Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros de Telecomunicación

RECOMENDADOR PRO-ACTIVO Y SENSIBLE A CONTEXTO DE ELEMENTOS EDUCATIVOS REUTILIZABLES

TRABAJO FIN DE MÁSTER

Washington A. Velásquez Vargas

Ingeniero en Telemática

2014

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

RECOMENDADOR PRO-ACTIVO Y SENSIBLE A CONTEXTO DE ELEMENTOS EDUCATIVOS REUTILIZABLES

Autor

Washington A. Velásquez Vargas

Director

Joaquín Salvachúa Rodríguez

Departamento de Ingeniería de Sistemas Telemáticos

2014

Resumen

El presente proyecto consiste en la implementación de un sistema recomendador sensible al contexto basado en elementos educativos reutilizables, estos recursos serán obtenidos a través del uso de las APIs de Youtube, Flickr y Soundcloud para lograr tener videos, imágenes y músicas respectivamente, y poder administrarlos desde una fuente externa, en nuestro caso será una interfaz de búsqueda. La base de datos utilizada en el desarrollo del proyecto es una base de datos orientada a grafos "Neo4j" que actualmente está tomando buena acogida en el mercado debido a su funcionalidad y alta escalabilidad que posee.

Para la realización del proyecto se aplican dos etapas. La primera etapa está basada en obtener los recursos; esto lo hacemos mediante las búsquedas por parte de los usuarios. En esta etapa, existen dos caminos al momento de almacenar los recursos en la base, que son:

- Si el usuario no está registrado en la base (como un usuario del sistema), las búsquedas no se asociarán al usuario, sino que solamente se creara el objeto del recurso y servirá para usuarios en futuras búsquedas.
- Si el usuario está registrado, todas las búsquedas se asocian a su perfil y su vez se almacenan como objeto independiente.

La segunda etapa, consta de la recomendación del sistema a los usuarios. Esta etapa solo puede ser posible a los usuarios que estén registrados en el sistema.

El proyecto se lo ha estructurado en 4 capítulos como se lo explica a continuación:

En el primer capítulo, se exponen los objetivos generales y específicos del proyecto, además de especificar claramente los alcances y limitaciones.

En el segundo capítulo, se da a conocer la parte teórica, explicando los conceptos básicos de los elementos y software usados para el desarrollo del proyecto, además de conocimientos adicionales que se tomó en cuenta con lo que respecta al manejo de las interfaces de programación "APIs".

En el tercer capítulo, se describe el desarrollo e implementación del proyecto, explicando claramente las instalaciones de los modulo y paquetes necesarios para su funcionamiento, se describe paso a paso las distintas etapas que se tuvieron que implementar para cumplir con los objetivos propuestos.

En el cuarto capítulo, se presentan los resultados obtenidos de toda la implementación, así mismo se presentan ejemplos de búsquedas y se puede notar claramente las recomendaciones que el sistema realiza.

Finalmente, en base a los resultados obtenidos estamos en la capacidad de dar una conclusión general de toda la funcionalidad del proyecto, la misma que nos permite proporcionar recomendaciones que servirán como base para futuras aplicaciones o mejoras que se le quiera dar al presente proyecto.

Abstract

This project involves the implementation of a context-aware recommender systems based on reusable educational elements , these resources will be obtained through the use of APIs from Youtube, Flickr and Soundcloud to achieve have videos, pictures and music respectively, and to manage from an external source, in our case will be a search interface. The database used in the project is a database oriented graphs "Neo4j" it's taking good reception in the market due to its high functionality and scalability it has.

For the project two stages are applied. The first stage is based on obtaining the resources; we do this by searching by users. At this stage, there are two paths when storing in the database the resources, which are:

- If the user is not registered to the base (as a user of the system), searches will not be associated to the user, but only the subject of the application is created and will serve to users in future searches.
- If the user is registered, all searches are associated with their profile and time is stored as a separate object.

The second stage consists of the recommendation system to users. This step can only be possible for users who are registered in the system.

The project has structured in 4 chapters as explained below:

In the first chapter, the general and specific objectives of the project are discussed, in addition to clearly specify the scope and limitations.

In the second chapter, disclosed the theoretical part, explaining the basics of the elements and software used for the project, plus additional knowledge was taken into account with regard to the management of programming interfaces "APIs ".

In the third chapter, the development and implementation of the project is described, clearly explaining the facilities of the module and packages required for its operation, is described step by step the various stages that had to be implemented to meet the objectives.

The results of all the implementation are presented in the fourth chapter, also examples of searches are presented and recommendations can clearly notice that the system performs.

Finally, based on the results we obtained are in the ability to give an overall assessment of all the functionality of the project, which allows us to provide recommendations to serve as a basis for future applications or improvements that you want to give to this project.

Índice general

Índice de figuras

Siglas

ACID	Atomicity, Consistency, Isolation and Durability
API	Application Programming Interface
APP	Application
BDOG	Bases de Datos Orientados a Grafos
CNRI	Corporación Internacional para la Búsqueda de Iniciativas
CRUD	Create, Read, Update, Delete
DM	Data Mining
NoSQL	Not only SQL
OLAP	On-Line Analytical Processing
OLTP	Online Transaction Processing
PSF	Python Software Foundation
RDBMS	Relational Database Management System
REST	Representational State Transfer
SOR	System of Record
SQL	Structured Query Language
SRSC	Sistemas de Recomendación Sensibles al Contexto

Introducción

En algunas circunstancias nos vemos en la necesidad de seleccionar entre varias alternativas sin tener un conocimiento fijo de cada una de ellas. En estas situaciones, la decisión final puede depender de las recomendaciones que tengamos de otras personas, pero; actualmente las empresas están usando un estilo de marketing personalizado para guiar a los usuarios mediante el uso de recomendaciones.

El nombre que se les ha dado a estos sistemas es de "Sistemas de Recomendación – (SR)". Estos sistemas se encargan de suministrar a los usuarios información personalizada y diferenciada sobre determinados productos y/o servicios que pueden ser de interés. Es decir, se encargan de guiar al usuario mediante recomendaciones en la búsqueda de aquellos servicios o productos que puedan ser más atractivos, modificando el proceso de navegación o búsqueda. El uso de estos sistemas se está poniendo cada vez más de moda debido a su utilidad para evaluar y filtrar la gran cantidad de información disponible en la Web, para asistir a los usuarios en sus procesos de búsqueda y recuperación de información. Esto es sin duda una gran ventaja para los clientes, que encontrarán lo que necesitan de una forma más rápida, cómoda y fácil dentro de las tiendas electrónicas en Internet y además descubrirán nuevos productos o servicios que le puedan ser atractivos, que de una u otra manera les hubiese sido mucho más difícil o incluso imposible de encontrar.

Una vez comentado brevemente la situación actual del mercado de Internet y las características de los novedosos y útiles sistemas de recomendación podemos pasar a introducir brevemente en qué consiste el proyecto que se documenta en esta memoria.

La idea principal de este trabajo es que se pretende realizar un algoritmo de recomendación basado en filtrado colaborativos bajo el contexto "observado-a-observar", en donde las fuentes de búsqueda provienen de tres grandes empresas Web que son: Youtube, Flickr y SoundCloud. Para ello utilizaremos las interfaces de programación que proporcionan cada una de ellas para conectarse a sus bases de datos y así poder gestionar las búsquedas. Luego, procedemos a almacenarlas en la base de datos, que por motivos de este proyecto utilizaremos la base de datos orientada a grafos "Neo4j" por su robustez y alta escalabilidad que posee.

Capítulo 1

1 Generalidades

1.1 Justificación

Un Recomendador es un sistema que se encarga de suministrar a los usuarios información personalizada y diferenciada sobre determinados productos y/o servicios que pueden ser de interés en base a su perfil o búsquedas anteriormente realizadas. Como sabemos estos sistemas ya acaparan el mercado, el motivo de realizar este proyecto es de agrupar tres fuentes de búsquedas principales, en donde podamos obtener elementos de videos, fotos y músicas que sirvan para carácter educativo agrupándolos en una sola fuente de búsqueda y así facilitar enormemente el trabajo del usuario.

Además, independientemente de que el usuario ingrese a la aplicación, cada una de sus búsquedas queda guardada en la base "Neo4j", facilitando la exploración posterior de recursos por parte de los usuarios en nuestra base de datos.

Finalmente, si el usuario decide ingresar al sistema, automáticamente se ejecuta el algoritmo de recomendación basado en la técnica de filtrado colaborativo para darle a conocer sugerencias de videos, fotos o músicas en base al perfil del usuario en sesión.

1.2 Identificación del Problema

Debido a la gran masa de información que actualmente circula en la red, es necesario poder conocer que información es útil y cual es denominada basura, por tal motivo, se pretende implementar un sistema de información basado en recolección de elementos de otras fuentes, en donde estos archivos sirvan para el ámbito educativo.

Finalmente, logrando tener toda esta información en una base de datos orientada a grafos que facilite las búsquedas de archivos por parte de los usuarios y al mismo tiempo generar un algoritmo de recomendación que facilite dichas búsquedas, permitirá a los usuarios utilizar está información de forma productiva.

1.3 Objetivos

1.3.1 Objetivos Generales

- Aprender el manejo de interfaces de programación de varias aplicaciones para que mediante servicios Restfull poder acceder a sus archivos y de esta manera extraer los atributos principales de cada uno de ellos.

- Obtener habilidades en el diseño e implementación de algoritmos de recomendación para dar solución a problemas específicos que se presenta al tener gran cantidad de información circulando en la web.

- Comprender el uso de la base de dato orientada a grafos "Neo4j", para lograr diseñar un modelo óptimo que facilite la búsqueda de recursos por parte de los usuarios.

1.3.2 Objetivos Específicos

- Comprender el uso de las APIs de Youtube, Flickr y SoundCloud para extraer archivos de carácter educativo.

- Implementar un algoritmo general que extraiga la información de las fuentes externas y al mismo tiempo permita almacenar la información en la base de datos Neo4j.

- Diseñar un modelo de objeto adecuado de cada uno de los recursos obtenidos de las fuentes externas.

- Implementar un algoritmo de recomendación que permita inferir soluciones a los usuarios en base a su contexto de búsqueda y/o perfil.

- Implementar una interfaz web amigable que permita al usuario poder realizar búsquedas de recursos educativos y al mismo tiempo poder observar el rendimiento de todos los algoritmos desarrollados.

1.4 Alcances y Limitaciones

Los sistemas de información como sus aplicaciones en el mundo real brindan un sin número de aplicaciones, en este caso se trata de la implementación de un recomendador proactivo, pero siendo este un proyecto con finalidad académica se presentan a continuación alcances y limitaciones del mismo:

1.4.1 Alcances

- El recomendador podrá hacer búsquedas en Youtube, Flickr, y SoundCloud a través de las interfaces de programación.

- La información obtenida por las búsquedas de los usuarios se irán almacenando en la base de datos Neo4j.

- Se asocian las búsquedas del usuario para que en el futuro indicarle una recomendación en base a sus anteriores búsquedas.

1.4.2 Limitaciones

- Las pruebas del recomendador serán realizadas en un ambiente de laboratorio.

1.5 Tecnologías a utilizar

- Sistema Operativo Debian – Linux
- Interfaces de programación de aplicaciones
    - Youtube
    - Flickr
    - SoundCloud
- Lenguaje Python 2.7.3 o superior
- Bases de Datos orientada a Grafos “Neo4j”

Capítulo 2

2 Estado del Arte

2.1 Python

El lenguaje de programación Python fue creado a principios de 1990 por Guido van Rossum en Holanda como sucesor de un lenguaje anterior llamado ABC. En 1995 Guido continuó trabajando en Python en el CNRI (Corporación internacional para la búsqueda de iniciativas) en Reston, Virginia. Durante esta época lanzó varias versiones del lenguaje. A finales del año 2000 el equipo de Python se trasladó a Zope Corporation y un año después se formó PSF (Python Software Fundation), siendo Zope Corporation su principal patrocinador.

Todas las versiones de Python son Open Source. Además, la mayoría de las versiones de Python que han sido liberadas desde su creación también han tenido licencia GPL (las únicas sin licencia GPL son la 1.6, 2.0, 1.6.1 y 2.1). [1]. A continuación en la (**Figura 1)** se presenta el logo de Python:

Figura 1. Logo de Python [1]

2.1.1 ¿Qué es Python?

Python es un lenguaje de programación interpretado orientado a objetos de muy alto nivel. Sus estructuras de datos de alto nivel, combinadas con tipiado dinámico y ligadura dinámica lo hacen especialmente atractivo para el desarrollo de aplicaciones rápidas, así como lenguaje de scripts o medio de conexión de otros componentes ya existentes. Además, la claridad y sencillez de su sintaxis mejora notablemente la

legibilidad de los programas escritos en Python y por lo tanto reduce el coste de mantenimiento del software. Python soporta además módulos y paquetes, lo que incrementa la modularidad y la reutilización de código. Permite incorporar rutinas compiladas en C para realizar funciones críticas a alta velocidad. También implementa estructuras de datos muy avanzadas (lista, tuplas, diccionarios) que pueden ser combinadas para crear otras estructuras más complejas. [2]

Su intérprete y la librería de Python están disponibles para las principales plataformas (Linux, Windows, UNIX...) en código fuente o binario y pueden ser distribuidas libremente. Asimismo, Python se caracteriza por disponer de una excelente documentación y una de las mejores guías de aprendizaje. [2]

Python está escrito en C. Existe una implementación del lenguaje Python implementada únicamente en Java: Jython. La mayoría de las librerías del Python estándar han sido ya portadas a Jython, sin embargo el desarrollo de Python es tan rápido que los desarrolladores de Jython tienen problemas para portar todas las innovaciones. Para ejecutar el código de este lenguaje es necesario un intérprete, que va recibiendo líneas de código y las traducirá a lenguaje máquina para que se ejecuten. A diferencia de los lenguajes compilados no se produce código ejecutable, de modo que para ejecutar el código en diferentes plataformas solo hay que cambiar el intérprete, no el código. [2]

Existen diversas implementaciones del lenguaje:

- CPython es la implementación original, disponible para varias plataformas en el sitio oficial de Python.
- IronPython es la implementación para .NET
- Stackless Python es la variante de CPython que trata de no usar el stack de C
- Jython es la implementación hecha en Java
- Pippy es la implementación realizada para Palm
- PyPy es una implementación de Python escrita en Python y optimizada mediante JIT

2.2 Almacenamiento de Datos

En la actualidad existe una gran polémica alrededor del tema relacionado con la gestión de la información digital que se necesita almacenar, para su posterior recuperación y análisis por parte de los diseñadores, arquitectos y desarrolladores de aplicaciones informáticas de cualquier tipo. Por un lado están los tradicionales sistemas de gestión de bases de datos relacionales (RDBMS, por sus siglas en inglés) y por otro

los prometedores sistemas de bases de datos no relacionales y distribuidos conocidos como NoSQL (Not only SQL).

2.2.1 Bases de Datos Relacionales

El modelo relacional de datos supuso un gran avance con respecto a los modelos anteriores. Este modelo está basado en el concepto de relación. Una relación es un conjunto de n-tuplas. Una dupla, al contrario que un segmento, puede representar tanto entidades como interrelaciones (N : M). Los lenguajes matemáticos sobre los que se asienta el modelo relacional, el álgebra y el cálculo relacionales, aportan un sistema de acceso y consultas orientado al conjunto. La repercusión del modelo en los DBMSs comerciales actuales ha sido enorme, estando hoy en día la gran mayoría de los gestores de bases de datos basados en mayor o menor medida en el modelo relacional.

El concepto de modelo de datos en sí surgió al mismo tiempo que el modelo relacional de datos fuera propuesto por su creador, Ted Codd, después de que los modelos jerárquico y de red estuvieran en uso. Posteriormente, estos dos modelos fueron definidos independientemente de los lenguajes y sistemas usados para implementarlos. Con anterioridad no eran más que colecciones de estructuras de datos y lenguajes sin una teoría subyacente definida. En cuanto al modelo relacional, no se puede decir que sea en sí un modelo semántico de datos. Su enorme éxito no se debe a que permite de forma implícita operaciones conceptualmente abstractas sobre los datos, sino a los altos niveles de fiabilidad e integridad que aporta en el manejo de grandes cantidades de datos. [3]

Lo que realmente marca la diferencia entre los sistemas relacionales y los sistemas anteriores es el hecho de que su creador, Ted Codd, basó expresamente su funcionamiento sobre un modelo matemático muy específico: el álgebra relacional y el cálculo relacional, así como la progresiva adopción, por parte de su creador y algunos colaboradores, de un número de Reglas de Integridad Relacional y de Formas Normales, pero los programadores por varias décadas han venido revisando muy detalladamente la forma en que se implementan las bases de datos relaciones, esto se debe, a la falta de relaciones que existen en ellas. Es decir, para que una tabla pueda tener una relación con otra se debe tener un campo que contenga esa relación, con esto tendremos repetición de datos. Lo podemos observar en la **(Figura 2)**, en el ejemplo se ve claramente como el *"id"* de la tabla departamento pasa a la tabla profesor y así mismo el *"id"* del profesor es llevado a la tabla curso para mantener las relaciones entre ellas.

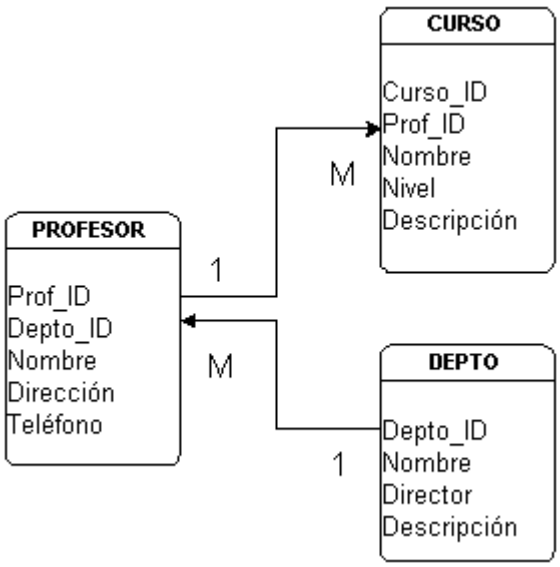

Figura 2. Modelo Relacional [3]

Las aplicaciones van a depender del modelo de datos en que estén implementadas y las consultas de datos se podrían facilitar o complicar dependiendo de la funcionalidad que posea la aplicación:

- Las tablas de "Join" añaden complejidad a la aplicación debido al uso de "Foreign Key"
- Las "Foreign Key" mantienen una sobrecarga adicional solo para mantener a las bases de datos relacionales
- Columnas con valores de "NULL" requieren una comprobación especial
- Sentencias SQL complejas al interactuar con varias tablas de la base

Finalmente, hemos de aclarar que las relaciones no sólo son las "tablas" de la base de datos, sino que también son los distintos tipos de relaciones que se pueden generar mediante consultas a las relaciones base. Podemos distinguir los siguientes tipos de relaciones:

- Relaciones base o reales: es lo que corresponde al concepto de tabla. El conjunto de éstas son las que componen la base de datos realmente.
- Conjunto dinámico de datos: no poseen datos almacenados propios y están representadas únicamente dentro del sistema mediante su definición en términos de otras relaciones (es decir, mediante consultas). También conocidas como dynasets.
- Instantáneas (snapshots): iguales que las anteriores, pero los datos que contienen no son virtuales, sino que están realmente almacenados en la instantánea. Se utilizan para manejar datos susceptibles de cambios.
- Resultados de consultas: la relación resultante de consultar una o más relaciones base.

- Resultados intermedios: el resultado de una operación anidada en una consulta, estos resultados son usados por la consulta externa para otra operación. La facilidad de anidar consultas dota de gran potencia al lenguaje de consulta relacional (SQL). [3]

2.2.1.1 Ventajas e inconvenientes del modelo relacional

Las ventajas de utilizar un RDBMS podrían ser resumidas en las siguientes:

- Compatibilidad y estandarización.
- Fiabilidad.
- Garantía de independencia de los datos.
- Existencia de numerosos sistemas comerciales entre los que escoger y consiguiente apoyo técnico.
- Conectividad garantizada con los lenguajes de programación estándar.

Sin embargo, también hemos de ser conscientes de los aspectos negativos, o más bien limitaciones, que conlleva la adopción de un modelo de datos con una veintena de años. Existen una serie de desventajas bien conocidas del modelo relacional de datos, que se ponen de manifiesto especialmente cuando lo comparamos con otros modelos más nuevos (p. ej. el modelo orientado al objeto o las modernas implementaciones basadas en marcos). Las más obvias son las siguientes:

- Imposibilidad de representar conocimiento en forma de reglas.
- Inexistencia de mecanismos de herencia de propiedades (y por supuesto de métodos).
- Falta de poder expresivo (por ejemplo, para representar jerarquías).
- Dificultad para gestionar datos no atómicos (p. ej. los valores estructurados de una estructura de rasgos).
- Incompatibilidad entre los tipos de estructuras de datos que se transfieren o desadaptación de impedancia (impedance mismatch).

Los cuatro primeros aspectos afectan directamente a la representación léxica, mientras que el último es un problema meramente técnico.

En resumen, un RDBMS supone una plataforma estable y compatible, con limitaciones en sus capacidades y poder expresivo. En este estado de cosas, pensamos que un cuidado diseño (modelado conceptual) puede vencer muchas de estas desventajas y aprovechar al máximo todas las ventajas mencionadas. La evolución del modelo relacional pasa por los modelos semánticos de datos, o de cuarta generación [3].

2.2.2 Bases de Datos NoSQL

El término NoSQL (No solo SQL) se presentan como un enfoque moderno de almacenar las bases de datos, con una amplia clase de sistemas de gestión de datos que difieren del modelo clásico relacional en aspectos muy importantes, el más destacado es que no usan SQL como el principal lenguaje de consultas, normalmente estas bases de datos no soportan "JOIN" para unir tablas como lo hace el modelo relacional. Actualmente, en el mercado nos podemos encontrar con muchas soluciones flexibles y adaptables a un uso en específico. La intención original ha sido modernizar las bases de datos por lo que se presenta a continuación las diferentes implementaciones que se tienen de estas bases: [4]

- Basados en Familias de Columnas
  - Hadoop/HBase
  - Cassandra
  - Hypertable
  - Accumulo
  - Amazon SimpleDB
  - Cloudata
  - Cloudera
  - HPCC
  - Stratosphere
- Almacenamiento de Documentos
  - MongoDB
  - Elasticsearch
  - Couchbase Server
  - CouchDB
  - RethinkDB
  - RavenDB
  - MarkLogic Server
  - Clusterpoint Server
  - ThruDB
  - Terrastore
  - JasDB
  - RaptorDB
  - Djondb
  - EJDB
  - Amisa Server
  - Densodb
  - SisoDB
  - SDB
  - NoSQL embedded db
- Clave-Valor / Almacenamiento basado en Duplas
  - DynamoDB
  - Azure Table Storage
  - Riak
  - Redis
  - Aerospike
  - FoundationDB
  - LevelDB
  - Berkeley DB
  - GenieDB
  - BangDB
  - Chodless
  - Scaralis
  - Tokyo Cabinet/ Tyrant
  - Scalien

    - Voldemort
    - Dynomite
    - KAI
    - MemcacheDB
- Bases de datos orientadas a grafos.
    - Neo4j
    - Infinite Graph
    - DFX
    - InfoGrid
    - HyperGraphDB
    - GraphBase
    - Trinity
    - AllegroGraph
    - BrightstarDB
    - BigData
    - Meronymy
    - WhiteDB
    - VertexDB
    - FlockDB
    - Execom IOG
    - Fallen 8
- Bases de datos multimodelos
    - ArangoDB
    - OrientDB
    - Datomic
    - FatDB
    - AlchemyDB
    - CortexDB
- Bases de datos orientado a Objetos
    - Versant
    - Db4o
    - Objectivity
    - Starcounter
    - Perst
    - VelocityDB
    - HSS Database
    - ZODB
    - Magma
    - NEO
    - Siaqodb
    - Sterling
    - Morantex
    - EyeDB
    - FramerD
    - NDatabse
    - PicoLisp
    - Acid-state
    - ObjectDB
- Soluciones de Bases de datos en Grid & Cloud
    - GigaSpaces
    - GemFire
    - Infinispan
    - Queplix
    - Hazelcast
- Bases de datos en XML
    - EMC Documentum xDB
    - eXist
    - Sedna
    - BaseX
    - Qizx
    - Berkeley DB XML
- Bases de datos Multidimensionales

    - Globals
    - Intersystems Cache
    - GT. M
    - SciDB
    - MiniM DB
    - Rasdaman
- Bases de datos Multivalores
    - U2
    - OpenInsight
    - TigerLogic Pick
    - Reality
    - OpenQM
    - Model 204 Database
    - ESENT
    - jBASE

El enfoque general del algoritmo se encuentra en las bases de datos orientadas a grafos, por lo que nos centraremos en explicar su estructura, el uso y la implementación.

2.3 Bases de Datos Orientadas a Grafos

Con la alta demanda que se requiere al momento de almacenar información y la gran capacidad que se necesita en las aplicaciones informáticas cuando se realizan consultas sobre ellas, ha generado la búsqueda de nuevas herramientas tecnológicas que ayuden a tener una mejor gestión de los datos. Recordemos que en muchas ocasiones los datos necesitan ser analizados e interpretados para convertirlos en información útil y provechosa, tal situación ha propiciado una creciente necesidad por técnicas/herramientas computacionales que nos ayuden en estas tareas. Por lo tanto, si pudiéramos representar los datos y las relaciones entre objetos como un solo conjunto de datos, podríamos generar patrones de conocimiento que describan nuestro conjunto de datos conteniendo estos elementos de manera conjunta. Para tal efecto se presentan las bases de datos en grafos que son lo suficientemente flexibles y poderosas para representar estos elementos. Se utiliza la estructura de control "grafos".

2.3.1 Estructura de Datos: Grafos

Un grafo es definido como un par G = (V, E), en donde, V denota un conjunto finito de elemento llamados vértices y E es un conjunto de arcos, juntos permiten representar relaciones binarias entre pares de objetos, es decir, un grafo es una colección de vértices y de aristas, donde las aristas representan la relación que se tiene entre los vértices. A continuación en la **(Figura 3)** se presenta una figura de un grafo dirigido:

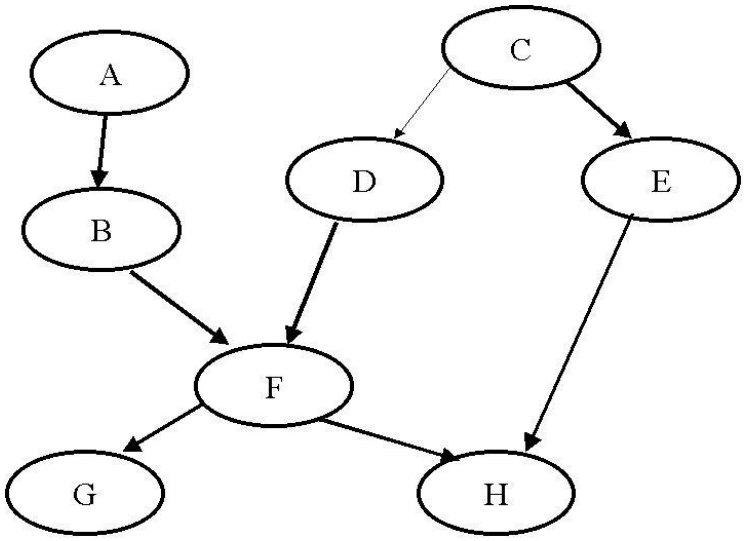

Figura 3. Modelo de Grafos

Los grafos son muy útiles para poder tener una mayor comprensión de una amplia gama de datos, tales como la ciencia, el gobierno y los negocios, por esa razón, cuando entendamos los beneficios de usar grafos en el mundo real podríamos dar respuesta a algunas interrogantes, actualmente esta es una de las tecnologías utilizadas principalmente para la persistencia transaccional en línea (OLTP), la cual permite que se acceda a información en tiempo real usando alguna aplicación, que por lo general, usan la teoría de grafo como estructura de datos. A esta tecnología que permite la gestión de métodos de crear, leer, actualizar y eliminar datos del grafo (CRUD), se le conoce con el nombre de bases de datos orientados a grafos (BDOG). [5] [6]

2.3.2 Definición de una BDOG

Las bases de datos orientadas a grafos representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos ya que esta puede describir atributos de los nodos (entidades) y las aristas (relaciones).

Una base de datos orientada a grafos debe estar absolutamente normalizada, esto quiere decir que cada tabla tendría una sola columna y cada relación tan solo dos, con esto se consigue que cualquier cambio en la estructura de la información tenga un efecto tan solo local. Lo que nos indica que un sistema de gestión de base de datos gráfica se construye generalmente para el uso transaccional de sistemas, optimizar el rendimiento, integridad de los datos y disponibilidad operacional, para entrar en el campo de la investigación de esta tecnología se debe tener en cuenta dos propiedades importantes que poseen las bases orientadas a grafos, que son las siguientes:

- *Almacenamiento Subyacente*

  Algunas bases de datos utilizan el almacenamiento nativo que se ha optimizado y diseñado para el almacenamiento y gestión de los grafos, aunque no todas las bases de grafos la utilizan para el almacenamiento. Sin embargo, algunos serializan los datos en una base de datos relacional, base de datos orientado a objetos o alguna base de datos de uso general.

- *Motor de Procesamiento*

  Se deben tener muy presente el concepto de aristas y vértices cuando se estudia esta tecnología, debido a que las bases orientadas a grafos usan un concepto más amplio de los mismo, pero; visto desde la perspectiva del usuario se comportan como un grafo normal que permite hacer (CRUD) sobre una base de datos.

  Es importante tener en cuenta que el almacenamiento y procesamiento de una base orientada a grafos no son buenas ni malas, son simplemente implementaciones de ingeniería que ayudan al rendimiento de unas ciertas aplicaciones pero así mismo perjudican otras. El beneficio del almacenamiento en grafos y en el que se ha construido su propósito, es el de tener un alto rendimiento y facilidad de escalabilidad.

Por definición, una base de datos orientada a grafos es cualquier sistema de almacenamiento que permite la adyacencia libre de índice. Esto quiere decir que cada elemento contiene un puntero directo a sus elementos adyacentes por lo cual no es necesario realizar consultas por índices.

Otro punto a tener en cuenta, es que las relaciones entre entidades son parte de la teoría de los grafos, por tanto, el manejo de las relaciones se facilita al tener una base de datos orientada a grafos. Al reunir unas simples abstracciones de los nodos y las relaciones en las estructuras que se tiene por las aristas, las bases de datos orientadas a grafos nos permiten construir modelos sofisticados que se asignan arbitrariamente a nuestro dominio del problema, por tanto; los modelos resultantes son más simples y al mismo tiempo más expresivos que los producidos utilizando las bases de datos relacionales tradicionales y otras bases NoSQL.

Este tipo de base de datos está diseñada para los datos cuyas relaciones son bien representadas en forma de grafo, o sea, los datos son elementos interconectados con un número no determinado de relaciones entre ellos. La información a gestionar por este tipo de almacenamiento pudiera ser las relaciones sociales, el transporte público, mapas de carreteras o topologías de red, entre otros ejemplos.

A pesar que las estructuras de datos en forma de grafos son normalizables en teoría, incluso en sistemas relacionales, esto tendría serias implicaciones en el rendimiento de las consultas debido a las características de implementación de bases de datos relacionales. En un sistema relacional cada operación sobre una relación resultaría en una operación de unión para el gestor de datos, lo cual es un proceso lento y no escalable ante un creciente número de tupas en estas tablas.

No existe un consenso general sobre la terminología existente en el área de grafos pues hay muchos tipos diferentes de modelos de grafos. Sin embargo, se están realizando algunos esfuerzos para crear el modelo de Grafo de Propiedad, que unifica la mayoría de las diferentes implementaciones de grafos. De acuerdo con este Modelo, la información en un grafo de propiedad se modela utilizando tres elementos básicos:

- El nodo (vértice)
- La relación (arista) con dirección y tipo (etiquetado y dirigido).
- La propiedad (atributo) en los nodos y en las relaciones.

Una base de datos orientada a grafos almacena datos en un grafo, los datos pueden ser genéricos logrando tener estructuras muy complejas en sus nodos, en el siguiente esquema **(Figura 4)**, podemos observar claramente la estructura de una base de datos orientada a grafos, en donde los grafos constan de archivos que vendrían a ser los "nodos y las relaciones", a su vez estos poseen propiedades dejando por último a las relaciones que vienen a ser las encargadas de organizar los nodos.

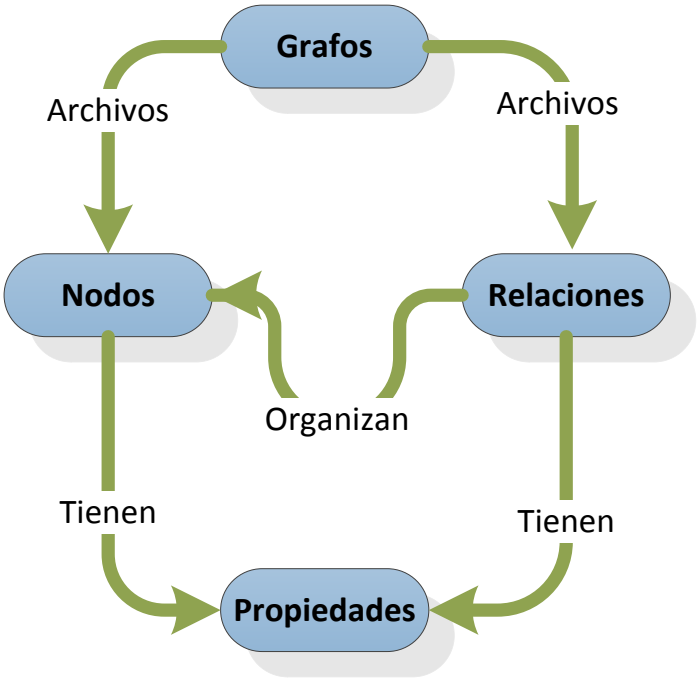

Figura 4. Esquema de una Base de datos Orientada a Grafos

2.3.3 Fortalezas de una BDOG

A pesar de que cualquier cosa puede ser modelada en una base de datos orientada a grafos, debido a que en el mundo en el que vivimos tenemos bastantes plazas de innumerables proyectos con normas corporativas de consumo masivo. Pero que una base de datos orientada a grafos ofrezca una poderosa y novedosas técnica de modelado de datos no constituye una justificación suficiente para la sustitución de una técnica bien conocidas como lo son las bases de datos tradicionales.

En el caso de las bases de datos con grafos existen ciertos casos en los que el uso de datos en una aplicación el rendimiento mejora en uno o más pedidos si se lo implementa con grafos, y cuya latencia es menor en comparación con el procesamiento por lotes tradicional. Además de este beneficio las bases de datos con grafos ofrecen un modelo de datos muy flexible, y un modelo para la entrega de datos con la agilidad que necesita el software, como son [5]:

- *Rendimiento*

  Una razón de importancia para el uso de bases de datos con grafos es el gran rendimiento que ofrecen en comparación con las bases de datos relacionales y otras NoSQL.

  A diferencia de las bases de datos relacionales, donde se deteriora el rendimiento al realizar intensivas consultas que tengan un procesamiento de datos muy alto, pero con el uso de bases de datos con grafos el rendimiento tiende a permanecer constante, así como el crecimiento de los datos. Esto es porque las consultas se realizan de manera gráfica, recorriendo entre las relaciones (aristas) que posea la base. Como resultado, el tiempo de ejecución de esta consulta es proporcional solo al tamaño de la parte grafica que tenga que recorrer para satisfacer esta consulta, en lugar del tamaño global del grafo.

- *Flexibilidad*

  Como desarrolladores nos interesa conectar los datos permitiendo una estructura en donde el esquema que surja del conjunto de problemas pueda crecer sin ningún problema. Las bases de datos orientadas a grafos abordan directamente este tema, porque el modelo de datos se expresa y se acomoda a las necesidades del negocio de una manera en que se permita mover a mayor velocidad en el problema.

  Los grafos son aditivos, lo que significa que podemos añadir nuevos tipos de relaciones, nuevos nodos y sub-grafos a una estructura existente sin alterar las consultas y la funcionalidad de la aplicación. Estas cosas tienen consecuencias positivas en general para desarrolladores debido a la flexibilidad del modelo

con grafos, lo que también tiene tendencia a realizar menos migraciones para poder reducir los gastos de mantenimiento y el riesgo que conlleva.

- *Agilidad*

  A medida que la aplicación valla creciendo se desea así mismo evolucionar el modelo de datos para que se acople a las nuevas necesidades, Las bases de datos nos equipan para llevar a cabo el mantenimiento progresivo de los sistemas, en particular, la naturaleza sin esquema, junto con la comprobación de las aplicaciones y el lenguaje de consulta, damos paso para que evolucionen las aplicaciones en un entorno controlado.

  Al mismo tiempo, porque son libres de esquemas, las bases de datos con grafos carecen de la clase de mecanismos a las que se está familiarizado en las bases de datos relacionales. Pero esto no es un riesgo debido a que estas se alinean al desarrollo ágil del software, lo que permite respaldar la evolución que tienen las aplicaciones actualmente para que se acoplen al ritmo cambiante del entorno de negocio.

2.3.4 Motores de Modelamiento

Modelos de bases de datos gráficas pueden ser definidas como aquellas en los que las estructuras de datos se modelan en forma de gráficos o generalizaciones de ellos, además como la manipulación de datos se expresa por operaciones orientadas a gráficos que poseen sus propios tipos de constructores.

Un motor de modelamiento gráfico es una tecnología que permite a los algoritmos de grafos poder ser ejecutados contra un conjunto de datos. Estos motores están diseñados para identificar los grupos de datos y responder algunas preguntas tales como: "el número de relaciones que posee", el "número de amigos que puede tener en una red social", entre otras interrogantes. [7]

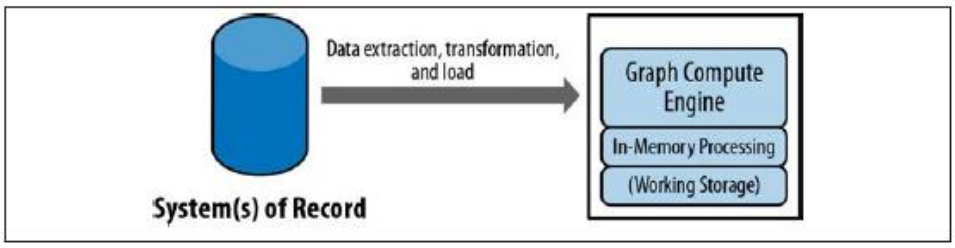

Figura 5. Despliegue típico de un motor Gráfico [5]

En la **(Figura 5),** se muestra una arquitectura común de un motor de bases de datos con grafos, la arquitectura incluye un sistema de registro (SOR), base de datos con

propiedades OLTP (Como MySQL, Oracle o Neo4j), que recibe peticiones y las responde en un tiempo de ejecución adecuado.

Debido a su énfasis en las consultas globales, los motores de computación gráfica son normalmente optimizados para la digitalización y procesamiento de grandes cantidades de información, y en eso son similares a otras tecnologías de análisis de datos, como la minería de datos y OLAP, que están familiarizados en el mundo relacional. Mientras que algunos motores de cómputo gráfico incluyen una capa de almacenamiento, los demás (y posiblemente más) se ocupan estrictamente del procesamiento de datos que se alimenta desde una fuente externa, y así mismo la devolución de los resultados.

La teoría de grafos ha sido testigo de una gran utilidad y pertinencia de muchos problemas en miles de dominios. Los algoritmos de grafos más teóricos aplicados, incluyen varios tipos de cálculos como el camino más corto, rutas geodésicas, medidas de centralidad como PageRank, centralidad del vector propio, cercanía, intermediación, HITS, y muchos otros más. Sin embargo, la aplicación de estos algoritmos, en muchos casos se ha limitado a la investigación, ya que en la práctica no ha habido ningún producto listo para escenarios de alto rendimiento en cuanto a implementaciones de bases de datos orientadas a grafos. Afortunadamente, en los últimos años, esto ha cambiado. Hay varios proyectos que se han desarrollado teniendo en cuenta los grandes escenarios de producción como:

- Neo4j
- Infinite Graph
- InfoGrid
- HyperGraphDB
- DEX
- GraphBase
- Trinity

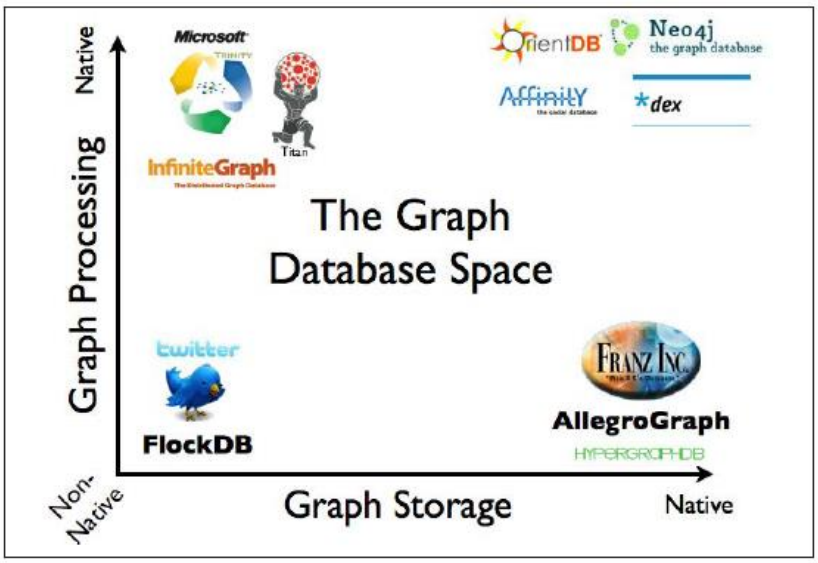

Figura 6. Visión General de los motores Gráficos [5]

En la **(Figura 6)**, se muestra un análisis de los distintos motores que se tiene para usar bases de datos orientada a grafos, la cual se puede apreciar que la base Neo4j debido a su alto rendimiento para almacenar y procesar los datos se encuentra primero en relación a otros motores.

2.3.4.1 Neo4j

Figura 7. Logo de Neo4j [8]

Es un robusto (totalmente ACID) altamente escalable de base de datos orientadas a grafos. Neo4j es utilizado en aplicaciones de misión crítica por miles de nuevas empresas líderes, empresas y gobiernos de todo el mundo. A continuación en la **(Figura 8)** se presenta un esquema de lo que vendría a ser "NEO4J", como implementación en una base de datos orientada a grafos. [8]

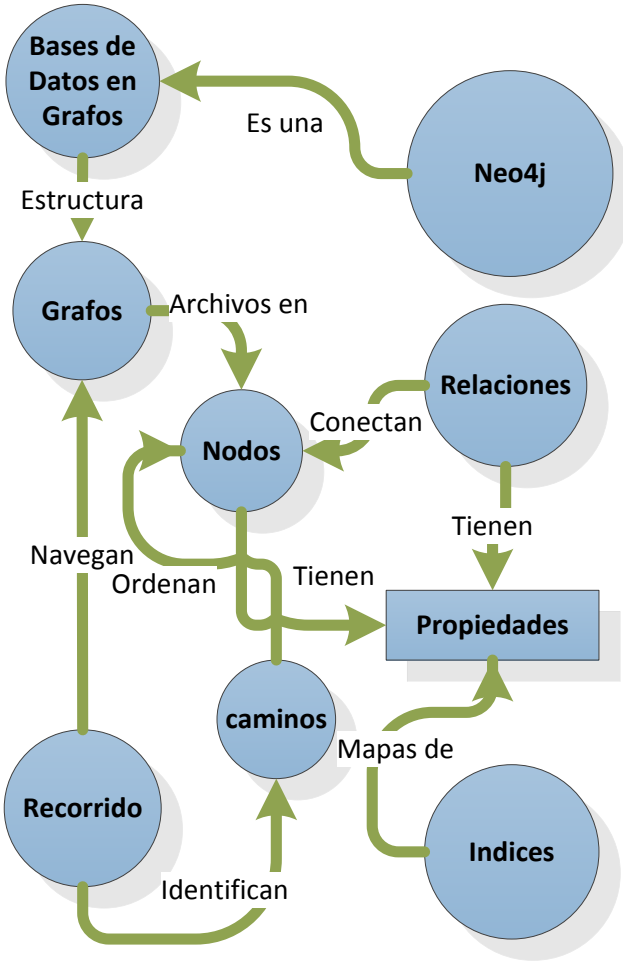

Figura 8. Modelo Neo4j

Características Principales de Neo4j

- Intuitivo: Usa un modelo de datos gráfico para la representación de datos.
- Confiable: Transacciones ACID completas.
- Durable y Rápido: Motores de almacenamiento nativo.
- Altamente escalable: Miles de nodos, relaciones y propiedades.
- Alta disponibilidad: Distribución a través de varias máquinas.
- Expresivo: Potente lenguaje de consulta.
- Embebido
- Simple: Accesible con una interface REST o una API orientada a objetos (JAVA).

Debido al énfasis en las consultas globales, los motores son normalmente optimizados para la digitalización y procesamiento de grandes cantidades de información.

2.3.5 Modelamiento de Datos en Grafos

Durante varias décadas, los desarrolladores han tratado de acomodar y semi-estructurar un conjunto de datos dentro de las bases de datos relacionales. Pero mientras que las bases de datos relacionales se diseñaron inicialmente para codificar los formularios en papel y tabular estructuras - algo que hacen en extremo bien en su lucha de intentar modelar las relaciones. Irónicamente, las bases de datos relacionales se tratan mal con las relaciones, debido a que se necesita unir las tablas para poder especificar las relaciones que existen entre ellas, la cual provoca la multiplicación de datos y la estructura general se vuelve más compleja y menos uniforme, por tanto; el modelo relacional se convierte en una carga al unir tablas. [5]

Por otro lado, las bases de datos relacionales tienen un nivel de abstracción lógico, con estructura de datos relacional y enfoque de la información mediante atributos, pero el aumento de las conexiones se traduce en el mundo relacional como un incremento de uniones, la cual impiden el rendimiento y hacen que sea difícil evolucionar una base de datos existente en respuesta a las necesidades empresariales del momento.

En comparación a las bases de datos relacionales se tiene también las bases de datos NoSQL, ya sea clave-valor-documento, o columna-orientado-almacenamiento estas se colocan de manera separa documentos/valores/columnas. Esto hace que sea difícil conectar datos y grafos en estas bases, una estrategia bien conocida para agregar relaciones es el uso de claves foráneas. Pero esto requiere hacer unión de tablas a nivel de aplicación, el cual se convierte rápidamente muy costoso cuando los datos se encuentran de manera distribuida y a su vez se tenga demasiada información en una consulta.

2.3.5.1 Cypher

Las bases de datos orientadas a grafos tienen un nivel de abstracción de manera lógica/usuario y el enfoque de información mediante relaciones. Como podemos observar claramente en la **(Figura 9)**, un esquema de usuarios con relaciones, estas pueden ser: amigos, colegas, jefe, amor, entre otras. De esta manera podemos establecer relaciones entre todos los usuarios tan solo teniendo un nuevo enlace.

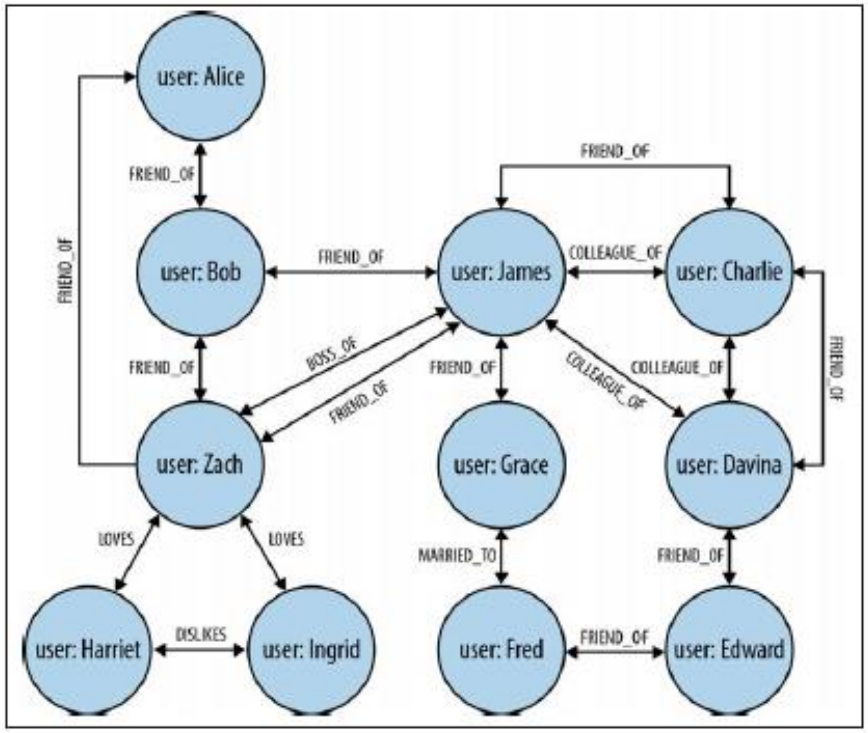

Figura 9. Modelamiento en Grafos [5]

Hasta ahora podemos observar todo de manera gráfica, pero como toda tecnología que vaya a ser aplicada sobre una base de datos necesitamos un mecanismo que permita manipular, consultar y crear datos, para esto se ha implementado el lenguaje CYPHER.

Cypher es un lenguaje de consulta gráfica diseñado de una manera que pueda ser fácilmente entendible por los desarrolladores y profesionales de bases de datos. Su facilidad de uso se deriva del hecho que se concuerda con la forma en que intuitivamente se describen gráficos usando simplemente diagramas. Como lo podemos observar en la **(Figura 10)**, en donde un nodo conoce a otro mediante la relación "Conoce".

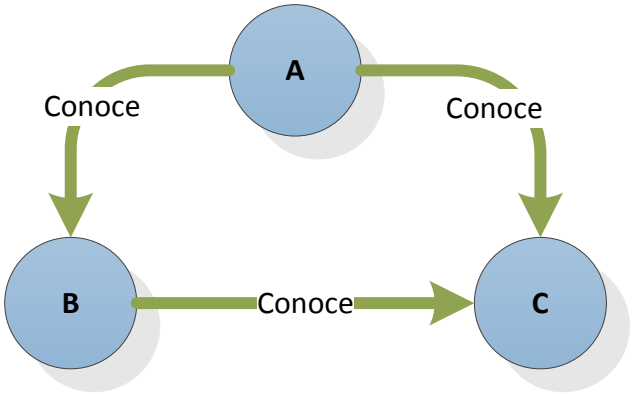

Figura 10. Expresando el uso de diagramas en un grafo

Como todo lenguaje de consulta, cypher está compuesto por clausulas, las más comunes para poder armar unos simples queries consiste de la cláusula para empezar una consulta *START*, seguido por *MATCH* y terminando con *RETURN*. Aquí un ejemplo de Cypher, el cual se usa para encontrar a los amigos de Michael, asumiendo que *Michael* se encuentra almacenado en el nodo "A".

START A=node:user(name='Michael') MATCH (A)-[:Conoce]->(B)-[:Conoce]->(C),(A)-[:Conoce]->(C) RETURN B,C

El uso de cypher pretende ser muy simple para ayudar a los desarrolladores cuando se trabaje con bases de datos gráficas, pero a su vez que sea diferente para saber que se trata de un conjunto no relacional.

A continuación se presentará las cláusulas básicas que Cypher usa para realizar consultas a una base de datos orientadas a Grafos en Neo4j.

- ***START:*** Especifica uno o más puntos de partida, que pueden ser nodos o relaciones en un grafo, estos puntos de partida son obtenidos a través de los índices, aunque también se pueden acceder directamente a los nodos y relaciones a través de sus identificadores.
- ***MATCH:*** Esta cláusula utiliza caracteres ASCII para representar los nodos y las relaciones, en ella expresamos la data que nos interesa en el grafo, utilizando paréntesis para representar a los nodos y flechas para representar las relaciones entre ellos.
- ***RETURN:*** Esta cláusula especifica que nodos, relaciones y propiedades del *MATCH* deben ser devueltos a los clientes.
- ***WHERE:*** Proporciona criterios para filtrar las búsquedas.
- ***CREATE*****:** Crea nodos y relaciones
- ***DELETE*****:** Remueve nodos, relaciones y propiedades.
- ***SET:*** Coloca valores a las propiedades

- **UNION:** Une los resultados de dos o más consultas.

En la **figura 11**, se muestra un modelamiento de usuarios como empleados de un ingeniero.

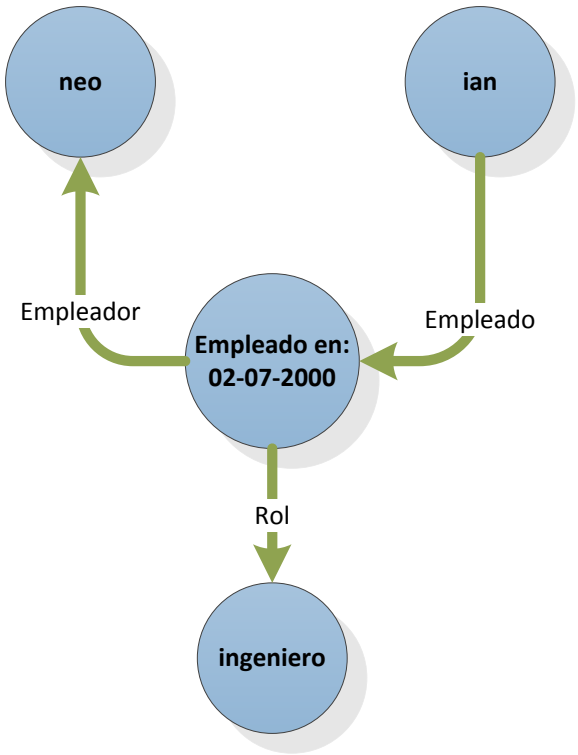

Figura 11. ian empleado de neo

Teniendo en consideración la **(Figura 11)**, su representación en cypher de "ian empleado de neo" sería:

(ian)-[:Empleado]->(Empleado en)-[:Empleador]->(neo),(Emplado en)-[:Rol]->(ingeniero)

2.3.6 Uso de BDOG en el mundo real

Cuando se trata de aplicaciones de bases de datos para solucionar un problema del mundo real, se presentan ciertas limitaciones de técnicas y negocios al querer usarlas. Por tanto, el rendimiento de consultas y capacidad de respuesta son parte de las preocupaciones de muchas organizaciones con respecto a sus plataformas de datos. Sistemas transaccionales en línea, aplicaciones web grandes en particular, debe responder a los usuarios finales en milisegundos para que puedan ser exitosas.

Las aplicaciones exitosas raramente permanecen tal cual como fueron creadas al inicio, los cambios se van dando dependiendo del comportamiento de los usuarios, esto ha requerido las organizaciones tengan mucho cuidado al realizar las migraciones

de datos, la naturaleza de las bases de datos sin esquema grafico podrían ocasionar problemas al realizar estas actualizaciones, pero una solución gráfica permite que los datos evolucionen a medida que evoluciona el negocio, permitiendo así reducir los riesgos y el tiempo de lanzamiento al mercado. Actualmente las bases de datos orientadas a grafos se están usando en los siguientes casos:

- *SOCIAL*

  Estamos empezando a descubrir el poder de los datos sociales, tal como dice en el libro de Nicholas Christakis y James Fowler, ellos muestran cómo, a pesar de no saber nada acerca de un individuo, se puede predecir mejor el comportamiento de esa persona por la comprensión de con quién está conectado, de lo que podríamos saber por la acumulación de hechos históricos acerca de él. [9]

  Un ejemplo de bases de datos gráficas es la aplicación de Facebook que usa los modelos de datos gráficos que permiten un paso natural del dominio centrado en las relaciones, debido a que las redes sociales nos ayudan a identificar las relaciones directas e indirectas entre las personas, los grupos y las cosas con las que interactúan, lo que permite a los usuarios valorar, revisar y describir uno a uno las cosas que le preocupan. Al entender quién interactúa con quién, cómo la gente puede estar conectada para así saber las afinidades que se pueden tener en común con otras personas.

- *GEO*

  Geo es el caso original del uso de datos en grafos: Euler resolvió los siete puentes de Königsberg problema postulando un teorema matemático que más tarde llegó a ser la base de la teoría de grafos. Aplicaciones geoespaciales de las bases de datos orientadas a grafos van desde el cálculo de rutas entre ubicaciones en una red abstracta como una red de carreteras y ferroviaria, la red del espacio aéreo, o red logística para operaciones espaciales, tales como encontrar todos los puntos de interés en un área limitada, encontrar el centro de una región, y calcular la intersección entre dos o más regiones.

  Operaciones geoespaciales dependen de estructuras de datos específicos, que van desde las relaciones ponderadas y dirigidos simples, a través de índices espaciales, como el de los Trees, que representan propiedades multidimensionales utilizando estructuras de datos de control de árboles. Como índices, estas estructuras de datos, naturalmente, tienen la forma de un gráfico,

por lo general en forma jerárquica, y como tales, son una buena opción para una base de datos gráfica. Debido a la naturaleza sin esquema de las bases de datos orientadas a grafos, los datos geoespaciales pueden residir en la base de datos junto a otros tipos de datos de la red social, por ejemplo, lo que permite consultas complejas multidimensionales a través de varios dominios.

Aplicaciones geoespaciales de las bases de datos orientadas a grafos son especialmente relevantes en las áreas de telecomunicaciones, logística, viajes, horarios y planificación de rutas.

- *REDES Y CENTRO DE GESTIÓN DE DATOS*

Una representación gráfica de una red nos permite catalogar los activos, visualizar la forma en que se implementan, e identificar las dependencias entre ellos. Estructuras conectadas en forma gráfica, junto con un lenguaje de consulta como Cypher, nos permiten llevar a cabo sofisticados análisis de impacto, respondiendo a preguntas tales como:

  - ¿De qué partes de la red dependen las aplicaciones, servicios, máquinas virtuales, las máquinas físicas, centros de datos, routers, switches y fibra-do?
  - Las aplicaciones y los servicios, y en última instancia, los clientes, en la red se verán afectados si en una red en particular alguno de sus elementos, ya sea un router o switch llegase a fallar.
  - ¿Hay redundancia en toda la red para los clientes más importantes?

Soluciones de bases de datos gráficas complementan las herramientas de gestión y análisis de redes existentes. Al igual que con la gestión de datos maestros, que pueden ser utilizados para reunir los datos de los sistemas de inventario, proporcionando una visión única de la red y su consumo, desde el elemento de red más pequeña de todo el camino a la aplicación, los servicios y los clientes que la utilizan. Una representación gráfica de la base de datos de la red también se puede usar para enriquecer la inteligencia operativa basado en correlaciones de eventos: siempre que un motor de correlación de eventos infiere un evento complejo de una corriente de eventos de red de bajo nivel, puede evaluar el impacto de ese evento con el modelo gráfico, y posteriormente, desencadenar cualquier compensación necesaria o acciones de mitigación.

Hoy en día, las bases de datos orientadas a grafos se están empleando con éxito en las áreas de telecomunicaciones, la gestión y el análisis de redes,

gestión de plataforma en la nube, centro de datos, la gestión de activos de TI, y el análisis de impacto en la red.

- *CONTROLES DE ACCESO (COMUNICACIONES)*

Información de soluciones de autorización y control de acceso a ciertas aplicaciones por parte de usuarios (por ejemplo, administradores, unidades organizativas, usuarios finales) y los recursos (por ejemplo, archivos, recursos compartidos, dispositivos de red, productos, servicios, convenios), junto con las normas que rigen el acceso a esos recursos; que luego se aplican estas reglas para determinar quién puede acceder o manipular un recurso. El control de acceso que tradicionalmente se ha aplicado ya sea usando los servicios de directorio o mediante la construcción de una solución personalizada dentro del backend de la aplicación. Estas estructuras de directorios jerárquicos, no pueden hacer frente a las estructuras de dependencia de recursos y organización no jerárquicas que caracterizan a las cadenas de suministro distribuidas multipartidistas. Soluciones liadas a mano, especialmente las desarrolladas en una base de datos relacional, sufren al querer unirse cuando el tamaño del conjunto de datos crece, lo que provoca que cada vez sea más lento y no responda a las consultas, con esto, se tendría una experiencia de usuario final pobre.

- *SOFTWARE DE RECOMENDACIÓN*

Recomendaciones efectivas son un excelente ejemplo de la generación de valores para el usuario final a través de la aplicación de una capacidad inferencial o sugestiva. Mientras que las aplicaciones de línea de negocio suelen aplicar los algoritmos de cálculo de nómina deductivo y preciso, aplicando impuestos, y así sucesivamente generar valor para el usuario final, los algoritmos de recomendación son inductivos y sugerentes, personas que identifican, productos o servicios de una persona o grupo. Los algoritmos de recomendación establecen relaciones entre las personas y las cosas: la gente, los productos, los servicios, los medios de comunicación el contenido que sea relevante para el dominio en el que se emplea la recomendación. Las relaciones se establecen en base a los comportamientos de los usuarios, ya sea comprar, producir, consumir, habilitación o la revisión de los recursos en cuestión. El motor puede identificar recursos de interés para un individuo en particular o grupos de individuos que puedan tener algún interés en un recurso en particular.

2.4 Sistemas de Recomendación

A menudo a lo largo del día nos vemos involucrados en tomar ciertas decisiones que afectan nuestras vidas, indistintamente del valor que estas posean nos vemos inmiscuidos en tener que decidir, a pesar de que no tenemos la información necesaria o suficiente experiencia personal sobre el tema. Por lo general, nos basamos en las recomendaciones de otras personas (a veces conocidos o amigos) o bien por el boca a boca que se posee de los medios de comunicación o guías generales.

Los sistemas de recomendación tratan de hacer esto de forma automática, dando asistencia a los usuarios y así aumentar el proceso natural de proveer "recomendaciones". Otra definición amplia que define a los sistemas de recomendación como: "herramientas de software y técnicas que ofrecen sugerencias de temas a ser de utilidad para un usuario". [10]

Estos sistemas se centran en hacer recomendaciones a usuarios de ítems en específicos, tales como: libros, películas, músicas entre otras. Lo que se pretende con este trabajo es implementar una recomendación efectiva y precisa que se centre en dar una sugerencia cuando el usuario más lo requiera, ofreciendo recomendaciones entre dominios de elementos que pertenecen a varias categorías o dominios.

Los sistemas de recomendación se dirigen principalmente a las personas que carecen de experiencia personal o competencia para evaluar el número abrumador de artículos ofrecidos por los sitios web, aplicaciones móviles o plataformas de comercio electrónico. [11]

2.4.1 Vista General

Los Sistemas de recomendación emergen como una iniciativa de investigación independiente a mediados de 1990, [12] [13] [14] Goldberg, Resnick y Shardanand probaron que está era una poderosa herramienta para ayudar a los usuarios a encontrar contenidos, productos o servicios acorde a sus necesidades, estos sistemas usan tecnologías de análisis para generar recomendaciones personalizadas que son ofrecidas a los usuarios como una lista de elementos.

Al inicio de esta investigación las recomendaciones estaban limitadas a solo películas y compras, desde el año 2007 este tema se empezó a extender a otras listas de interés como: libros, documentos o música [15], siendo este uno de los más importantes avances en el filtrado de información colaborativa. Esto era de esperarse debido a que los sistemas de recomendación han sido tradicionalmente aplicados al ámbito comercial, debido a que conducen al incremento de las compras por parte de los clientes.

- *FUNCIONES DE LOS PROVEEDORES DE SERVICIOS*

Podemos referirnos como proveedores de servicios a plataformas comerciales como: Amazon[1] o Netflix[2] que se encargan de promocionar sus productos (libros y películas) a los usuarios mediante estos sistemas de recomendación o también los podemos encontrar en las redes sociales como: Facebook[3] y Twitter[4], estás principalmente recomiendan una nueva amistad en la red.

Sin importar el tipo de proveedor de servicios, las razones para integrar los sistemas de recomendación en sus entornos se pueden describir por al menos una o más de las siguientes funciones [16] [11]:

- *Incrementar el número de ítems vendidos.* Esta es probablemente la función más importante de los sistemas de recomendación, porque de esta manera los usuarios descubren nuevos "ítems" que pueden ser comparados al instante. En general, desde el punto de vista de los proveedores de servicios la meta principal de introducir los sistemas de recomendación es incrementar la tasa de usuarios que aceptan las recomendaciones y consumen los "ítems" de los que al contrario llegan a la página navegando a través de la web.
- *Vender una variedad de artículos diferentes.* Los sistemas de recomendación ayudan a seleccionar artículos que podrían ser difíciles de encontrar sin la precisión de la recomendación. Por ejemplo, en un sistema de recomendación de libros como Amazon, es inútil recomendar sólo los libros más vendidos. El proveedor de servicios necesita hacer publicidad a los usuarios de todo el catálogo, pero la mejor manera de hacerlo sin dar el riesgo de que se sugiera un libro no apropiado para un usuario es utilizando un sistema de recomendación que genere recomendaciones personalizadas a los usuarios adecuados.
- *Incrementar la satisfacción de los usuarios.* Al ayudar a los usuarios a encontrar los artículos que necesitan en términos de relevancia o exactitud, unos sistemas de recomendación pueden aumentar el uso y la probabilidad de que se acepte una recomendación. Por otra parte, si la

[1] www.amazon.com
[2] www. netflix.com
[3] www. facebook.com
[4] www.twitter.com

experiencia del usuario es buena debido a una correcta interacción humano-computador, también podrán disfrutar de utilizar el sistema.

- *Incrementar la fidelidad del usuario.* Cuanto más tiempo el usuario interactúa con el sitio es más probable fomentar la lealtad al sitio.
- *Incrementar la personalización*. Como consecuencia de todas las funciones anteriores, el sistema entiende mejor lo que el usuario quiere o necesita, aumentando de esa manera el nivel de personalización no sólo en las recomendaciones generadas, sino también en la forma en que el sitio trata al usuario.

- *FUNCIONES PARA LOS USUARIOS*

Herlocker, [17] define un conjunto de metas y tareas que los sistemas de recomendación pueden implementar para ayudar a los usuarios a tener una mejor experiencia, tales como [11]:

- *Anotaciones en contexto.* Dada una situación existente, seleccionar sólo aquellos elementos que son adecuados para el contexto del usuario. Por ejemplo, en una recomendación de restaurantes, considere la ubicación del usuario como información de contexto para recomendar solo restaurantes que se encuentran a una distancia a pie.
- *Encontrar buenos artículos.* Proporcionar al usuario una lista de recomendaciones sobre los mejores artículos, entre todos los disponibles en el sistema, ordenadas de acuerdo a sus preferencias.
- *Encontrar todos los buenos artículos.* Consiste en recomendar todos los elementos disponibles en el sistema que son adecuados a las necesidades del usuario.
- *Recomendar una secuencia.* Se basa en la recomendación de los elementos de una manera secuencial, atendiendo a la idea de que son aptos para ser consumidos en un orden específico. Por ejemplo, si pensamos en un sistema centrado en la recomendación de los trabajos de investigación de un tema dado, podemos imaginar una secuencia de importantes documentos que se deben leer en un cierto orden para hacer más fácil la comprensión de la zona de estudio.
- *Recomendar un paquete.* Esta función es similar a la nombrada previamente, pero considerando que el orden no es relevante. Por ejemplo, un turista recomienda un punto de interés de una ciudad que se debería visitar.

- *Recomendar paquetes sinérgicos.* Los elementos que en conjunto tienen más utilidad para el usuario que recomendarlos por separado. Por ejemplo, tenedor, cuchillo y cuchara en un solo dominio llamado: cubiertos.
- *Navegadores Ocasionales.* A veces los usuarios visitan sitios cuando no tienen la necesidad de comprar, porque les resulta agradable, o simplemente por curiosidad. En esta situación, la tarea del recomendador es ayudar al usuario a navegar por los elementos que tienen mayor probabilidad de ser consumida por el usuario o que son parte de su ámbito de interés.
- *Recomendaciones creíbles.* Los usuarios con frecuencia no confían automáticamente en un sistema de recomendación. Es típico que muchos de ellos buscan recomendaciones que saben que les gusta antes de empezar a creer en el sistema (por ejemplo, sus libros favoritos en sitios como, Amazon). Por esa razón, los usuarios pueden cambiar sus perfiles con el fin de probar el recomendador. Una tarea interesante sería dejar que los usuarios lo hagan para probar cómo se comporta el sistema. En otros casos, los usuarios no confían en el sistema debido a un posible nivel de sesgo en la fuente de recomendaciones. Por lo tanto, los sistemas de recomendación tienen que ofrecer confiabilidad mediante el uso de los datos adecuados.
- *Mejorar el perfil.* Los sistemas de recomendación debe ser capaces de recoger la opinión de los usuarios después de que hayan sido recomendados para mejorar sus perfiles. De lo contrario, sólo puede proporcionar a un usuario un destino con la misma recomendación que se entrega a un usuario "normal".
- *Expresar ideas y comentario.* Algunos usuarios no se preocupan por las recomendaciones en absoluto. Más bien, lo que es importante para ellos es que se les permita expresar sus opiniones y sentimientos. Los sistemas de recomendación puede aprovechar eso para animar a los usuarios, proporcionándoles métodos de retroalimentación con el fin de recuperar la información que mejorará la calidad de las recomendaciones futuras.
- *Ayudar a los demás*. Algunos usuarios de Internet están dispuestos a contribuir con la información (p. ej., las evaluaciones de artículos), porque creen que la comunidad y el sistema de recomendación en sí se benefician de su contribución. Por ejemplo, en una plataforma de e-learning, un profesor puede ayudar a sus colegas en mostrar la forma de

aprendizaje que utiliza para facilitar que la comunidad distinga entre los recursos pedagógicos ricos y pobres.

- *Influir en los demás.* En los sistemas de recomendación, hay usuarios cuyo objetivo principal es influir explícitamente a otros a comprar o consumir un conjunto de elementos específicos. Tratando de evitar esta práctica, o al menos mitigarlo, es una de las tareas más importantes en cualquier sistema de recomendación.

2.4.2 Elementos de un Sistema de Recomendación

Los sistemas de recomendación son conocidos por el procesamiento complejo de información y las herramientas de filtrado con entradas heterogéneas de información y finalmente generar una recomendación efectiva. El código de los datos puede ser muy diverso debido a los distintos escenarios que el sistema de recomendación debe atender. En la **(Figura 12)** se presenta un esquema de los datos utilizados en los sistemas de recomendación, estos se refieren a tres tipos de objetos que estarán involucrados en el proceso: artículos, usuarios y transacciones.

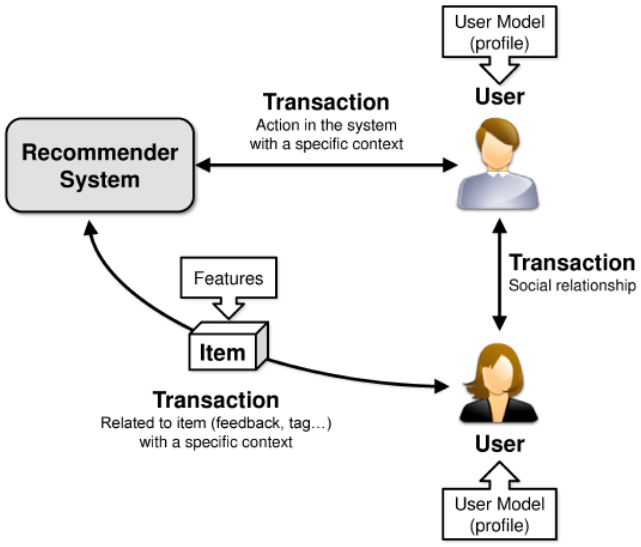

Figura 12. Relación general entre artículos, usuario y transacciones [11]

- *ATICULOS*

Son los elementos recomendados a los usuarios, estos pueden ser caracterizados por el valor que posean. El valor de un elemento puede ser positivo si el artículo es útil, o negativo si su idoneidad determina que se trata de una decisión equivocada para el usuario cuando lo selecciona. Aparte de esto, el valor de un elemento puede estar relacionado con el coste de su adquisición, que incluye no sólo un coste monetario real que es

finalmente pagado, sino también el costo cognitivo de la búsqueda o de decidir si aceptarla o rechazarla [11].

- *USUARIOS*

Los usuarios en un sistema de recomendación pueden tener diversos objetivos y características basadas en el escenario de la recomendación. Con esto se tiene una gama de información acerca de los usuarios que pueden ser explotadas para aumentar el nivel de personalización de las recomendaciones. La selección de la información que se considera para construir el modelo del sistema depende de la técnica de recomendación elegida, pero como regla general, el modelo del sistema se debe centrar en el perfil del usuario, es decir, codificar todas las preferencias, necesidades y el comportamiento, dado que ninguna personalización es posible sin un modelo de usuario conveniente.

- *TRANSACCIONES*

Una transacción puede ser definida como información grabada entre un usuario y los sistemas de recomendación, en los que pueden implicar, además de un elemento, otros usuarios **(Figura 12).** Esta información se genera durante la interacción humano-ordenador y puede ser útil para el algoritmo de recomendación que se esté utilizando.

Por ejemplo, un registro de transacciones puede contener una referencia a un elemento seleccionado por el usuario y una descripción del contexto (e. g., ubicación de usuario) en el que esa situación en particular se llevó a cabo. Además de esto, también puede incluir una retroalimentación explícita proporcionada por el usuario sobre el artículo, permitiendo que el sistema sea adaptable. Un buen ejemplo de retroalimentación serían las calificaciones otorgadas por el usuario para un elemento seleccionado.

Las calificaciones son la forma más común y popular de la transacción. Ellos pueden ser recogidos de forma explícita (p. ej., el usuario presta su opinión sobre un artículo) o implícitamente (p. ej., el recomendador anota si el usuario acepta o rechaza la recomendación). De acuerdo con la clasificación propuesta por Schafer [18], las calificaciones pueden tomar una variedad de formas (los más comunes se presentan en la **Figura 13**):

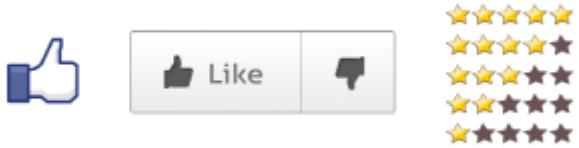

Figura 13. Ejemplos del uso común de calificaciones en la web [11]

- *Calificaciones Unarias.* Estas calificaciones pueden indicar que un usuario ha observado o comprado un artículo, y lo ha marcado de forma positiva. La ausencia de calificación indica que no tenemos ninguna información sobre el usuario con respecto al elemento. Un buen ejemplo de este tipo de calificación ha sido popularizado por Facebook[5] a través del botón "Me gusta".
- *Calificaciones Binarias.* Indican que las elecciones del usuario pueden estar entre el bien o el mal cuando evalúan un ítem. YouTube[6] utiliza esta opción para permitir a los usuarios calificar los videos de forma positiva ("me gusta") o negativa ("No me gusta") con una representación visual de íconos con un pulgar hacia arriba y otro hacia abajo.
- *Calificaciones Escalares.* Consisten en cualquier calificación numérica, como las cinco estrellas de Netfilx[7] o el control deslizante, así como las calificaciones ordinales como: muy de acuerdo, de acuerdo, neutral, en desacuerdo, muy en desacuerdo, etc.

Otra forma de transacciones que los usuarios usan son los llamados “tags”, estos son etiquetas que los usuario utilizan para nombrar algo, utilizando esto se puede generar una recomendación más eficiente.

2.4.3 Técnicas de Recomendación

La funcionalidad básica de un sistema de recomendación consiste en identificar los elementos útiles para el usuario entre los ya existentes. Por lo tanto, se debe predecir la utilidad de algunos de ellos para decidir por comparación, ¿cuál de ellos vale la pena recomendar?.

Esta predicción no es un paso sencillo, y existe así diferentes métodos y algoritmos para calcularlo. Como resultado, se han propuesto varios tipos de sistemas de recomendación desde que aparecieron los primeros sistemas de filtrado de

[5] www.facebook.com
[6] www.youtube.com
[7] www.netflix.com

colaboración. Los principales se describen en las siguientes líneas que asisten a la taxonomía proporcionada por Burke [19], que se ha convertido en una forma clásica de distinguir entre los sistemas de recomendación y a lo que se refieren cada uno de ellos. [11]

- *FILTRADO COLABORATIVO*

  Estos sistemas de recomendación presentan elementos que les han gustado a otros usuarios con gustos similares y así calcular la similitud entre usuarios. Para lograr tener un gran desempeño los usuarios deben realizar una evaluación previa sobre algunos elementos, de esta forma permiten al sistema formar el perfil del usuario.

  En el filtrado colaborativo, el sistema no analiza los elementos evaluados, sino que las recomendaciones se basan solamente en la similitud entre usuarios. Esto trae consigo algunos problemas, como se comenta a continuación:

  Cuando un usuario llega al sistema, no es posible hacerle recomendaciones hasta que su perfil sea lo suficientemente completo para encontrarle a su grupo de vecinos cercanos. Además si los gustos del usuario son poco comunes, encontrarle un conjunto de vecinos cercanos será una tarea complicada. Esto hace notar que las recomendaciones dependan directamente del número y variedad de usuarios en el sistema.

- *BASADO EN CONTEXTO*

  Estos sistemas recomiendan artículos similares a un usuario en base a una descripción del elemento y un perfil de los intereses del usuario. [20] La similitud de los elementos se calcula en base a las características asociadas a los elementos comparados y a los antiguos que al usuario le han gustado. Por ejemplo, si Bob tiene valores positivos de la película "Iron Man", que pertenece al género de "acción", entonces el sistema puede aprender a recomendar otras películas del mismo género. Sin embargo, esta técnica tiene algunos problemas importantes que deben tenerse en cuenta cuando se la utiliza:

  - *Análisis de Contenido Limitado.* Las técnicas basadas en contenido están limitadas por las características que se asocian de manera explícita a los objetos recomendados por el sistema.

- *Over-specialization.* El sistema solo permite hacer las recomendaciones acorde al perfil del usuario, debido a que estos se encuentran limitados a solo recomendar "ítems" similares a los que tenga el usuario en su perfil.
- *Problemas de nuevos Usuarios.* Al igual que el caso de filtrado de contenido, se debe primeramente evaluar una serie de "items" antes de que el sistema aprenda las preferencias del usuario.

- *DEMOGRAFICO*

Estos sistemas de recomendación se basan en el perfil demográfico del usuario, es decir; las recomendaciones van siendo generadas por las diferentes zonas demográficas que impliquen el estilo de vida del usuario.

- *BASADO EN EL CONOCIMIENTO*

Los sistemas de recomendación basados en el conocimiento se basan en tener "items" específicos de un dominio de conocimiento acerca de ciertas características que satisfacen las necesidades y preferencias de los usuarios.

Estos sistemas tienden a funcionar mejor que otros re-comendadores al comienzo de la implementación debido al stock de conocimientos iniciales, pero estos deben aprender los comportamientos de los usuarios para ser útil a largo plazo. Por esa razón, los sistemas basados en el conocimiento tienen menos problemas en este sentido, ya que no tienden a confiar en los datos históricos sobre las preferencias del usuario.

- *BASADO EN COMUNIDAD/SOCIAL*

Sistemas de recomendación basados en la comunidad sugieren elementos en función de las preferencias de los amigos del usuario teniendo en cuenta el epigrama "Dime con quién andas y te diré quién eres". Las investigaciones han señalado que las personas tienden a confiar más en las recomendaciones de la gente de su confianza (amigos), que en las recomendaciones de individuos similares pero anónimos [11] [21].

Esta observación, junto con la creciente popularidad de las redes sociales está generando un interés en este tipo de sistemas, Por ejemplo, en una red

social como Facebook[8], es común que un usuario que se le recomiendan los contenidos o productos que a sus conocidos les gustan o han consumido con anterioridad. Las recomendaciones generadas de esta manera se basan en información proveniente de la red de confianza del usuario, es decir, una red social que expresa la cantidad de los miembros de la comunidad entre sí. Por lo tanto, podemos decir que los sistemas de recomendación de confianza, utilizan el conocimiento que se origina a partir de estas redes sociales para generar recomendaciones más personalizadas: los usuarios reciben recomendaciones de los artículos valorados muy positivamente por las personas en su red o por personas que gozan de la confianza de los miembros a través de las relaciones de un amigo-de-un-amigo.

- *HIBRIDO*

  Los sistemas de recomendación híbridos son aquellos que combinan dos o más de las técnicas descritas anteriormente para mejorar el rendimiento de las recomendaciones, por lo general para hacer frente al problema de arranque en frío. Al hacer esto, tratan de combinar las ventajas de varias técnicas y al mismo tiempo para evitar sus desventajas. [11]

2.4.4 Métodos Data Mining

Los sistemas de recomendación suelen aplicar técnicas utilizadas en la minería de datos para construir sus algoritmos, por lo que también es posible utilizarlos para la toma de decisiones, así como para calcular los patrones de similitud o predecir el efecto de las decisiones. Por esa razón la mayoría de los investigadores de sistemas de recomendación utilizan técnicas de minería de datos para mejorar el rendimiento y la precisión de sus recomendaciones.

Existen algunos modelos que usan técnicas para resolver problemas en DM, éstas dependen del objetivo buscado. Las técnicas se dividen en, técnicas *predictivas*, donde las variables pueden clasificarse en dependientes e independientes, y técnicas *descriptivas*, donde todas las variables tienen inicialmente el mismo status. [22]

Las técnicas predictivas requieren de un proceso de aprendizaje supervisado: la técnica supervisa en el modelo en construcción el grado de ajuste a la realidad conocida. En este sentido, dichos modelos pretenden estimar valores futuros o desconocidos de una variable respuesta: cuando es una variable respuesta categórica (para predecir etiquetas de clase), se conoce como un modelo de clasificación. Si el

[8] www.facebook.com

valor para ser predicho es numérico (variable respuesta continua) se llama modelo de regresión.

Las técnicas descriptivas se rigen por un proceso de aprendizaje no supervisado: su objetivo es identificar patrones en los datos sin indicadores externos que guíen al algoritmo (es decir, sin conocer la realidad "a priori"). De esta forma, las técnicas descriptivas, sirven para explotar las propiedades de los datos examinados. Entre las técnicas ampliamente utilizadas en los procesos de DM destacan: la regresión, agrupamiento o clustering, reglas de asociación, árboles de decisión, redes neuronales y algoritmos genéticos.

- El *análisis regresivo* es una técnica utilizada para inter y extrapolar las observaciones, que pueden clasificarse como regresión lineal o no lineal. Hablamos de modelo de regresión cuando la variable de respuesta y las variables explicativas son todas ellas cuantitativas. Si sólo disponemos de una variable explicativa hablamos de regresión simple, mientras que si disponemos de varias variables explicativas se trata de un problema de regresión múltiple.
- *Agrupamiento o Clustering* es un procedimiento que agrupa datos por su similitud en conjuntos bien cohesionados y definidos. Los conjuntos no están predefinidos, sino que se determinan a partir de los datos. Se utilizan con frecuencia en procesos de descripción de datos. Este método puede utilizarse de tanto de manera descriptiva como predictiva (a qué grupo pertenecerá un nuevo dato). Entre los algoritmos que utilizan agrupamiento tenemos: K-Means, K-Medoids.
- Las *reglas de asociación* se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos, generalmente orientadas al uso de datos nominales. Este análisis permite encontrar correlaciones o concurrencias en los sucesos de los datos a analizar y se formalizan en la obtención de reglas del tipo, si… entonces.
- Los *árboles de decisión* son modelos que tienen estructura de árbol que representan conjuntos de decisiones. Son decisiones que generan reglas para clasificar un conjunto de datos. Entre los algoritmos que aplican este modelo tenemos ID3 y C4.5.
- Una *red neuronal* es una de las técnicas que se puede usar para entrenar la red. Permite la construcción de un modelo a partir de una determinada cantidad de ejemplos. Los mapas auto-organizativos o también conocidos como Kohonen, son un algoritmo que aplica la técnica de red neuronal.

- Los *algoritmos genéticos* son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización, se basan en la evolución genética para resolver problemas.

2.5 Sistemas de Recomendación Sensibles al Contexto

En los sistemas de recomendación sensibles al contexto (SRSC), se analizan y se toman en consideración diferentes factores contextuales para lograr inferir el contexto que tenga en ese momento el usuario, debido a que se tiene que poder adaptar las recomendaciones a las circunstancias descritas. La información de contexto puede ser proporcionada por varios medios: web, móvil, etc.

Por tal motivo, incorporar "el contexto" en los sistemas de recomendación ayudan a sugerir temas a los usuarios, en determinadas circunstancias, o teniendo en cuenta su situación actual. Por ejemplo, utilizando el contexto temporal y la ubicación en un escenario de sistema de recomendación móvil, los usuarios podrían ser recomendados con los puntos más adecuados de cierta ciudad.

2.5.1 Definición de Contexto

Contexto es un término que deriva del vocablo latino *contextus* y que se refiere a todo aquello que rodea, ya sea física o simbólica de un acontecimiento. A partir del contexto se puede interpretar o entender un hecho.

El contexto está formado por una serie de circunstancias (como el tiempo y el espacio físico) que facilitan el entendimiento de un mensaje. Por ejemplo: un portal que publica un título como "*Carlos descansó*" no brinda los datos necesarios para que el lector logre decodificar el mensaje. En cambio, el titular "*Tras jugar cuatro partidos en dos días, el tenista Carlos López descansó y no se presentó a entrenar en el comienzo de la preparación para la Copa Davis*", puede ser interpretado sin problemas ya que presenta información relevante sobre el contexto. [23]

Como nos centramos en los sistemas de recomendación, vamos a revisar las diferentes definiciones utilizadas en los campos relacionados con la investigación de los sistemas de recomendación, atendiendo a la clasificación proporcionada por Adomavicius y Tuzhilin [24]:

- *DATAMINING.* Para la comunidad de Datamining, el contexto es definido por los eventos en la vida del cliente y se pueda determinar el cambio en las preferencias, estados y valores que afecten su cotidianidad. Por ejemplo, el contexto puede ser definido al tener un nuevo trabajo, nacimiento de un hijo, cambios en su estado civil, etc.

- *PERSONALIZACIÓN E-COMMERCE.* C. Palmisano, A. Tuzhilin y M. Gorgoglione, [25] definen el contexto como la intención de compra realizada por un cliente en una aplicación de comercio electrónico. Los diferentes intentos de compra pueden dar lugar a diferentes tipos de comportamientos. Por ejemplo, el mismo cliente puede comprar el mismo producto por diferentes razones (p. ej., un libro para sí mismo o un libro como regalo para un amigo). Dicha segmentación contextual de los clientes es muy útil, porque da lugar a mejores modelos de predicción a través de diferentes aplicaciones de comercio electrónico.
- *SISTEMAS DE CONTEXTO MÓVIL.* Estos sistemas principalmente fueron definidos por la posición del usuario, en donde, podamos obtener ciertos valores como: localización, temperatura, fecha, estación del año, etc. Este contexto es crucial en los sistemas móviles debido a que actualmente la mayoría de los sistemas utilizan esa información para poder realizar sus funciones.
- *DATABASES*. La capacidad de contexto ha sido agregado a algunos de los sistemas de bases de datos, al incorporar preferencias de usuarios y devolver diferentes respuestas dependiendo de los "queries". Siendo más precisos, se introduce un conjunto de parámetros contextuales y las preferencias se definen para cada combinación de atributos relacionales.
- *RECUPERACIÓN DE LA INFORMACIÓN*. La información contextual ha demostrado ser útil en la recuperación de información y su respectivo acceso [24], aunque la mayoría de los sistemas existentes basan sus decisiones de recuperación únicamente en consultas y colecciones de documentos. La búsqueda en la web es un buen ejemplo del uso de contexto en la recuperación de información, considerando temas potencialmente relacionadas con el término de búsqueda.
- *MARKETING Y GESTIÓN.* Este contexto es asociado a la situación en donde se realiza una transacción, ya que influye en el cliente a la hora de tomar una determinada decisión de seleccionar un producto/marca. Actualmente se distinguen tres tipos: temporal (entregar la experiencia del cliente), espacial (a dónde puede enviar) y tecnológico (la forma de entregar datos).

2.5.2 Modelando Información Contextual en Sistemas de Recomendación

El proceso de recomendación suele comenzar con las especificaciones que le proporciona explícitamente el usuario o implícitamente el sistema, cuando estas calificaciones se han especificado, el sistema trata de estimar la relación entre los ítems con los usuarios, mediante la siguiente ecuación:

$$R: Usuario\ x\ Articulo \rightarrow\ Puntuación \tag{2.1}$$

En donde, el Rating es un conjunto totalmente ordenado (por ejemplo, números enteros no negativos o números reales dentro de un cierto rango), cuando la función R se calcula se le permite a un sistema de recomendación sugerir los elementos de mayor audiencia para cada usuario. Estos sistemas de recomendación son conocidos como tradicional o de dos dimensiones (2D) ya que consideran sólo las dimensiones de usuario y del item en el proceso de recomendación. En otras palabras, el problema se reduce a las recomendaciones de los artículos que no han sido revisados por el usuario.

Por otro lado, SRSC incorpora información contextual en el proceso de la recomendación para mejorar la predicción de los gustos de los usuarios. En consecuencia, el modelo SRSC considera no sólo las dimensiones del usuario y del artículo, sino también del contexto:

$$R: Usuario\ x\ Articulo\ x\ Contexto \rightarrow\ Puntuación \tag{2.2}$$

Desde este punto de vista, el Contexto se puede modelar como un conjunto predefinido de parámetros contextuales relacionados con una aplicación determinada. Tras Palmisano y col. [25] y Adomavicius [26], dicen que la información contextual puede ser definida por un conjunto de dimensiones contextuales, cada uno de ellos siguiendo una estructura jerárquica en función del nivel de granularidad relacionada con la dimensión contextual en estudio.

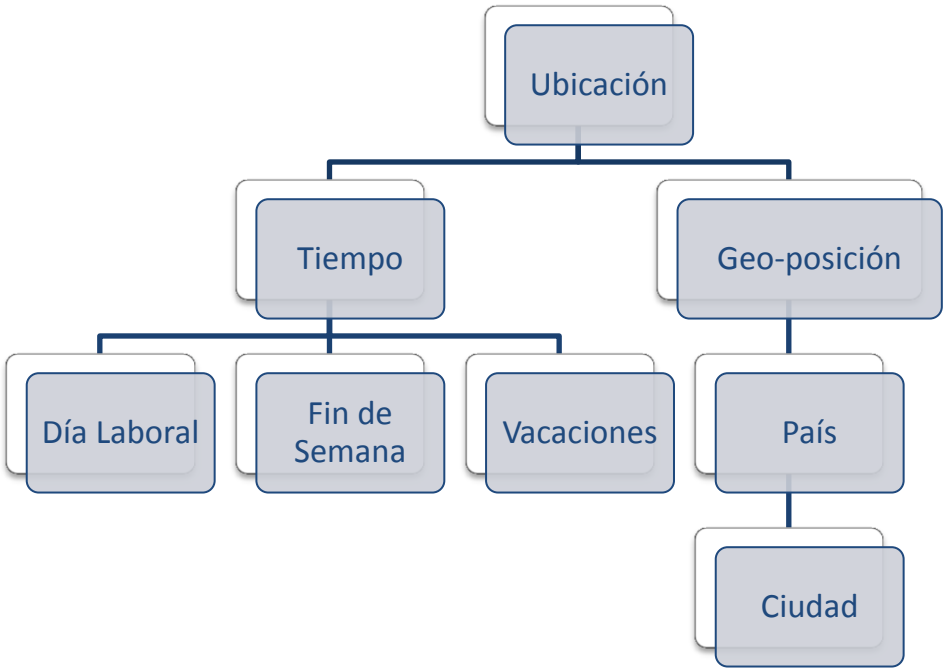

Figura 14. Información Contextual - Estructura Jerárquica para la ubicación de contexto

Por ejemplo, la **(Figura 14)** ilustra un ejemplo sencillo de la estructura relacionada con la dimensión contextual "ubicación", que se puede entender como la combinación de tiempo e información de geo-posición, siendo a su vez cada uno de ellos compuesto por varios atributos que representan la información que podría ser extraída en una situación contextual específica.

Por supuesto, este enfoque multidimensional para el modelado de contexto es una alternativa más de modelado, pero el modelo dependerá de cada escenario en el que el sistema de recomendación posea información contextual relevante o útil que deba ser identificada.

2.5.3 Obtener Información Contextual

La forma en que se obtiene la información contextual dependerá principalmente del diseño del sistema de recomendación, pero algunos métodos ya han sido identificados para su uso [24]:

- *Explicita.* Consiste en acercarse directamente a las personas y otras fuentes de información contextual para recopilar los datos, ya sea por medio de preguntas directas o solicitar esta información a través de otros medios. Por ejemplo, un sistema de recomendación "restaurante" puede pedir al usuario una dirección o zona determinada con el fin de sugerir sólo los restaurantes más cerca de él.
- *Implícita.* Consiste en la extracción de los datos de contexto desde el entorno de recomendación, por lo que este proceso sería transparente para el usuario. Tras el último ejemplo, en un sistema de recomendación "restaurante móvil", la aplicación podría utilizar las capacidades de dispositivo GPS para detectar la ubicación actual del usuario en términos de su posición geográfica.
- *Inferido.* Se basa en el uso de métodos de minería de datos. Por ejemplo, la identidad de las personas en un hogar al cambiar los canales de televisión (p. ej., esposo, esposa, hijo, hija, etc.), pueda que no sepamos de forma explícita la empresa de televisión, pero si podríamos inferir con razonable precisión utilizando el histórico de los programas vistos. Para ello, es necesario el diseño de un modelo de predicción (es decir, un clasificador) que permita entrenar el sistema con datos de semillas adecuados.

2.5.4 Paradigmas para incorporar Contexto

En el apartado 2.5.2, hemos visto que el sistema de recomendación tradicional se basa en los datos de entrada representados por los registros <usuario, artículo, puntuación>. Se ilustra gráficamente en la **(Figura 15)**. Esto demuestra que la entrada

general de datos está compuesta por la información sobre los usuarios, elementos y valoraciones que se procesa en un recomendador 2D para predecir las calificaciones de los elementos de un usuario. [11]

Figura 15. Componentes en un Sistema de Recomendación Tradicional [11]

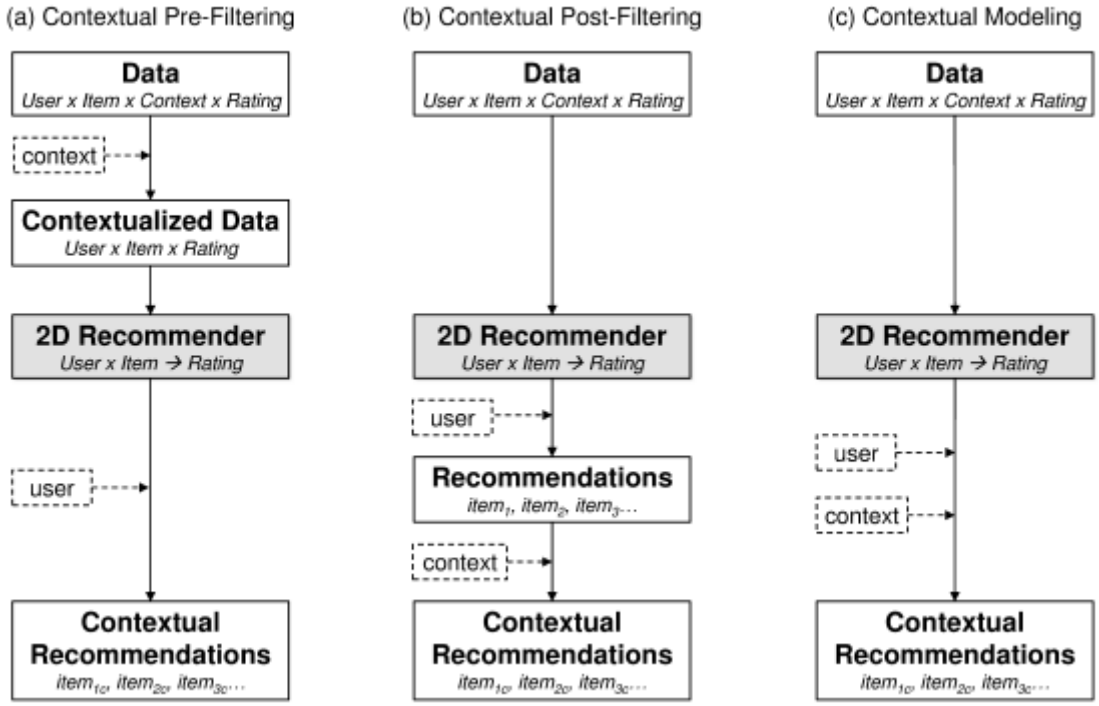

Figura 16. Paradigmas para incorporar "Contexto" en un Sistema de Recomendación. [11]

Pero, la introducción de contexto como una nueva entrada de datos en el proceso de recomendación puede mejorar el nivel de personalización en las recomendaciones al usuario. A continuación se presentan tres paradigmas algorítmicos diferentes para la incorporación de información contextual en un sistema de recomendación que se los resume en la **(Figura 16)**. [24]

2.5.4.1 Pre-filtrado Contextual

Este paradigma se basa en contextualizar la entrada de recomendación mediante el uso de la información contextual disponible para seleccionar los datos más relevantes en un contexto específico. Después de eso, cualquier recomendador 2D tradicional puede ser utilizado para generar la recomendación final a un usuario. En otras

palabras, este enfoque puede ser visto esencialmente como una consulta para la selección o filtrado de los elementos "*candidatos*" en la que sus calificaciones se ajusten a las condiciones del contexto.

Un ejemplo de una recomendación pre-filtrado contextual en una recomendación de "*restaurante*" sería: si una persona quiere un restaurante para cenar en Madrid, sólo los restaurantes mejores valorados por la noche en Madrid se utilizan en el proceso de recomendación.

Una ventaja importante de este enfoque es que se puede implementar en cualquier sistema de recomendación tradicional, debido a que el proceso de pre-filtrado no altera los datos de entrada necesarios para esas técnicas.

2.5.4.2 Post-filtrado Contextual

En este paradigma la información contextual se ignora inicialmente, y las calificaciones se predicen utilizando cualquier sistema de recomendación tradicional. Luego, la lista de los N elementos inicialmente recomendados por el recomendador 2D son los que se filtran para asistir a la información contextual con el fin de aumentar el nivel de personalización del usuario. Este ajuste de las listas se puede hacer de dos maneras:

- Filtrando las recomendaciones que son irrelevantes en un contexto dado. Por ejemplo, en el caso del "*restaurante*", la recomendación anterior genera una lista de los artículos proporcionados por el recomendador 2D, esta puede contener restaurantes de otras ciudades y, por tanto, el proceso de filtrado sería eliminarlos de la lista final que se le sugiere al usuario.
- Clasificación de las recomendaciones de la lista en base a un contexto dado. Siguiendo con el mismo ejemplo, esta vez la recomendación debe reordenar los elementos para mostrar primero aquellos que se ajustan a las condiciones contextuales.

2.5.4.3 Modelamiento Contextual

El último paradigma se basa en un enfoque de modelado contextual. Se utiliza la información contextual directamente en la función de recomendación (2.2), como un predictor explícito de la calificación de un usuario para un elemento. Esto da lugar a funciones de recomendación verdaderamente multidimensionales, que en esencia representan modelos de predicción (p. ej., sobre la base de los árboles de decisión, regresiones o modelos probabilísticos) o cálculos heurísticos que incorporan contexto, además de los datos del usuario y de elementos, es decir Puntuación => R (Usuario; Artículo; Contexto).

2.5.4.4 Combinación de múltiples enfoques

Por último, mediante la combinación de varios métodos se puede tener un mejor rendimiento y a su vez contar con una recomendación más precisa, debido a que el resultado de las recomendaciones pasa por varios filtros contextuales. [26]

2.6 Pro-actividad

Hasta ahora, hemos visto que los sistemas de recomendación tradicionales suelen seguir un patrón de respuesta según la petición del usuario, es decir, estos sistemas sólo devuelven sugerencias de elementos cuando un usuario hace una petición explícita. En SRSC, la experiencia del usuario, posiblemente, podría mejorarse mediante la incorporación de la pro-actividad: El sistema empuja las recomendaciones al usuario cuando la situación actual parece apropiado, sin necesidad de una petición explícita.

Por ejemplo, una guía de restaurantes móviles que se ejecuta en un Smartphone sugiere un restaurante al usuario cuando este se encuentra caminando muy cerca de los restaurantes que son afines a sus gustos.

2.6.1 Reducción al mínimo de los costos de la Pro-actividad

Sistemas de recomendación pro-activos tienen por objeto proporcionar automáticamente la recomendación en el momento adecuado (es decir, un proceso no iniciado por el usuario). Pero esto da lugar a un mayor desafío inherente a la pro-actividad, que es identificar el momento oportuno para notificar al usuario acerca de la recomendación.

Una situación en particular puede inducir una buena recomendación, como así mismo la recomendación puede llegar a ser mala. Por ejemplo, a la hora del almuerzo el sistema podría querer recomendar un restaurante de alto rango cercano. Sin embargo, en esos momentos el usuario puede estar al teléfono o en medio de una conferencia y esta notificación podría presentarse preocupante. Por lo que se debe considerar el estado actual del usuario antes de enviar las notificaciones y tener un momento oportuno para así lograr minimizar los problemas de interrupciones. A continuación se enuncian los efectos negativos provocados por las interrupciones:

- Recuperación lenta.
- Errores
- Eficiencia reducida.
- Estrés.

Todos estos efectos generales también son válidos en los sistemas de recomendación proactivos móviles. En consecuencia, lo que hay que minimizar son las interrupciones, (es decir, los costos de recomendaciones pro-activas), debe ser el objetivo a largo plazo.

2.6.2 Procesamiento de las Interrupciones

No todas las recomendaciones terminan siendo una interrupción, a continuación se presentan las distintas formas de tratar una notificación:

- *Detección y Remoción de los Intrusos.* El anuncio de la interrupción no se detecta porque el estímulo no es lo suficientemente saliente y la interrupción no se realiza.
- *Interpretación y despido involuntario*. Se detecta el estímulo de anunciar, pero la importancia del anuncio no se interpreta y la interrupción no se procesa.
- *Integración y Retiro intencional*. La importancia del anuncio se interpreta, pero el usuario decide continuar realizando la tarea en curso sin tomar asunto a dicha interrupción.
- *Integración Preferente.* La tarea de interrupción se inicia inmediatamente sin tener en cuenta las implicaciones de realizarla en ese punto, y la lleva a cabo hasta el final de la interrupción antes de reanudar la tarea que estaba en curso.
- *Integración intencional.* El usuario puede considerar parar la tarea en curso para tratar la nueva notificación.

2.6.3 Decidir sobre las notificaciones

Dada una nueva tarea o evento al que se necesite la atención del usuario (recomendación), la decisión de hacer una notificación debe depender del contexto. Según Kern y Schiele [27] cinco factores constituyen este contexto:

- *Importancia de una nueva tarea o evento.* Un usuario será más receptivo a una notificación cuando es importante para él en su situación actual (por ejemplo, recibir una recomendación de restaurante cuando el usuario está de vacaciones por lo general será más importante que recibir una cuando el usuario está en casa).

- *Actividad del usuario.* La actividad a la que se dedica el usuario durante la interrupción (por ejemplo, si el usuario está conduciendo, una notificación será menos oportuna a revisar que cuando está caminando en la calle).

- *La interacción social.* Si el usuario está interactuando con los demás, y si es así, de qué manera.
- *La situación social.* Por ejemplo, si el usuario está en un tren o en un restaurante.
- *Posición.* Latitud y Longitud.

El usuario, sin embargo, no es la única persona que puede ser interrumpida, sino también una notificación puede perturbar el medio ambiente. Por ejemplo, un usuario podría no importarle ser notificado en una biblioteca pública, en cambio, otros son propensos a sentirse perturbados.

2.6.4 Timing de una Notificación

A menudo, el tiempo de una nueva recomendación con una alta puntuación emergente no coincide con un buen momento para una notificación, debido a la baja de interrupción personal o social. Por tanto, es conveniente esperar a realizar la notificación en lugar de cancelarla.

Sin embargo, las recomendaciones sensibles al contexto son volátiles en la naturaleza, es decir, si el contexto cambia en cierta medida, un elemento recomendado podría no coincidir con el contexto. Por lo tanto, las notificaciones no pueden ser retenidas indefinidamente. El reto consiste en retrasar una notificación el momento justo o iniciar el proceso de recomendación sólo cuando la situación "tiempo" sea la adecuada para recomendar. [11]

2.6.5 Pro-actividad en Sistemas de Recomendación

Ricci [28] comenta que la pro-actividad es uno de los temas de investigación más interesantes en el campo de la recomendación. Ricci llega a la conclusión de que "ninguno de los sistemas revisados existentes es capaz de interrumpir de forma proactiva la actividad de los usuarios con las recomendaciones no solicitadas pero relevantes" y " las recomendaciones pro-activas pueden revolucionar el papel de los sistemas de recomendación en el tema orientado a la búsqueda de información y de herramientas de toma de decisiones para el descubrimiento de información".

Una reciente arquitectura de sistemas de recomendación pro-activos y sensibles al contexto fue propuesta por Al Tair [29]. Este sistema utiliza la información de contexto como la hora y el lugar para recomendar hoteles y restaurantes para el viaje de un usuario. El enfoque de este trabajo está en la inferencia de elementos adecuados. Después de que el usuario ha seleccionado los detalles del viaje, el sistema puede enviar notificaciones para recordarle al usuario acerca de los próximos eventos. Por lo

tanto, este sistema no investiga, de hecho lo que hace es generar una recomendación pro-activa.

En el dominio del sistema ubicuo, Aritoni y Negru [30] describieron un ambiente de sistema de recomendación inteligente que utiliza sensibilidad al contexto para lograr la conservación de energía en los hogares. El objetivo de este sistema es cambiar el comportamiento de los usuarios y no recomendar como prioridad elementos desconocidos.

En cuanto a las interrupciones, sólo se ha cubierto en escenarios muy específicos para recomendaciones pro-activas. Por ejemplo, la Puerta-Melguizo, [31] propuso un sistema de recomendación pro-activo para la escritura, que tiene como objetivo la reducción de las interrupciones en el proceso de composición de un artículo científico. En dos experimentos, se examinó la eficacia de las recomendaciones durante la planificación y revisión.

Capítulo 3

3 Desarrollo e Implementación

3.1 Análisis de Requerimientos

Para llevar a cabo el funcionamiento completo del Recomendador se deben cumplir con ciertas especificaciones establecidas previamente, las cuales nos servirán de guía a lo largo de todo el proceso de diseño e implementación final. Estas especificaciones se detallan a continuación:

- Lenguaje de programación "Python"
- La información de los elementos educativos es obtenida a través de las APIs:
  - Youtube
  - Flickr
  - SoundCloud
- Implementar un algoritmo de recomendación en base a los "Items" observados por el usuario
- Búsqueda de elementos usando "Sensibilidad al Contexto"
- Poblar la base de datos "Neo4j" con las búsquedas de los usuarios
- Interfaz para visualizar los datos obtenidos (Aplicación Web)

El desarrollo se lo implementa en Python debido a la gran flexibilidad que posee este lenguaje para integrarse con las API anteriormente mencionada, y a su vez ofrece un gran número de librerías que agilitan la implementación.

La información es obtenida a través de Youtube, Flickr y SoundCloud, esto es para tener en la base "Videos, Fotos y Músicas" respectivamente. Cada elemento educativo se almacena de manera distinta logrando de esta forma agilitar las búsquedas por parte del usuario. Por último, para realizar la recomendación se utiliza la técnica de filtrado colaborativo, implementando un algoritmo que recomienda "elementos" al usuario que se encuentra en sesión en base a sus antiguas búsquedas. La lógica para la implementación fue ("observado-a-observar") de elementos vistos por el usuario y de los que tendría intención de ver.

3.2 Interfaces de Programación de Aplicaciones (API)

Para poder obtener los elementos "Videos, Fotos y Músicas" de las distintas fuentes la realizamos mediante las interfaces de programación que ofrecen las aplicaciones, debido a la versatilidad de Python, cada una de las apps presenta su versión en el lenguaje para realizar las distintas gestiones a sus recursos. A continuación se muestran las implementaciones de las APIs:

3.2.1 Youtube

El API de datos de YouTube permite a las aplicaciones cliente poder recuperar y actualizar el contenido de YouTube en forma de "Feeds" de datos de Google. La aplicación cliente puede utilizar el API de datos de YouTube para buscar y actualizar vídeos, comentarios, respuestas, listas de reproducción, perfiles de usuario y los contactos de los usuarios, así como la consulta de los vídeos que coincidan con determinados criterios. [32]

- *Dependencias*
  - *httplib*
    httplib ha sido parte del núcleo de la biblioteca de Python desde la versión 2.0, por lo que ya debería estar presente en su sistema.

  - *urllib*
    urllib es también una biblioteca de Python, por lo que no debería tener que instalarlo.

La Instalación de la librería es muy fácil, solo se deben descargar los archivo desde la página de google code [33] y seguir los pasos de instalación que se encuentran en el archivo "INSTALL.txt".

3.2.1.1 Implementación

Para empezar a utilizar la librería "gdata" para obtener los videos de youtube debemos seguir los siguientes pasos:

- *Importa Librerías*
  - gdata.youtube
  - gdata.youtube.service
- *Instanciar un Cliente para la conexión con Youtube*
  - Cliente = gdata.youtube.service.YouTubeService()
  - Query = gdata.youtube.service.YouTubeVideoQuery()
- *Configurar filtros de Búsqueda*

- *Presentar la Información obtenida*

El código final que se utilizó para la implementación en Python para realizar las búsquedas de videos se muestra a continuación en la **(Figura 17)**:

```
import gdata.youtube
import gdata.youtube.service

#Instanciamos la API de Youtube
client = gdata.youtube.service.YouTubeService()
query = gdata.youtube.service.YouTubeVideoQuery()

# Obtenemos la palabra de busqueda
query.vq = form.getvalue("search")
query.max_results = 5
query.start_index = 1
query.racy = "exclude"
query.format = '5'
query.orderby = "relevance"

feed = client.YouTubeQuery(query)

#Obtenemos los elementos
for entry in feed.entry:
    url = entry.GetSwfUrl()
    op = con.searchNodeDuplicated(url, "Video")
    if (op == 0) :
        print "<li><p>Titulo: %s" % entry.media.title.text
        print "<br />Publicado en: %s" % entry.published.text
        #print "<br />description: %s" % entry.media.description.text
        print "<br />URL: <a href=%s target='_blank'>%s</a>" % (url,url)
        print "</p></li>"
        con.createNodeVideoYoutube(entry.media.title.text,entry.published.text, url)
```

Figura 17. Código Python para buscar Videos en Youtube

3.2.2 Flickr

Flickr ofrece un paquete de librerías en Python para manejar su aplicación [34], este api devuelve objetos que son fáciles de manejar, pero para poder hacer uso de la misma se deben tener claves de acceso, la cual las obtenemos en la siguiente dirección: http://www.flickr.com/services/api/ una vez que tengamos la clave de acceso procedemos con la implementación.

3.2.2.1 Implementación

Cuando tengamos las claves de acceso (API_KEY, API_SECRET) deben ser configurados en el archivo "*flickr.py*" para poder empezar a usar la librería. Estos archivos deben ir en el directorio raíz de la aplicación para importarlos a nuestras distintas clases y así mismo constar con los permisos necesarios para utilizarlos. A continuación se presentan los pasos para obtener imágenes de flickr:

1. *Importar Librería*
    a. import flickr
2. *Enviar a Buscar las fotos dependiendo de nuestro contexto. La función "photo_search" recibe varios parámetros, entre los más importante: per_page, tag_mode, y tags.*
    a. flickr.photos_search(tags=txtsch, tag_mode='all', per_page=5)

A continuación se muestra en la **(Figura 18)** el código completo en Python para realizar las búsquedas en flickr:

```
#!/usr/bin/env python
# -*- conding:utf-8 -*--
print "Content-type: text/html\n\n"

import flickr

#Busqueda en Flickr
print "-"
photos = flickr.photos_search(tags=txtsch, tag_mode='all', per_page=5)
for photo in photos:
    op = con.searchNodeDuplicated(photo.url, "Image")
    if (op == 0) :
        try:
            print "<li><p>Title: "+photo.title+ "<br />"
            print "Publicado en: "+photo.datetaken + "<br />"
            print "Tags: "
            tagsc = ""
            for tag in photo.tags:
                    tagsc = tagsc + tag.text +" "
                    print tag.text.encode('utf-8')
            print "<br />URL: <a href=%s target='_blank'>%s</a>" %  (photo.url,photo.url)
            print "</li>"
            con.createNodeImageFlickr(photo.id, photo.title, photo.url, photo.datetaken, tagsc.encode('utf-8'))
        except:
            continue
```

Figura 18. Código en Python para realizar búsquedas en Flickr

3.2.3 SoundCloud

Usando la API de SoundCloud, puede crear aplicaciones que tengan sonido en la web. En la guía que proporciona SoundCloud se explican muchos ejemplos de código para muchos casos de uso común: como jugar y subir sonidos o cómo tomar ventaja de las muchas características sociales de SoundCloud. [35]

SoundCloud SDK está disponible para Python, Ruby, PHP, Java, iOS y JavaScript, pero para hacer uso de la Api se deben tener claves de acceso, las podemos obtener registrándonos en SoundCloud como desarrollador en la siguiente página: https://soundcloud.com/

La instalación de esta librería la realizamos utilizando el siguiente comando, ejecutándolo en una terminal de linux:

pip install soundcloud

3.2.3.1 Implementación

Una vez que nos hayamos registrado en la página de SoundCloud obtendremos una clave de acceso que debe ser configurada para iniciar la búsqueda de música, la implementación en Python se muestra a continuación:

1. *Importar Librería*
    a. Import soundcloud
2. *Instanciar Cliente*
    a. soundcloud.Client(client_id='')
3. *Realizar las Búsquedas con nuestro contexto*

A continuación en la **(Figura 19)** se muestra el código completo en Python:

```
#!/usr/bin/env python
# -*- conding:utf-8 -*--
print "Content-type: text/html\n\n"

import soundcloud

# create a client object with your app credentials
client = soundcloud.Client(client_id='529295acc334c5fea23d339ac03300bc')
# find all sounds of buskers licensed under 'creative commons share alike'
tracks = client.get('/tracks', q=palsch, license='cc-by-sa')

print "-"
for track in tracks:
    var = "/tracks/"+ str(track.id)
    trackdef = client.get(var)
    op = con.searchNodeDuplicated(trackdef.permalink_url, "Music")
    if (op == 0) :
        try:
            print "<li><p>Title:  %s <br />" % (track.title)
            # get the tracks streaming URL
            print "URL: <a href=%s target='_blank'>%s</a> <br />" % (trackdef.permalink_url,trackdef.permalink_url)
            #print "Descripcion: %s </p>" % track.description
            print "Tags: %s" % track.tag_list
            print "</li>"
            con.createNodeMusicSoundCloud(track.id, track.title, trackdef.permalink_url,track.tag_list.encode('utf-8'))
        except:
            continue
```

Figura 19. Código en Python para realizar búsquedas en SoundCloud

3.3 Instalación de Neo4j

Neo4j ofrece desde su página oficial la descarga de la base [36], desde una versión libre a otra comercial, esto va a depender del uso que se le dé a la base al momento de hacer las aplicaciones.

Neo4j se ejecuta como un servidor de aplicaciones, basado en una interfaz web RESTful para realizar las gestiones de la base. Cuando haya descargado el paquete de la página y descomprimido en un directorio, se podrán observar los siguientes archivos:

- Bin Script y otros ejecutables
- Conf Configuración del Servidor

- Data Base de Datos, logs y otros archivos
- Doc lecturas de información de la base
- Lib Librerías
- Plugins Extensiones
- System

Para realizar la instalación de la base de datos nos guiamos por el archivo de configuración que trae el instalador, los pasos para la instalación se muestran a continuación:

1. Abrir una "Terminal" y dirigirse al directorio de instalación
2. Empezar el servidor:
    a. Windows bin\Neo4j,bat
    b. Linux ./bin/neo4j
3. En un cliente web ingresar la dirección:
    a. http://localhost:7474/db/data

A continuación en la **(Figura 20)** se muestra la pantalla de bienvenida del Servidor Neo4j:

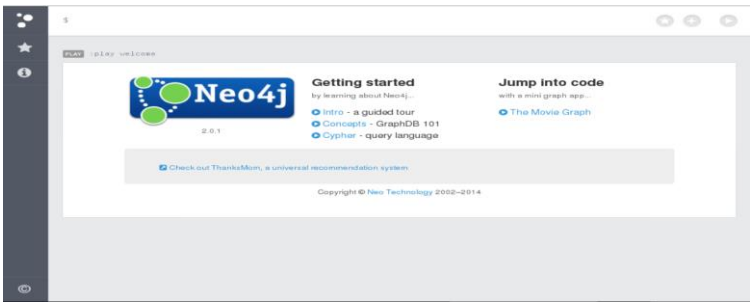

Figura 20. Pantalla de Bienvenida de Neo4j

3.4 Modelamiento de Objetos

Teniendo en cuenta que la base de datos es de forma gráfica, un mundo totalmente diferente al de las bases tradicionales, se presenta a continuación la forma en que se almacenan los videos, fotos y músicas obtenidos de las distintas fuentes y a sí mismo la manera en que se gestiona al usuario. Por tal motivo, se deben tener claro las propiedades o atributos que va a tener cada objeto en el recomendador.

3.4.1 Videos, Fotos y Músicas

EL almacenamiento de los videos, fotos y músicas obtenidos a través de las búsquedas de los usuarios la realizamos creando nodos a los distintos elementos, a cada nodo se le provee con una etiqueta que permite luego al momento de ejecutar

querys especificar los nodos a incluir, en la **(Figura 21)** se muestran las propiedades de cada elemento y el query en Cypher para crear el nodo respectivo.

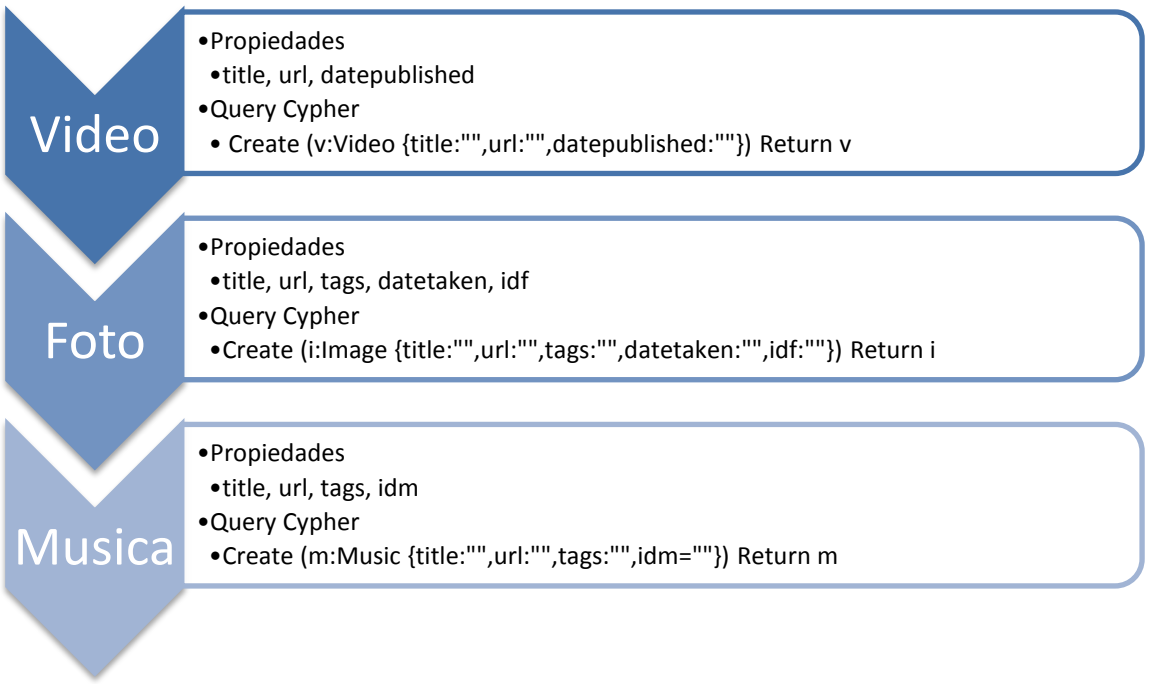

Figura 21. Modelamiento de Elementos

3.4.2 Usuarios

Para poder realizar las recomendaciones, necesitamos que un usuario ingrese al sistema, por tal motivo, debemos tener creados los usuarios en la base. De igual manera que almacenamos los elementos (Videos, Fotos, Músicas), se procederá a crear los nodos "Usuarios", tal como se muestra en la **(Figura 22)**:

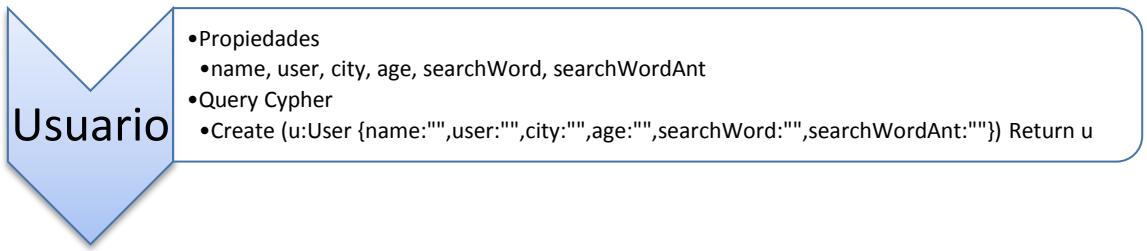

Figura 22. Modelamiento del Objeto Usuario

3.5 Descripción de la Implementación

En esta sección, se procederá a explicar cada una de las partes esenciales para el correcto funcionamiento del recomendador, empezando por una breve descripción de las librerías adicionales utilizadas, como por ejemplo: "numpy, py2neo, etc.". Luego, se

explicara el manejo de la sesión del usuario, manejo de las búsquedas y finalmente se detallará el algoritmo de recomendación.

3.5.1 Dependencias Adicionales

Para realizar cada una de las funciones del recomendador se han utilizados otras librerías que han facilitado el desarrollo, a continuación se listan algunas importantes [37]:

- *SHA.* Este módulo implementa la interfaz con algoritmo hash seguro del NIST, conocido como SHA-1. SHA-1 es una versión mejorada del algoritmo original hash SHA.
- *SHELVE.* Utilizado para objetos de persistencia en Python.
- *COOKIE.* Este módulo define clases para el manejo automático de cookies HTTP. Es útil para el acceso a sitios web que requieren pequeñas cantidades de datos.
- *TIME.* Aunque este módulo está siempre disponible, no todas las funciones están disponibles en todas las plataformas. La mayor parte de las funciones definidas en la presente convocatoria módulo de funciones de biblioteca plataforma C con el mismo nombre. A veces puede ser útil consultar la documentación de la plataforma, ya que la semántica de estas funciones varía entre plataformas.
- *OS.* Este módulo proporciona una manera portátil de utilizar sistema operativo.
- *CGI.* Este módulo define una serie de utilidades para el uso de scripts CGI.
- *CGITB.* El módulo cgitb proporciona un controlador excepción especial para los scripts de Python. (Su nombre es un poco engañoso. Originalmente fue diseñado para mostrar una amplia información de rastreo en el código HTML para scripts CGI.
- *RANDOM.* Este módulo implementa generadores de números pseudo-aleatorios.
- *NUMPY.* NumPy es el paquete fundamental para la computación científica con Python. Contiene entre otras cosas [38]:
    - un poderoso objeto de matriz N-dimensional
    - sofisticadas funciones (radiodifusión)
    - herramientas para la integración de C / C + + y Fortran
    - álgebra lineal útil, transformada de Fourier, y las capacidades de números aleatorios

- *PY2NEO.* Es una biblioteca de Python simple y pragmática que proporciona acceso a la popular base de datos Neo4j a través de su interfaz de servicios web RESTful. Sin dependencias externas, la instalación es sencilla y cómo empezar con la codificación es fácil. La biblioteca se mantiene activa en GitHub, actualizada regularmente en el índice de paquetes de Python y está construida exclusivamente para Neo4j. [39]

3.5.2 Sesión de Usuario

Debido a que para generar una recomendación se necesita de un usuario, se implementó la clase sesión que almacena una cookie en el cliente con la información del usuario. Para ver en mejor detalle la clase sesión. A continuación en la **(Figura 23)** se muestra la pantalla inicial de la aplicación, en donde el usuario debe ingresar su "user".

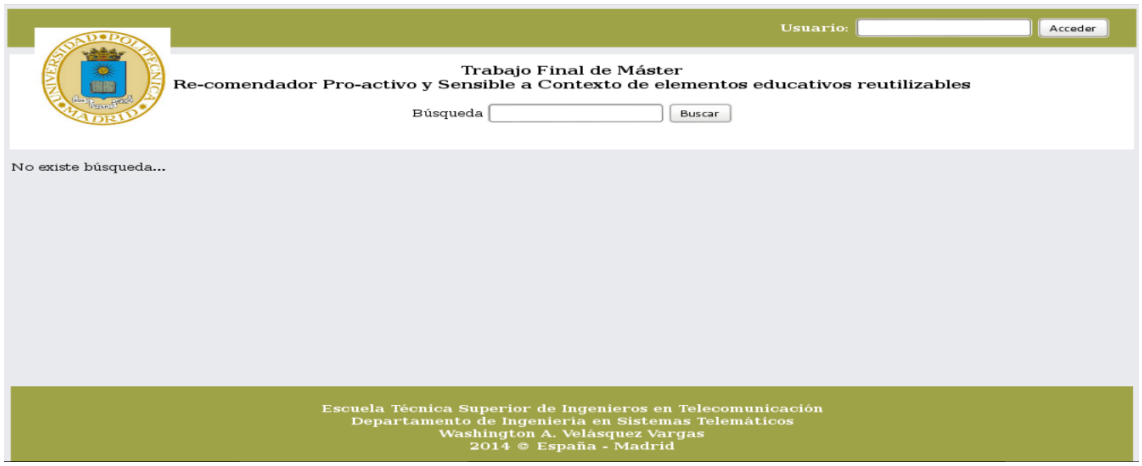

Figura 23. Pantalla Inicial App

Una vez que el usuario ha proporcionado su identificador, en el servidor, se proceden a realizar las acciones, tal como se muestra en el diagrama de interacción de objetos en la **(Figura 24)**:

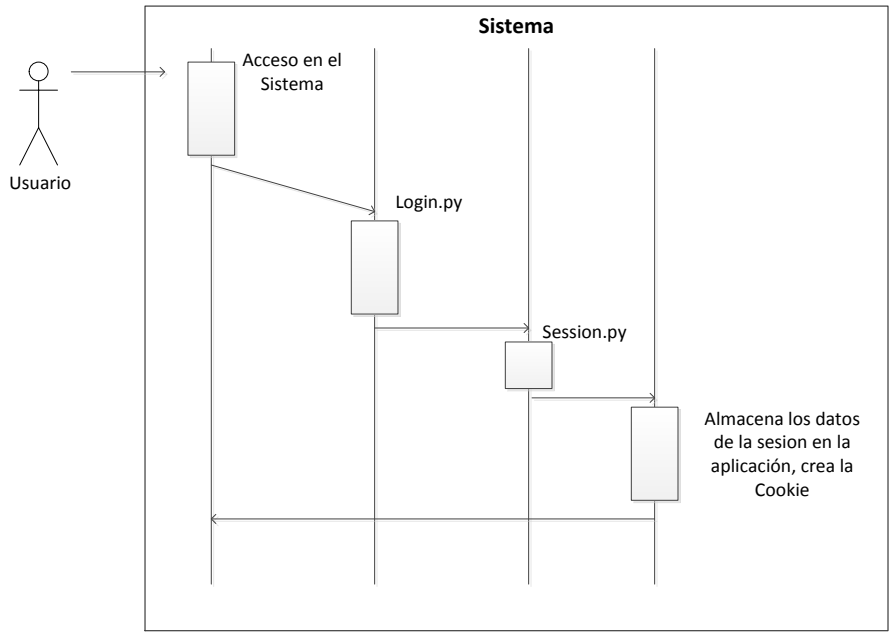

Figura 24. Diagrama de Interacción de Objetos: Acceso a la App

3.5.3 Conexión con la Base

Para realizar la conexión con la base de datos utilizamos la librería "py2neo", esta nos ayuda con una interfaz "Restful" para conectarnos con la base "Neo4j". Esta librería nos permite crear, modificar y eliminar nodos, relaciones y propiedades de un grafo. Para empezar con el manejo lo primero que debemos hacer es tener una instancia a la base de la siguiente manera:

```
graph_db = neo4j.GraphDatabaseService("http://localhost:7474/db/data/")
```

Luego de esto, ya podemos interactuar con la base de datos mediante la variable graph_db, no olvidar que debemos cargar la librería mediante las siguientes líneas de código [39]:

```
from py2neo import neo4j
from py2neo import node, rel
```

3.5.4 Manejo de Búsquedas

La búsqueda de elementos educativos la realizamos a través de un "form" que recibe los datos y los envía al server, estas búsquedas la hacemos de diferentes maneras:

- Búsqueda Sensible al Contexto
- Búsqueda en las APIs

3.5.4.1 Búsqueda Sensible al Contexto

Esta búsqueda la realizamos cuando el usuario va ingresando el contexto, es decir; a medida que el usuario va tecleando cada letra se va haciendo una petición al server para ir buscando en la base "Neo4j". Estas peticiones las hacemos utilizando Ajax para hacer los requerimientos, lo que nos ayuda a no hacer un rende rizado de toda la página. A continuación en la **(Figura 25)** se muestra el código en Ajax utilizado para el requerimiento al servidor:

```
function xmlhttpPostSensibility(strURL) {
    var form     = document.forms['fInV'];
    var word = form.search.value;
    if (word.length >=3){
        limpiar()
        var xmlHttpReq = false;
        var self = this;
        // Mozilla/Safari
        if (window.XMLHttpRequest) {
            self.xmlHttpReq = new XMLHttpRequest();
        }
        // IE
        else if (window.ActiveXObject) {
            self.xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
        }
        self.xmlHttpReq.open('POST', strURL, true);
        self.xmlHttpReq.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        self.xmlHttpReq.onreadystatechange = function() {
            if (self.xmlHttpReq.readyState == 4) {
                updatepage(self.xmlHttpReq.responseText);
                var img = document.getElementById('imgc');
                img.style.visibility = "hidden";
            }
        }
        self.xmlHttpReq.send(getquerystring());
      }
}
```

Figura 25. Enviar requerimientos Ajax al server

3.5.4.2 Búsqueda en las distintas API

La búsqueda en las Apis (Youtube, Flickr, SoundCloud) la hacemos con el archivo "searchApis.py", en este archivo se encuentran las implementación requeridas de cada api para realizar búsquedas en sus bases. El código completo de este archivo lo puede observar en el Anexo A. A continuación en la **(Figura 26)** se muestran los datos de una de las búsquedas por parte del usuario.

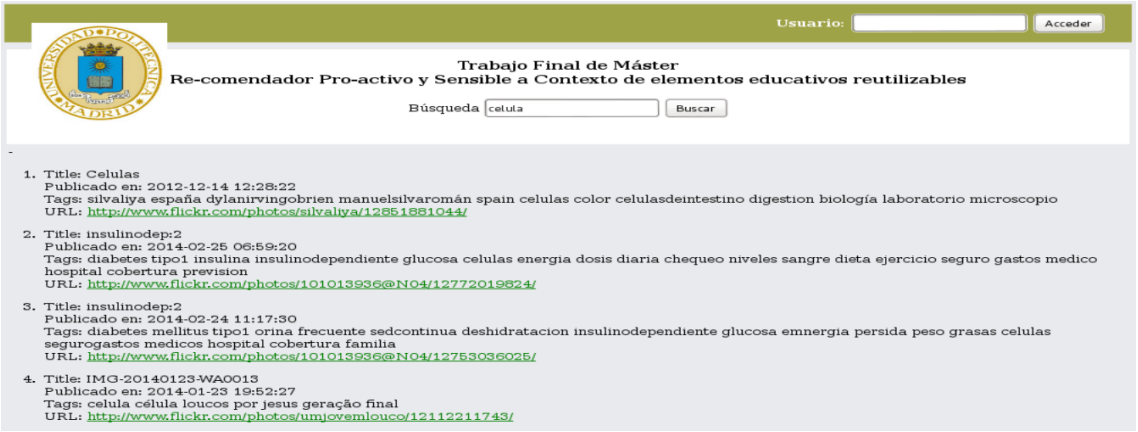

Figura 26. Búsqueda de Elementos

3.5.5 Algoritmo de Recomendación

El algoritmo de recomendación fue implementado para realizar sugerencias de elementos en base a lo que otros usuarios hayan visto y se acople al contexto del usuario en sesión. Para esto nos regimos según el tipo de recomendación de "Filtrado colaborativo" en el uso de la lógica "observado-a-observar". La implementación del código está divida en estas cinco secciones:

1. Obtener los elementos que el usuario en sesión haya visto en base al contexto de sus dos últimas búsquedas.
2. Localizar cada uno de los usuarios que hayan revisado los elementos obtenidos en el ítem 1.
3. Luego de tener los usuarios del ítem 2, procedemos a obtener los elementos adicionales que hayan revisado en base al contexto del usuario en sesión.
4. Procedemos a realizar el cálculo de inferencia entre cada elemento, para esto usamos el coseno entre cada par de elementos.

Como por ejemplo;

Tabla 1. Ejemplo de Cálculo de Recomendaciones

	Elemento 1	Elemento 2	Elemento 3
Usuario 1	Lo ha visto	No lo ha visto	Lo ha visto
Usuario 2	No lo ha visto	Lo ha visto	Lo ha visto

Cálculos de Inferencia mediante la ley del Coseno:

Inferencia entre el elemento 1 y 2: $(1,0).(0,1) \Big/ ||1,0||||0,1|| = 0$

Inferencia entre el elemento 1 y 3: $(1,0).(1,1) / ||1,0||||1,1|| > 0$

Inferencia entre el elemento 2 y 3: $(0,1).(1,1) / ||0,1||||1,1|| > 0$

Una vez que tenemos los valores de cada inferencia procedemos a elegir los valores mayores a cero. En Resumen, el ejemplo quedaría: Los usuarios que hayan revisado el elemento 1 podrían estar interesado en el elemento 3, los usuarios que hayan revisado el elemento 2 podrían interesarles el elemento 3 y los que hayan revisado el elemento 3 les podría interesar el 1 y el 2. Con esto tendremos una matriz de inferencias, todo va a depender del número de elementos que se tengan que procesar.

5. Escogemos una fila al azar de la matriz y procedemos a obtener los elementos que tengan valor mayor a cero.
6. Los elementos del ítem 5, procedemos a mostrárselos al usuario.

El algoritmo completo lo puede observar en el Anexo B.

A continuación en la **(Figura 27)** se muestra el panel de las recomendaciones en la app, con recomendaciones en base al contexto del usuario "silkend".

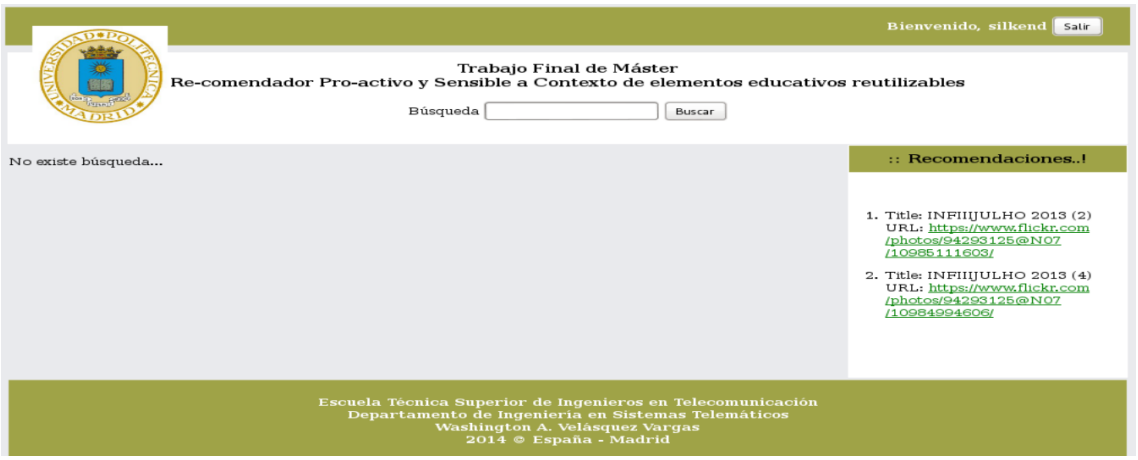

Figura 27. Panel de las Recomendaciones

Capítulo 4

4 Resultados Obtenidos

En el presente capítulo se muestran las pruebas realizadas que permiten mostrar el funcionamiento del algoritmo de recomendación y así mismo la obtención de datos a través de las diferentes APIs, se presentan a continuación los escenarios elegidos:

- Búsqueda de Elementos Educativos
- Usuario en sesión realiza búsqueda de elementos

Adicional si desea obtener el código completo del proyecto lo puede descargar desde la siguiente dirección: https://github.com/silkend/tfmRecomSystem

4.1 Búsqueda de Elementos Educativos

En este escenario se presenta la búsqueda de elementos educativos por parte de usuario que no poseen un identificador para ingresar a la app. Todas las búsquedas que se hagan se van a ir almacenando en la base Neo4j, esto ayuda a que las posteriores búsquedas no tengan que ir directamente a buscar a las fuentes externas, sino que ya se posee datos en la base que pueden ser de interés a otros usuarios. A continuación se presenta en la **(Figura 28)** una búsqueda en la app.

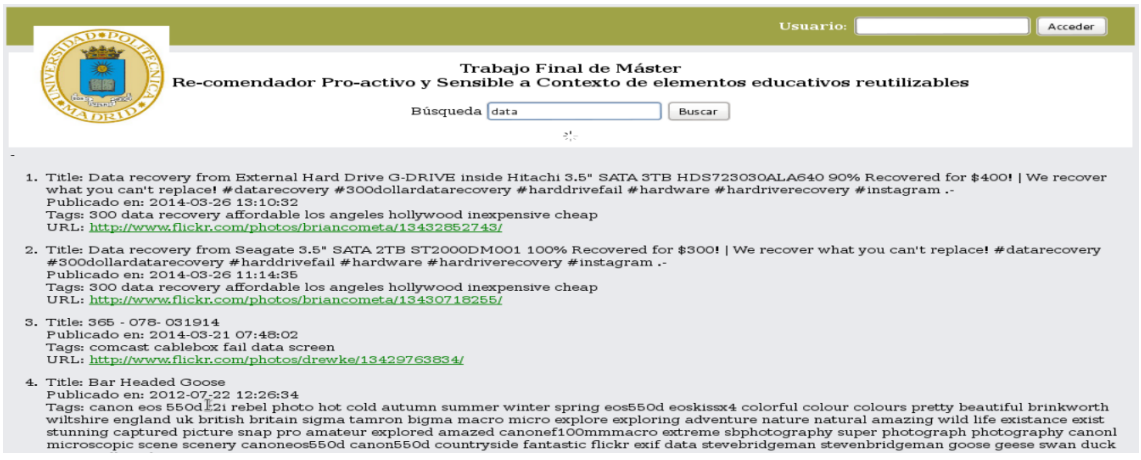

Figura 28. Búsqueda de Elementos Educativos

4.2 Usuario en sesión realiza búsqueda de elementos

Este escenario es parecido al "escenario 4.1" pero con la diferencia de que el usuario ingresa su identificador en la app, lo que ayuda a mantener un perfil de búsquedas y lograr hacer una recomendación en base a ellas.

En la **(Figura 29)** se muestra una búsqueda realizada por el usuario "silkend" y al mismo tiempo en el panel derecho podemos ver claramente las recomendaciones que el sistema ofrece en base a sus últimas búsquedas.

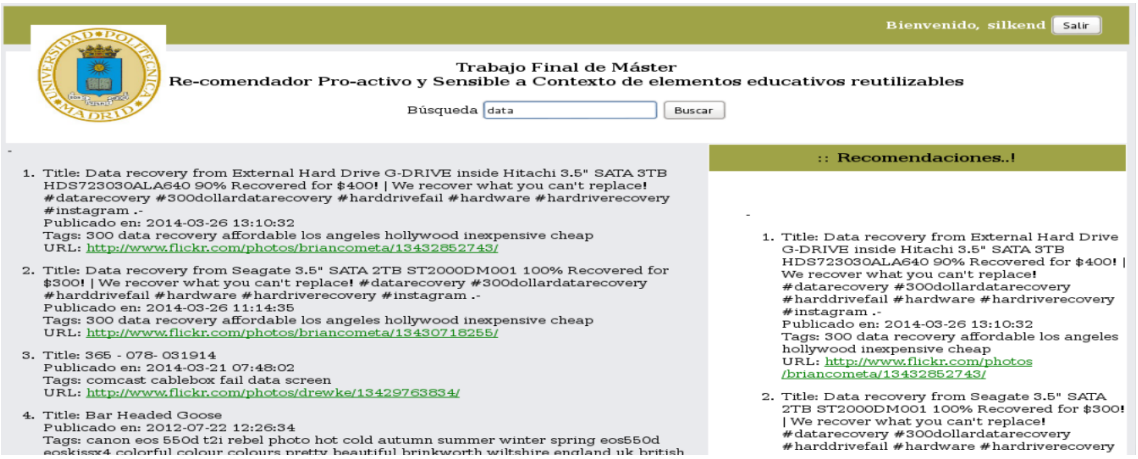

Figura 29. Búsquedas de un Usuario registrado en la app

A continuación en la **(Figura 30)** se muestran los nodos y relaciones de la base de datos "Neo4j" cuando un usuario (nodo azul) ha visto algún elemento, con esto se produce automáticamente una relación entre ese usuario y el ítem. Los ítems de color amarillo corresponden a las imágenes y los de color naranja a los videos. Además se aprecia el enlace que se forma con el nombre de "LOOKED".

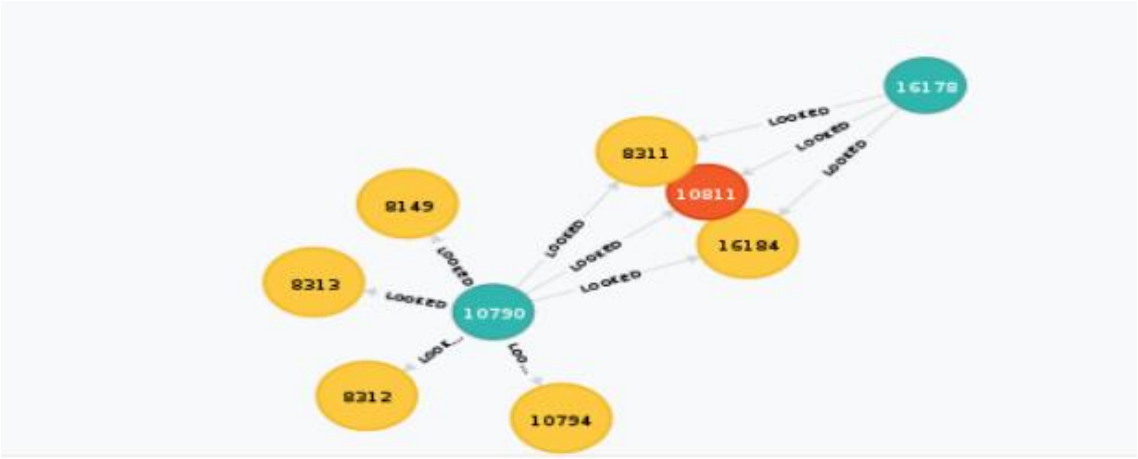

Figura 30. Nodos y Relaciones en Neo4j

5 Conclusiones y Comentarios

Los sistemas de recomendación son ampliamente utilizados por varios sitios en la red, esto se debe a la gran cantidad de información que circula a través de ella, por lo que la presencia de estos sistemas se ha vuelto indispensable para evitar que el usuario este perdido ante tanta información, sino más bien, lograr ofrecer recursos apropiados que se adapten a las necesidades de los consumidores.

El sistema de recomendación implementado es una versión prototipo y se basa en una arquitectura cliente/servidor con una interfaz web que permite a los clientes realizar sus búsquedas, esta versión toma en consideración las búsquedas anteriores. En función de esto el sistema compara con otros clientes, luego recomienda al usuario determinados ítems que puedan ser más atractivos para él.

Las bases de datos orientadas a grafos como se puede ver en el desarrollo de este proyecto son una clara alternativa a las bases de datos tradicionales, sobre todo para algunas aplicaciones sociales y web que requieren elevada escalabilidad. Estas bases de datos no son idóneas para todo, de hecho en la mayoría de los casos las bases de datos relacionales deberían seguir siendo la primera opción, debido a la capacidad de hacer JOIN y ciertas garantías que son muy importantes en algunas aplicaciones. Algo importante a decir es que debido al uso de las bases de datos orientadas a grafos es muy posible que las bases de datos tradicionales actuales evolucionen para incorporar capacidades NoSQL y así mejorar los motores de bases de datos obteniendo una persistencia transaccional políglota.

El trabajo con bases de datos de orientadas a grafos que en nuestro caso es “Neo4j”, requiere en la mayoría de los casos, conocer bien el negocio que se desea modelar para definir adecuadamente la estructura en la que se van a almacenar los datos. No obstante, este aspecto marca la diferencia en el rendimiento de las consultas, a favor de las NoSQL en comparación con las relacionales. Un esquema de datos bien ajustado a un negocio muy específico permite optimizar los resultados de las consultas desde la etapa de diseño.

Por otro lado, la construcción de un sistema de recomendación basado en contexto y que use filtrado colaborativo no es sencillo, debido a que se deben trabajar con datos estadísticos para tratar la manera de inferir lo que va a ser de agrado para el usuario. Para obtener una mejor calidad en las recomendaciones se trató con cálculos de correlación entre artículos que hayan sido vistos por otros usuarios que puedan tener las mismas aficiones que la persona que está realizando las búsquedas. Algo a destacar

es que gracias a la manera en que se almacenó los datos podemos seguir mejorando el algoritmo de recomendación, esto se debe a que cada artículo posee metadatos que pueden ser claramente filtrados o se podrían añadir fácilmente nuevos atributos que sirvan para realizar estas mejoras.

En cualquier caso aunque hay camino por andar, los sistemas de recomendación con filtrado colaborativo tienen actualmente una enorme funcionalidad lo que está provocando que más usuarios lo usen. Por último, para poder cumplimentar al sistema se le agrego parte de pro-actividad esto lo logramos cada vez que el usuario ingresa al sistema, se le ofrecen ciertos artículos utilizando las búsquedas anteriores como referencias y así mismo cada vez que se haga alguna actualización a la página se le ofrecerán nuevas recomendaciones, todo va a depender de las búsquedas del usuario. Debido a que no se pudo llevar el entorno a un modelo móvil para lograr obtener la localización como un buen factor y de esta manera ofrecer mejor este campo.

Finalmente, comentar que para el autor, el haber realizado este proyecto ha sido todo un desafío, porque en un principio la temática de sistemas de recomendación era totalmente desconocida. Pero obteniendo un aspecto positivo al haber podido aplicar muchas metodologías y habiendo adquirido nuevas habilidades durante el desarrollo del proyecto.

Anexos

Anexo A

Archivo para realizar las búsquedas en las fuentes externas.

```
#!/usr/bin/env python
# -*- conding:utf-8 -*--

print "Content-type: text/html\n\n"

import flickr
import soundcloud
import cgi, cgitb
import gdata.youtube
from graphbase.conexion import Conexion
import gdata.youtube.service

cgitb.enable()
# Creamos instancia de FieldStorage
form = cgi.FieldStorage()
txtsch = form.getvalue("search")
#Variable para interactuar con la bdog
con = Conexion()
print "<p><b>Nuevos Archivos Encontrados</b></p>"
print "<ol>"
#Busqueda en las APIS
#Busqueda en Flickr
photos = flickr.photos_search(tags=txtsch, tag_mode='all', per_page=5)
for photo in photos:
    op = con.searchNodeDuplicated(photo.url, "Image")
    if (op == 0) :
        try:
            tagsc = ""
            for tag in photo.tags:
                    tagsc = tagsc + tag.text +" "
            nodo = con.createNodeImageFlickr(photo.id, photo.title, photo.url,
photo.datetaken, tagsc.encode('utf-8'))
            print "<li><p>Title: "+photo.title+ "<br />"
            print "Publicado en: "+photo.datetaken + "<br />"
            print "Tags: "
            print tagsc.encode('utf-8')
            print "<br />URL: <a href=%s target='_blank' alt='%s' >%s</a>" %
(photo.url,nodo._id,photo.url)
            print "</li>"
        except:
            continue


#Instanciamos la API de Youtube
#Busqueda en Youtube
clienty = gdata.youtube.service.YouTubeService()
query = gdata.youtube.service.YouTubeVideoQuery()
query.vq = txtsch
query.max_results = 5
query.start_index = 1
query.racy = "exclude"
query.format = '5'
query.orderby = "relevance"
try:
    feed = clienty.YouTubeQuery(query)
```

```
    for entry in feed.entry:
        url = entry.GetSwfUrl()
        op = con.searchNodeDuplicated(url, "Video")
        if (op == 0) :
            nodo=
con.createNodeVideoYoutube(entry.media.title.text,entry.published.text, url)
            print "<li><p>Titulo: %s" % entry.media.title.text
            print "<br />Publicado en: %s" % entry.published.text
            print "<br />URL: <a href=%s target='_blank'alt='%s'>%s</a>" %
(url,nodo._id,url)
            print "</p></li>"

except:
    print ""

#Busqueda en SoundCLoud
# create a client object with your app credentials
client = soundcloud.Client(client_id='529295acc334c5fea23d339ac03300bc')
# find all sounds of buskers licensed under 'creative commons share alike'
try:
    tracks = client.get('/tracks', q=txtsch, license='cc-by-sa')
    for track in tracks:
        var = "/tracks/"+ str(track.id)
        trackdef = client.get(var)
        op = con.searchNodeDuplicated(trackdef.permalink_url, "Music")
        if (op == 0) :
            try:
                nodo = con.createNodeMusicSoundCloud(track.id, track.title,
trackdef.permalink_url,track.tag_list.encode('utf-8'))
                print "<li><p>Title:  %s <br />" % (track.title)
                # get the tracks streaming URL
                print "URL: <a href=%s target='_blank' alt='%s'>%s</a> <br />" %
(trackdef.permalink_url,nodo._id,trackdef.permalink_url)
                print "Tags: %s" % track.tag_list
                print "</li>"
            except:
                continue
except:
    print ""
print "</ol>"
```

Anexo B

Código del algoritmo de recomendación, archivo "recomendación.py".

```
#!/usr/bin/env python
# -*- conding:utf-8 -*--

print "Content-type: text/html\n\n"

from graphbase.conexion import Conexion
import flickr
import cgi, cgitb
import gdata.youtube
import numpy as np
import random
import gdata.youtube.service
cgitb.enable()
form = cgi.FieldStorage()
#Algoritmo basado en Filtrado Colaborativo
con = Conexion()
us = form.getvalue("user")

nodesItems = con.getRelationshipNodeUser(us)
if (nodesItems.__len__() > 0):
    Items = []
    ItemsUser = []
    for nodeI in nodesItems:
        ItemsUser.append(nodeI.values[0]._id)
        Items.append(nodeI.values[0]._id)


    Users = []
    for item in Items:
        nodesUsersofItem = con.getRelationshipNodeItem(item)
        for nodeU in nodesUsersofItem:
            Users.append(nodeU.values[0].get_properties()['user'])

    Users = list(set(Users))


    for user in Users:
        #print user
        nodesItems = con.getRelationshipNodeUser(user)
        for nodeI in nodesItems:
            Items.append(nodeI.values[0]._id)

    Items = list(set(Items))

    valores = np.zeros((Items.__len__(),Users.__len__()))

    i = 0
    for user in Users:
        nodesItems = con.getRelationshipNodeUser(user)
        for nodeI in nodesItems:
            if (Items.__contains__(nodeI.values[0]._id)):
                idx = Items.index(nodeI.values[0]._id, )
                valores[idx][i] = 1
        i= i+1

    cosenos = np.zeros((ItemsUser.__len__(),Items.__len__()))
    #print cosenos
    val = Items.__len__() - 1
    for i in range(ItemsUser.__len__()):
        for j in range(Items.__len__()):
            cosenos[i][j] = (valores[i].dot(valores[j]))/(np.linalg.norm(valores[i]) *
np.linalg.norm(valores[j]))

    fila = random.randrange(ItemsUser.__len__())
    ItemsRecom = []
    for i in range(cosenos[fila].__len__()):
        if (cosenos[fila][i] > 0):
            ItemsRecom.append(Items[i])
```

```
    #print ItemsRecom
    #print ItemsUser
    ItemsRecomFinales = set(ItemsRecom) - set(ItemsUser)

    #print ItemsRecomFinales
    for recom in ItemsRecomFinales:
        nodoI = con.getNodeItem(recom)
        dataN = nodoI.get_properties()
        print "<li><p>Title: "+dataN['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %  (dataN['url'],nodoI._id,dataN['url'])
        print "</li>"
    print ""
    if (len(ItemsRecomFinales) == 0):
        userB = con.getNodeUser(us)
        nodeF = con.searchNodesFlickr(userB.get_properties()['searchWord'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
        nodeF = con.searchNodesFlickr(userB.get_properties()['searchWordAnt'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
        nodeF = con.searchNodesYoutube(userB.get_properties()['searchWord'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
        nodeF = con.searchNodesYoutube(userB.get_properties()['searchWordAnt'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
        nodeF = con.searchNodesSoundCloud(userB.get_properties()['searchWord'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
        nodeF = con.searchNodesSoundCloud(userB.get_properties()['searchWordAnt'])
        for n in nodeF:
            if not (ItemsUser.__contains__(n.values[0]._id)):
                print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
                print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
                print "</li>"
else:
    userB = con.getNodeUser(us)
    nodeF = con.searchNodesFlickr(userB.get_properties()['searchWord'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
    nodeF = con.searchNodesFlickr(userB.get_properties()['searchWordAnt'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
```

```
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
    nodeF = con.searchNodesYoutube(userB.get_properties()['searchWord'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
    nodeF = con.searchNodesYoutube(userB.get_properties()['searchWordAnt'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
    nodeF = con.searchNodesSoundCloud(userB.get_properties()['searchWord'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
    nodeF = con.searchNodesSoundCloud(userB.get_properties()['searchWordAnt'])
    for n in nodeF:
        print "<li><p>Title: "+n.values[0]['title'].encode('utf-8')+ "<br />"
        print "URL: <a href=%s target='_blank' alt='%s'
onclick='javascript:doRelation(this)' >%s</a>" %
(n.values[0]['url'],n.values[0]._id,n.values[0]['url'])
        print "</li>"
```

Bibliografía

[1] P. S. Foundation, «python,» 2014. [En línea]. Available: http://www.python.org/. [Último acceso: 21 2 2014].

[2] M. Casado Martín y S. Guadalajara Pérez, «Departamento de Informática y Automática - Universidad de Salamanca.,» 2003. [En línea]. Available: http://zarza.fis.usal.es/~fgarcia/docencia/poo/02-03/trabajos/S1T8.pdf. [Último acceso: 21 2 2014].

[3] A. M. Ortiz, «rediris,» 2000. [En línea]. Available: http://elies.rediris.es/elies9/4-2-3.htm. [Último acceso: 22 02 2014].

[4] NoSQL, «NoSQL Archive,» 2009. [En línea]. Available: http://nosql-database.org/. [Último acceso: 23 02 2014].

[5] I. Robinson, J. Webber y E. Eifrem, Graph Database, First ed., published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2013.

[6] M. P. Palacio, J. A. Gonzales, R. Laurin y A. Tchouniki, «Minería de Datos Espaciales usando un Enfoque de Grafos,» Universidad de las Américas, Puebla, Instituto Nacional de Astrofísica Óptica y Electrónica, [En línea]. Available: http://www.academia.edu/449894/Mineria_De_Datos_Espaciales_Usando_Un_Enfoque_De_Grafos. [Último acceso: 12 2013].

[7] R. A. Roja, «Modelos de base de datos de Grafo y RDF,» Claudio Gutiérrez Gallardo, 8 2009. [En línea]. Available: http://www.tesis.uchile.cl/tesis/uchile/2009/angles_r/html/index-frames.html. [Último acceso: 10 12 2013].

[8] Neo4j, «Documentación Oficial de Neo4j,» 2013. [En línea]. Available: http://www.neo4j.org. [Último acceso: 12 2013].

[9] N. C. Fowler, «The Amazing Power of Social Networks and How They Shape Our Lives (HarperPress, 2011).,» 2011.

[ ] F. Ricci, «The context of a recommendation. In Proceedings of the Workshop on

10] Context-Aware Movie Recommendation,» CAMRa '10, pages 1–1, , New York, NY, USA, , 2010. ACM..

| D. G. Vico, «Contribution to proactivity in mobile context-aware recommender
11] systems,» Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid - Tesis doctoral, Madrid, 2013.

| D. Goldberg, D. Nichols, B. M. Oki y D. Terry, «Using collaborative filtering to
12] weave an information tapestry,» Vols. %1 de %2vol. 35, no. 12,, Commun. ACM, 1992, p. 61–70.

| P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom y J. Riedl, «In Proceedings of
13] the 1994 ACM conference on Computer supported cooperative work,» de *GroupLens: an open architecture for collaborative filtering of netnews.*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM., 1994.

| U. Shardanand y P. Maes, «Social information filtering: algorithms for
14] automating 'word of mouth'.,» de *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95*, New York, NY, USA,, 1995.

| D. H. Park, H. K. Kim, I. Y. Choi y J. K. Kim, «A literature review and
15] classification of recommender systems research,» de *Expert Systems with Applications*, vol. 39, 2012, p. 10059 – 10072.

| F. Ricci, L. Rokach y B. Shapira, «Introduction to Recommender Systems
16] Handbook.,» de *Recommender Systems Handbook*, L. R. B. S. a. P. B. K. In Francesco Ricci, Ed., Springer US, 2011, pp. 1 - 35.

| J. L. Herlocker, J. A. Konstan, L. G. Terveen y J. T. Riedl, «Evaluating
17] collaborative filtering recommender systems,» de *ACM Trans. Inf. Syst.*, January 2004, pp. 5 - 53.

| J. B. Schafer, D. Frankowski, J. Herlocker y S. Sen, «Collaborative filtering
18] recommender systems,» Springer-Verlag, Berlin, Heidelberg, In Peter Brusilovsky, Alfred Kobsa and Wolfgang Nejdl, 2007, p. 291–324.

| R. Burke, «Hybrid Web Recommender Systems,» de *The Adaptive Web*, vol. 4321,
19] A. K. a. W. N. In Peter Brusilovsky, Ed., Springer Berlin Heidelberg,, 2007, pp. 377 - 408 .

| M. J. Pazzani y D. Billsus., «Content-Based Recommendation Systems.,» de

20] *Lecture Notes in Computer Science*, vol. 4321, A. K. a. W. N. Peter Brusilovsky, Ed., Springer Berlin Heidelberg, 2007, pp. 325 - 341.

21] R. R. Sinha y K. Swearingen, «Comparing Recommendations Made by Online Systems and Friends,» de *Personalisation and Recommender Systems in Digital Libraries*, 2001.

22] N. F. Escobar Vinueza, «Data Mining Overview and Stream Mining Approaches,» Madrid, 2013.

23] Definicion.de, «Definición de contexto - Qué es, Significado y Concepto,» Definicion.de , [En línea]. Available: http://definicion.de/contexto/#ixzz2vkKN2E6U. [Último acceso: 12 03 2014].

24] G. Adomavicius y A. Tuzhilin, «Context-Aware Recommender Systems.,» In Francesco Ricci, Lior Rokach, Bracha Shapira and Paul B. Kantor, editeurs, Recommender Systems Handbook, 2011. [En línea]. Available: http://ids.csom.umn.edu/faculty/gedas/nsfcareer/CARS-chapter-2010.pdf. [Último acceso: 12 03 2014].

25] C. Palmisano, A. Tuzhilin y M. Gorgoglione, «Using Context to Improve Predictive Modeling of Customers in Personalization Applications,» *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 20, nº 11, 11 2008.

26] G. Adomavicius, R. Sankaranarayanan, S. Se y A. Tuzhilin, «Incorporating contextual information in recommender systems using a multidimensional approach,» *ACM Trans. Inf. Syst.,* vol. 23, nº 1, pp. 103 - 145, January 2005.

27] N. Kern y B. Schiele, «Context-Aware Notification forWearable Computing,» de *In Proceedings of the 7th IEEE International Symposium on Wearable Computers, ISWC*, Washington, DC, USA, IEEE Computer Society, 2003.

28] F. Ricci., «Mobile recommender systems,» *Information Technology & Tourism,* vol. 12, nº 3, pp. 205 - 231, 2010.

29] H. A. Tair, M. Zemerly, M. Al-Qutayri y M. Leida, «Architecture for Context-Aware Pro-Active Recommender System.,» *International Journal Multimedia and Image Processing (IJMIP),* vol. 2, nº 1/2, 2012.

O. Aritoni y V. Negru, «A Multi-Agent Recommendation System for Energy

30] Efficiency Improvement,» *Communications in Computer and Information Science,* vol. 171, pp. 156 - 170, 2011.

| M. C. P. Melguizo, L. Boves, A. Deshpande y O. M. Ramos., «A proactive
31] recommendation system for writing: helping without disrupting,» New York, NY, USA, In Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!, ECCE '07, ACM, 2007, pp. 89 - 95.

| Youtube, «Google Developers,» [En línea]. Available:
32] https://developers.google.com/youtube/1.0/developers_guide_python. [Último acceso: 3 2014].

| Google, «Google Code,» [En línea]. Available:
33] https://code.google.com/p/gdata-python-client/. [Último acceso: 2014].

| J. Clarke, «Google Code,» [En línea]. Available:
34] https://code.google.com/p/flickrpy/. [Último acceso: 2014].

| SoundCloud, «SoundCloud,» [En línea]. Available:
35] developers.soundcloud.com/docs. [Último acceso: 3 2014].

| Neo4j, «Download Neo4j,» [En línea]. Available:
36] http://www.neo4j.org/download. [Último acceso: 2 2014].

| T. P. S. Foundation, «The Python Software Foundation,» The Python Software
37] Foundation, 1990. [En línea]. Available: https://docs.python.org. [Último acceso: 2014].

| N. developers, «Numpy,» [En línea]. Available: http://www.numpy.org/.
38] [Último acceso: 2014].

| N. Small, «Py2Neo,» Created using Sphinx 1.2, 2011. [En línea]. Available:
39] http://book.py2neo.org/en/latest/. [Último acceso: 2014].