Algoritmo de recomendación sensible a contexto de elementos educativos reutilizables con almacenamiento Orientado a Grafos

Washington Adrián Velásquez Vargas

Escuela Superior Politécnica del Litoral - Gerencia de Tecnologías y Servicios de Información (GTySI)
wavelasq@gmail.com

Resumen. El proyecto consiste en la implementación en Python de un algoritmo de recomendación sensible al contexto basado en elementos educativos reutilizables, en donde estos recursos serán obtenidos a través del uso de las APIs de Youtube, Flickr y Soundcloud para lograr tener videos, imágenes y músicas respectivamente. En donde los elementos pasarán antes por un filtro de búsqueda que permita solo obtener ítems con carácter educativo, para luego ser presentados al usuario que está realizando la búsqueda, utilizando una base de datos orientada a grafos como motor de almacenamiento.

Palabras Clave: Recomendación Basada en Contexto, Filtrado Colaborativo, API Youtube, API Flickr, API SoundCloud, Interfaces de programación, Bases de datos Orientadas a Grafos, Neo4j, BDOG, NoSQL

1 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 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. Esto es sin duda una gran ventaja para los clientes, que encontrarán lo que necesitan de una forma más rápida, cómoda y fácil dentro las tiendas electrónicas en Internet y además les permiten descubrir nuevos productos o servicios que le puedan ser atractivos, que de otra manera les hubiese sido mucho más difícil o incluso imposible de encontrar.

La idea principal de este trabajo es que se pretende realizar un algoritmo de recomendación sensible al contexto bajo el seudónimo "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.

2 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". [1]

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. [2]

Los Sistemas de recomendación emergen como una iniciativa de investigación independiente a mediados de 1990, [3] [4] [5] 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 [6], 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.

2.1 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 (Fig. 1) 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.

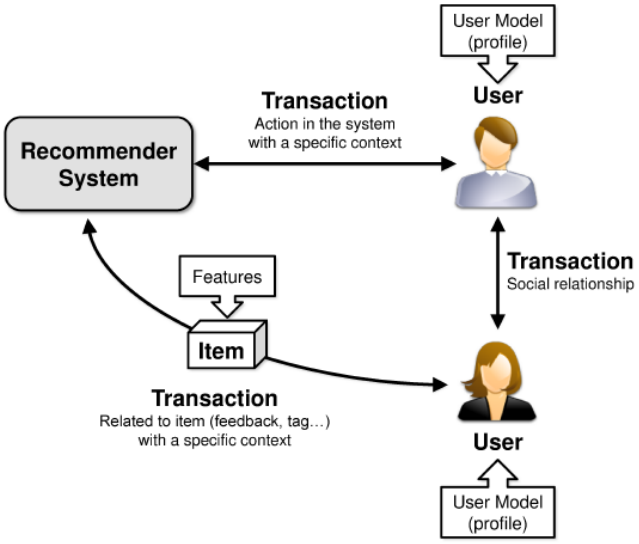

Fig. 1. Relación general entre artículos, usuario y transacciones [2]

Artículos. 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. [2]

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. [2]

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 (Fig. 1). 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.

2.2 Técnica 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 colaboración. Los principales se describen en las siguientes líneas que asisten a la taxonomía proporcionada por Burke [7], 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. [2]

2.2.1 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.

2.2.2 Basado en el 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.

Estos sistemas recomiendan artículos similares a un usuario en base a una descripción del elemento y un perfil de los intereses del usuario. [8] 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.

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 [9]:

- *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.

3 Interfaces de Programación de Aplicaciones

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:

3.1 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. [10] A continuación se muestra el código para realizar las búsquedas en 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()
```

```
            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 ""
```

3.2 Flickr

Flickr ofrece un paquete de librerías en Python para manejar su aplicación [11], 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.

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 muestra el código para realizar las búsquedas en Flickr:

```
#Busqueda en Flickr
photos = flickr.photos_search(tags=txtsch, tag_mode='all',
per_page=5)
for photo in photos:
        try:
            tagsc = ""
            for tag in photo.tags:
                    tagsc = tagsc + tag.text +" "
            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
```

3.3 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. [12]

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/

A continuación se muestra el código para realizar las búsquedas en SoundCloud:

```
#Busqueda en SoundCLoud
# create a client object with your app credentials
client =
soundcloud.Client(client_id='529295acc334c5fea23d339ac03300bc')
try:
    tracks = client.get('/tracks', q=txtsch, license='cc-by-sa')
    for track in tracks:
        var = "/tracks/"+ str(track.id)
        trackdef = client.get(var)
            try:
                print "<li><p>Title:  %s <br />" % (track.title)
                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 ""
```

4. Base de datos Orientada 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”.

4.1 Neo4j

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 (Fig.

2) se presenta un esquema de lo que vendría a ser “NEO4J”, como implementación en una base de datos orientada a grafos. [13]

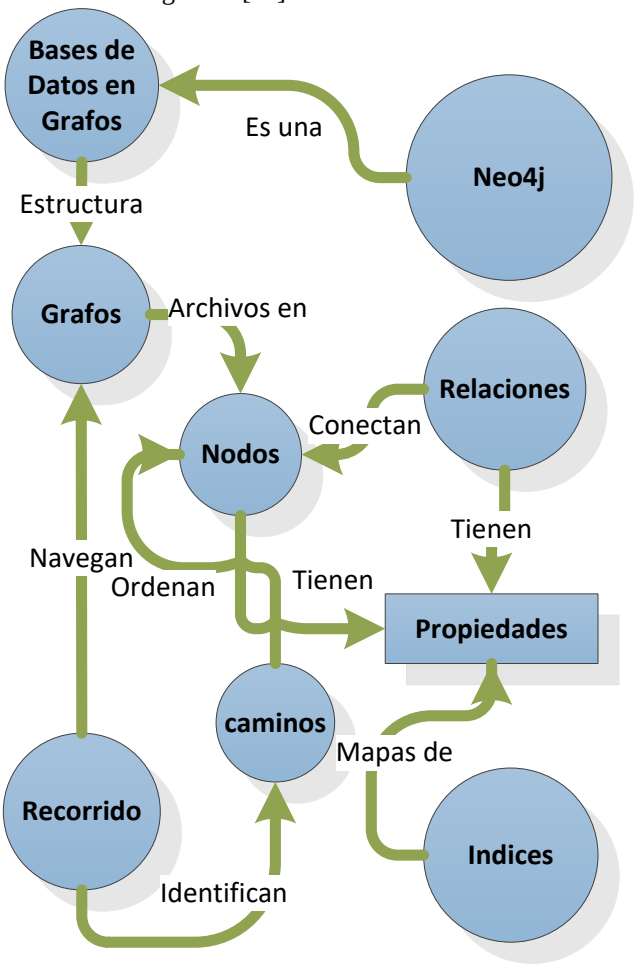

Fig. 2 Esquema del motor de base de datos Neo4j

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.

4.2 Modelamiento de los Objetos en la base

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.

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.

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 a continuación en la Tabla 1:

Tabla 1. Modelamiento de los Objetos en la Base de datos Neo4j

	Nodos
Video	**Propiedades:** title, url, datepublished **Query:** Create (v:Video {title:"",url:"",datepublished:""})
Fotos	**Propiedades**: title, url, tags, datetaken, idf **Query:** Create (i:Image {title:"",url:"",tags:"",datetaken:"",idf:""})
Música	**Propiedades:** title, url, tags, idm **Query:** Create (m:Music {title:"",url:"",tags:"",idm=""})
Usuarios	**Propiedades:** name, user, city, age, searchWord **Query:** Create(u:User{name:"",user:"",city:"",searchWord:""})

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 2 . Ejemplo del cálculo de la recomendación

	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

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

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

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. 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.

A continuación se muestra la implementación del algoritmo:

```
#!/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 Hibrido: Filtrado Colaborativo y Basado en el
Contexto
con = Conexion()
us = form.getvalue("user")
```

```
#Obtenemos las relaciones entre los elementos y los usuarios
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
#Obtenemos las correlaciones entre los elementos
    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]))
#Obtenemos un elemento de la matriz
    fila = random.randrange(ItemsUser.__len__())
    ItemsRecom = []
    for i in range(cosenos[fila].__len__()):
        if (cosenos[fila][i] > 0):
            ItemsRecom.append(Items[i])
#Items a recomendar
    ItemsRecomFinales = set(ItemsRecom) - set(ItemsUser)
```

6. Comentarios y Conclusiones

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 polígota.

El trabajo con bases de datos orientados 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.

Referencias

1. Adomavicius, G., & Tuzhilin, A. (2011). *Context-Aware Recommender Systems.* (In Francesco Ricci, Lior Rokach, Bracha Shapira and Paul B. Kantor, editeurs, Recommender Systems Handbook) Recuperado el 12 de 03 de 2014, de http://ids.csom.umn.edu/faculty/gedas/nsfcareer/CARS-chapter-2010.pdf
2. Burke, R. (2007). Hybrid Web Recommender Systems. En A. K. In Peter Brusilovsky (Ed.), *The Adaptive Web* (Vol. 4321, págs. 377 - 408). Springer Berlin Heidelberg,.
3. Clarke, J. (s.f.). *Google Code*. Recuperado el 2014, de https://code.google.com/p/flickrpy/
4. Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Commun. ACM.
5. Neo4j. (2013). *Documentación Oficial de Neo4j*. Recuperado el 12 de 2013, de http://www.neo4j.org
6. Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. En *Expert Systems with Applications* (Vol. 39, págs. 10059 – 10072).
7. Pazzani, M. J., & Billsus., D. (2007). Content-Based Recommendation Systems. En A. K. Peter Brusilovsky (Ed.), *Lecture Notes in Computer Science* (Vol. 4321, págs. 325 - 341). Springer Berlin Heidelberg.
8. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). In Proceedings of the 1994 ACM conference on Computer supported cooperative work. *GroupLens: an open architecture for collaborative filtering of netnews.* CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.
9. Ricci., F. (2010). Mobile recommender systems. *Information Technology & Tourism, 12*(3), 205 - 231.
10. Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating 'word of mouth'. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95* (págs. 210 - 227). New York, NY, USA,: ACM Press/Addison-Wesley Publishing Co.
11. SoundCloud. (s.f.). *SoundCloud*. Recuperado el 3 de 2014, de developers.soundcloud.com/docs
12. Vico, D. G. (2013). *Contribution to proactivity in mobile context-aware recommender systems.* Madrid: Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid - Tesis doctoral.
13. Youtube. (s.f.). *Google Developers*. Recuperado el 3 de 2014, de https://developers.google.com/youtube/1.0/developers_guide_python