Universidad del Valle de Guatemala
Facultad de Ingeniería
Algoritmos y Estructura de Datos
25/5/2018

Antonio Reyes #17273
Esteban Cabrera#17781
Miguel #17102

Proyecto 2

Algoritmo de Recomendación de Películas y Series

Documentación

El objetivo de este proyecto es el diseñar un algoritmo que permita realizar recomendación de películas y series de televisión. El problema que queremos resolver con este algoritmo es que muchos algoritmos se encargan solo de mostrar contenido que está seguro de que el usuario verá, no se trata que el usuario experimente y conozca el resto de contenido que provee el servicio de entretenimiento.

Con el fin de que las recomendaciones que se le hagan al usuario sean películas y programas que le gusten y de variedad de géneros, se utilizará una lista que servirá cono historial. Esta lista se encargará de determinar cuales son los géneros a los que se les dará prioridad para cuando se recomiende al usuario. Esta lista verá cuales son los géneros que más se repiten (debido a que la persona ve muchas películas y series con dicho género) y se buscarán en la base de datos películas y series que tengan por lo menos uno de estos géneros. Lo importantes de esta lista es que siempre se está actualizando, por lo que evoluciona junto con los gustos del usuario.

Métodos del programa:

def add_Excel():

Este método se encarga de agregar una base de datos de Excel en el grafo. Dento de esta base de datos una columna está reservada para el nombre de película o series y otras tres columnas para 3 géneros que posee cada película o serie. Se hace todo lo posible para que los géneros de lo que se vaya a agregar estén en orden alfabético por razones de orden.

```
def add_Excel():
    lista_gen = []
    for x in range(sheet.nrows):
        name = sheet.cell_value(x,0)
        gen1 = sheet.cell_value(x,1)
        gen2 = sheet.cell_value(x,2)
        gen3 = sheet.cell_value(x,3)

        lista_gen = []

        lista_gen.append(gen1)
        lista_gen.append(gen2)
        lista_gen.append(gen3)

        lista_gen.sort()

        gen1 = lista_gen[0]
        gen2 = lista_gen[1]
        gen3 = lista_gen[2]

        generos = []

        generos.append(gen1)
        generos.append(gen2)
        generos.append(gen3)

        database[name] = generos

        unidad = db.nodes.create(nombre=name, genero1=gen1, genero2=gen2, genero3=gen3)
        dataB.add(unidad)
```

Figura 1 muestra el código para agregar todos los valores de la base de datos al grafo.

Luego de agregar los nodos se realizarán las relaciones hacia los géneros mostrados en el resto del código de def add_Excel mostrados a continuación. Se intenta realizar la relación entre la película y gen1, gen2 y gen3. En el caso de que no se pueda (porque el nodo todavía no existe) se crea el género en el label Genero y se intenta hacer de nuevo la relación.

```
try:
    unidad.relationships.create("contains", gen.get(genero=gen1)[0])
    gen.get(genero=gen1)[0].relationships.create("contains", unidad)
except Exception:
    genNode = db.nodes.create(genero=gen1)
    gen.add(genNode)
    unidad.relationships.create("contains", gen.get(genero=gen1)[0])
    gen.get(genero=gen1)[0].relationships.create("contains", unidad)

try:
    unidad.relationships.create("contains", gen.get(genero=gen2)[0])
    gen.get(genero=gen2)[0].relationships.create("contains", unidad)
except Exception:
    genNode = db.nodes.create(genero=gen2)
    gen.add(genNode)
    unidad.relationships.create("contains", gen.get(genero=gen2)[0])
    gen.get(genero=gen2)[0].relationships.create("contains", unidad)

try:
    unidad.relationships.create("contains", gen.get(genero=gen3)[0])
    gen.get(genero=gen3)[0].relationships.create("contains", unidad)
except Exception:
    genNode = db.nodes.create(genero=gen3)
    gen.add(genNode)
    unidad.relationships.create("contains", gen.get(genero=gen3)[0])
    gen.get(genero=gen3)[0].relationships.create("contains", unidad)
```

add_Database():

Este método se encarga de ingresar elementos independientes al grafo en caso de que no haya una película o serie en la base de datos. Además de ingresar el nodo se realizarán las relaciones con los géneros. Al igual que en add_Excel(), en caso de que no exista un género, se crearán los géneros necesarios en el label Genero para poder realizar la relación.

def watch():

Este método no solo se encarga de hacer que los géneros de la película o serie se agreguen en la lista para la recomendación, también se encarga de dar el parámetro para el método de popular_topics() para mostrar al usuario cuales son los 5 géneros que más ha visto y realizar la recomendación.

```
def watch():
    name = raw_input("Ingrese el nombre de la Pelicula o Serie: ")

    try:
        query = "MATCH (n:Database) WHERE n.nombre='"+name+"' RETURN n.genero1, n.genero2, n.genero3"
        results = db.query(query, data_contents=True)
        a = results.rows
        for x in a:
            historial.append(x[0])
            historial.append(x[1])
            historial.append(x[2])



    except Exception:
        print("No se encuentra en la base de datos, si desea agregarlo elija la opcion 1")

    popular_topics(name)
```

def popular_topics(name):

Este método se encarga de mostra los 5 métodos más vistos y recomendar películas y series, la recomendación empezará desde el nodo que fue ingresado en el método watch(). La manera para determinar cuáles son los géneros que más se repiten es recorriendo toda la lista

utilizando un ciclo for donde en caso de que no se encuentra la palabra en el diccionario, agregarla como llave y darle un valor de 1, en caso de que si se encuentre la palabra como llave, sumar 1 al valor que posee.

Luego se ordena el diccionario utilizando *sorted(word_counter, key = word_counter.get, reverse = True) y se elijen las primeras 5 categorias (las 5 categorías que se repiten más veces).*

```
def popular_topics(name):
    nombre = name
    #diccionario que determinará cuales son los 5 generos más vistos
    top_5 = []
    #por cada genero en la lista....
    word_counter = {}
    for word in historial:
        if word in word_counter:
            word_counter[word] += 1
        else:
            word_counter[word] = 1

    popular_words = sorted(word_counter, key = word_counter.get, reverse = True)

    top_5 = popular_words[:5]

    #se ordenan los generos en orden alfabetico
    lista = []
    print "Los generos que más miras son: "
    for x in top_5:
        lista.append(x)
        print x
```

Por último, para poder recomendar las películas y series, se tendrá que utilizar el código (manera que debe escribirse en neo4j Desktop)

match (n:Database{nombre:'"+**nombre**+"'})-[:contains*1..3]->(a:Database{**generoX**:'"+**top_5[generos principales]**+"'}) return collect(distinct a.nombre).

donde nombre es la película o serie que vió el usuario (desde donde se empezará a buscar), generoX es si se quiere buscar en generos1, genero2 o genero3 y top_5[] tendrá la posición cero (genero que más se ve) , uno (2do) o dos (3ero).

Esta búsqueda se realizara para los tres género que más se repiten y la búsqueda para los tres será en generos1, generos2 y generos3.

def show_genre():

Se encarga de mostrar todos los nodos que poseen una relación con un género en específico.

```
#método para mostrar todas las series y peliculas de un genero
def show_genre():
    genre = raw_input("Ingrese el género: ")
    try:
        query = "MATCH (n:Database {genero1:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print x

    except Exception:
        pass

    try:
        query = "MATCH (n:Database {genero2:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print x

    except Exception:
        pass

    try:
        query = "MATCH (n:Database {genero3:'"+genre+"'}) RETURN n.nombre"
        results = db.query(query, data_contents=True)
        a = results.rows
        b = []
        for x in a:
            if x not in b:
                b.append(x)
                print x
```