetro base de requerimientos de sistema para su procesamiento. Se espera que salgan nuevas necesidades de desarrollo conforme se esté utilizando esta solución de manera frecuente.

BIBLIOGRAFÍA

[1] D. Simpson, “Big Data:a tool for inclusion or exclusion,” in *The Use of Big Data : Benefits, Risks, and Differential Pricing Issues*, New York: Nova Science Publishers, Inc, 2016.

[2] Amine Louati, Marie-Aude Aufaure, and Yves Lechevallier, “Graph Aggregation : Application to Social Networks.,” *HDSDA*, p. 157, 2011.

[3] C. Coronel, S. Morris, and P. Rob, *Bases de Datos, Diseño, Implementacion y Administracion*. Colombia, South America, 2016.

[4] John Faber, “Big Data: Clasificación en base de la estructura,” *JOHNEl fácil acceso a la información*, Jul. 27, 2016. https://johnfaberblog.wordpress.com/2016/07/27/big-data-clasificacion-en-base-de-la-estructura/ (accessed Sep. 14, 2018).

[5] Luis Fernando Gutiérrez Preciado, Francisco Cervantes, and Víctor Hugo Ortega Guzmán, “Analyzing summary graphs by applying aggregation functions,” *International Journal of Hybrid Information Technology*, p. 5, 2020.

[6] M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, and A. rokhim, “Design an MVC Model using Python for Flask Framework Development,” *2019 International Electronics Symposium (IES), Electronics Symposium (IES), 2019 International*, pp. 214–219, Sep. 2019, doi: 10.1109/ELECSYM.2019.8901656.

[7] M. Buerli, “The Current State of Graph Databases,” *Department of Computer Science*, p. 7, Dec. 2012.

[8] Shengdong Zhao, M. J. McGuffin, and M. H. Chignell, “Elastic hierarchies: combining treemaps and node-link diagrams,” *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on, Information Visualization*, pp. 57–64, Jan. 2005

[9] N. Henry, J.-D. Fekete, and M. J. McGuffin, “NodeTrix: a Hybrid Visualization of Social Networks,” *IEEE Transactions on Visualization and Computer Graphics, Visualization and Computer Graphics, IEEE Transactions on, IEEE Trans. Visual. Comput. Graphics*, vol. 13, no. 6, Nov. 2007.

[10] D. Holten, “Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data,” *IEEE Transactions on Visualization and Computer Graphics, Visualization and Computer Graphics, IEEE Transactions on, IEEE Trans. Visual. Comput. Graphics*, vol. 12, no. 5, pp. 741–748, Sep. 2006.

[11] Andreas Kanavos, Georgios Drakopoulos, and Athanasios K. Tsakalidis, “Graph Community Discovery Algorithms in Neo4j with a Regularization-based Evaluation Metric.,” *WEBIST*, p. 8, 2017.

[12] Richard Brath and David Jonker, *Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data*. Indianapolis, IN: John Wiley & Sons, Inc., 2015.

[13] Social Media Research Foundation, “NodeXL | Your Social Network Analysis Tool for Social Media,” *Social Media Research Foundation*. https://www.smrfoundation.org/nodexl/ (accessed Feb. 24, 2020).

[14] S. M. F. Akhtar, “The cypher query language,” in *Big data architect’s handbook: a guide to building proficiency in tools and systems used by leading big data experts*, Packt Publishing, 2018.

[15] Corey L. Lanum, *Visualizing Graph Data*. Shelter Island, NY: Manning Publications, 2016.
[16] "Bioinformatics and Biomarker Discovery: 'Omic' Data Analysis for Personalized Medicine.," *Information Technology Business*, NewsRX LLC, 2010.
[17] Stephen H. Muggleton and Huma M. Lodhi, *Elements of Computational Systems Biology*. Wiley, 2010.
[18] "Visualizing a Neo4j Graph Database," *yWorks, the diagramming experts*. https://www.yworks.com/pages/visualizing-a-neo4j-graph-database (accessed Mar. 02, 2020).
[19] yWorks company the diagramming, "Visualizing Neo4j Database Contents Like a Pro!," *yWorks, the diagramming company*, Jun. 12, 2018. https://www.yworks.com/blog/neo4j-visualization-like-a-pro (accessed Mar. 02, 2020).
[20] C. Maté Jiménez, "Big data. Un nuevo paradigma de análisis de datos," *Anales de Mecánica y Electricidad*, vol. XCI, no. VI, pp. 10–16, 2014.
[21] K. KAMBATLA, G. KOLLIAS, V. KUMAR, and A. GRAMA, "Trends in big data analytics : Perspectives on Parallel and Distributed Processing," *Journal of parallel and distributed computing (Print)*, no. 7, p. 2561, 2014.
[22] "Vista de Big Data: el Valor de la Información Personal y la Privacidad." https://www.jdc.edu.co/revistas/index.php/rciyt/article/view/75/71 (accessed Sep. 14, 2018).
[23] Nataraj Dasgupta, "Sources of big data," in *Practical Big Data Analytics*, Packt Publishing Ltd., 2018.
[24] M. Parsian, *Data algorithms: recipes for scaling up with Hadoop and Spark*, 1. ed. Beijing: O'Reilly, 2015.
[25] R. S. Kevin Wayne, "APPLICATIONS," in *Algorithms*, Fourth., Addison-Wesley Professional, 2011.
[26] C. Pinilla, M. Bello, and C. Peña, "Bases de datos orientadas a grafos," *Tecnología Investigación y Academia*, vol. 5, no. 2, p. 8, 2017.
[27] Hansel Gracia del Busto and Osmel Yanes Enríquez, "Bases de datos NoSQL," *Septiembre-Diciembre, 2012*, Revista Telem@tica., pp. 21–33, Sep. 2012.
[28] Neo4j, Inc., "The Neo4j Graph Platform," *Neo4j Graph Database Platform*, Jan. 26, 2020. https://neo4j.com/product/ (accessed Jan. 26, 2020).
[29] S. Batra and C. Tyagi, "Comparative Analysis of Relational And Graph Databases," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, p. 4, May 2012.
[30] G. Drakopoulos, A. Baroutiadi, and V. Megalooikonomou, "Higher order graph centrality measures for Neo4j," *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*, pp. 1–6, Jul. 2015.
[31] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, "Fast unfolding of community hierarchies in large networks," *CoRR*, 2008, [Online]. Available: http://ezproxy.iteso.mx/login?qurl=http%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3dedsdbl%26AN%3dedsdbl.journals.corr.abs.0803.0476%26lang%3des%26site%3deds-live.

[32] T. von Landesberger, A. Kuijper, T. Schreck, J. van Wijk, J.-D. Fekete, and D. W. Fellner, “Visual analysis of large graphs : State-of-the-art and future research challenges,” *Computer Graphics Forum*, pp. 1719–1749, 2011.
[33] “The Neo4j Graph Algorithms User Guide v3.5.” https://neo4j.com/docs/graph-algorithms/current/ (accessed Nov. 10, 2019).
[34] Iqbal H. Sarker and K. Apu, “MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application,” *International Journal of Hybrid Information Technology*, no. 5, p. 317, 2014.
[35] D.-P. Pop and A. Altar, “Designing an MVC Model for Rapid Web Application Development,” 2014.
[36] “XML Web Services.” https://www.w3schools.com/xml/xml_services.asp (accessed Feb. 08, 2020).
[37] I. G. Baltopoulos, “Introduction to Web Services,” presented at the CERN School of Computing (iCSC), Geneva, Switzerland, 2005, [Online]. Available: https://www.cl.cam.ac.uk/~ib249/teaching/Lecture1.handout.pdf.
[38] R. A. Bahlool and A. M. Zeki, “Comparative Study between Web Services Technologies: REST and WSDL,” *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2019 International Conference on*, pp. 1–4, Sep. 2019.
[39] Erin Cavanaugh, “Web services: Benefits,challenges, and a unique,visual development solution,” *Altova WhitePaper*, Beverly, MA, p. 21, 2006.
[40] B. C. McNurlin and R. H. Sprgue hijo, *Information Systems Management in Practice.*, 7a edición. Prentice-Hall, 2002.
[41] Jake VanderPlas and Matt Harrison, *What Is Python?* Sebastopol, CA: O’Reilly Media, Inc., 2018.
[42] Ryan Mitchell, “Preface,” in *Web Scraping with Python, 2nd Edition*, Sebastopol, CA: O’Reilly Media, Inc., 2018.
[43] Sam Washington, Dr. M. O. Faruque Sarker, and Jose Manuel Ortega, “Introducing aiohttp,” in *Learning Python Networking - Second Edition*, Packt Publishing, 2019.
[44] R. Kosara, “Presentation-Oriented Visualization Techniques,” *IEEE Computer Graphics and Applications, Computer Graphics and Applications, IEEE, IEEE Comput. Grap. Appl.*, vol. 36, no. 1, pp. 80–85, Jan. 2016.
[45] M. Ghoniem, J.-D. Fekete, and P. Castagliola, “A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations,” *IEEE Symposium on Information Visualization, Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 17–24, Jan. 2000.
[46] M. Bostock, “D3.js - Data-Driven Documents,” Feb. 2020. https://d3js.org/ (accessed Feb. 09, 2020).
[47] Matthew Huntington, *Getting Started with D3.js - D3.js Quick Start Guide*. Birmingham, UK.: Packt Publishing Ltd., 2018.
[48] Mike Bostock, “D3 Gallery,” Feb. 2020. https://observablehq.com/@d3/gallery (accessed Feb. 09, 2020).
[49] M. Needham and A. E. Hodler, *Graph algorithms: practical examples in Apache Spark and Neo4j*. 2019.

[50] E. Drabiková and E. F. Škrabul'áková, "Reducing costs by graph algorithms," in *2018 19th International Carpathian Control Conference (ICCC)*, May 2018, pp. 113–117, doi: 10.1109/CarpathianCC.2018.8399612.
[51] R. Faran and O. Kupferman, "A Parametrized Analysis of Algorithms on Hierarchical Graphs," p. 21,2017.
[52] A. Holzinger, K. H. Struggl, and M. Debevc, "Applying Model-View-Controller (MVC) in design and development of information systems: An example of smart assistive script breakdown in an e-Business application," in *2010 International Conference on e-Business (ICE-B)*, Jul. 2010, pp. 1–6.