Recomanador de Gustos Musicals basat en les interaccions entre usuaris

Andoni Carmona Chavarría

Resum—L'objectiu principal d'aquest treball consisteix en implementar un sistema recomanador a la base de dades de l'app mòbil de la sala Apolo, incorporant una manera de veure-ho a l'app i l'algorisme en si. Per a fer això s'han estudiat els diferents tipus de recomanadors i s'ha implementat el més apropiat per a aquesta base de dades, també se li ha afegit la possibilitat a l'usuari de determinar els seus gustos musicals mitjançant la seva compta d'Spotify. Finalment s'han fet proves per a determinar dintre del possible si l'algorisme és efectiu i coherent amb les recomanacions.

Paraules clau—Recomanador, musical, etiquetes, col·laboratiu, MongoDB, Spotify, Apolo, Relatiooon, Android

Abstract— The principal objective of this work consists in implementing a recommender system to the Apolo's mobile app database, adding a way to see it in the app and the algorithm itself. To do so the diferent types of recommenders have been studied and the most apropiate has been implemented, also It has been added the possibility to determinate user's musical genres from his Spotify account. Finally some tests have been doned to determinate if the algorithm is effective and coherent with its recommendations.

Index Terms—Recommender, musical, tags, colaborative, MongoDB, Spotify, Apolo, Relatiooon, Android

1 Introducció

EL món de la música és un extens terreny que ha anat evolucionant molt en aquests darrers anys. No fa més de 15 anys la nostra col·lecció musical consistia en un conjunt de cintes o CDs i l'unica manera d'ampliar el nostre repertori en cerca de nous descobriments musicals era la recomanació de coneguts. Avui en dia aquest àmbit igual que molts altres ha canviat dràsticament gràcies a l'internet i l'evolució dels dispositius personals. Ara aquell diskman o reproductor de cassetes s'ha vist substituït pel nostre mòbil, i per descobrir música nova ja no preguntem tant als nostres coneguts sinó que busquem a veure que ens recomana Spotify o Youtube.

Les recomanacions que donen aquests serveis però no ve donada per art de màgia, els algorismes d'intel·ligència artificial o IA és un camp que ha anat guanyant terreny aquests anys. És mitjançant aquests algorismes que podem obtindré recomanacions bastant ajustades als nostres gustos.

En aquest treball es pretén implementar un algorisme el qual permeti recomanar esdeveniments de la Sala Apolo als usuaris de la seva App mòbil tenint en compte els seus gustos musicals.

- *E-mail de contacte: andoni.carmona@e-campus.uab.cat*
- *Menció realitzada: Enginyeria Computació.*
- *Treball tutoritzat per: Josep Llados (Departament de Ciències de la Computació)*
- *Curs 2015/16*

Per a fer això primer s'ha implementat un algorisme de recomanació de gustos musicals que consisteix en un servidor en python que crea una matriu d'usuaris semblants i un ranking d'esdeveniments més populars i després quan es demanen els esdeveniments recomanats d'un usuari (via l'app mòbil) aquest utilitza la matriu i el ranking d'esdeveniments per a donar recomanacions, junt amb els estils de música que l'usuari ja ha indicat que són de la seva preferència.

Un cop implementat l'algorisme s'ha provat la seva eficàcia amb un seguit d'usuaris creats específicament per a la tasca amb un altre algorisme creat per a fer les proves.

Finalment s'ha integrat la possibilitat de rebre recomanacions a l'app d'Apolo per a poder veure'n els resultats de l'algorisme d'una manera més "user-friendly".

La resta d'aquest article està organitzat de la següent manera:

- A la secció 2 s'especifiquen els objectius d'aquest treball o en altres paraules fins a on pretén abastarde .
- A la secció 3 s'explica una mica quina és la situació dels algorismes i de l'aplicació mòbil de la qual es parteix.
- A la secció 4 es detalla la metodologia utilitzada per a dur a terme aquest treball.

- A la secció 5 s'explica la integració i implementació que s'ha dut a terme per a crear el sistema.

- A la secció 6 es mostra detalladament la part de la base de dades d'Apolo amb la qual es treballarà i la seva estructura.

- A la secció 7 es detalla com és l'algorisme implementat per a recomanar.

- A la secció 8 es mostren les proves realitzades amb l'algorisme per tal de comprovar el seu comportament.

- A la secció 9 es detalla també la integració d'Spotify.

- Finalment a la secció 9 s'expliquen les conclusions a les quals s'ha arribat després de realitzar tot aquest treball.

2 OBJECTIUS

L'objectiu principal d'aquest treball és aconseguir recomanar automàticament als usuaris sessions o concerts de la sala Apolo de Barcelona d'acord amb els seus gustos musicals. Per a fer això s'han marcat uns objectius específics a seguir per a l'acompliment de l'objectiu final:

- Explorar els diferents tipus d'algorismes de recomanació existents i escollir-ne el més adient.

- Implementar un algorisme basat en la similitud d'usuaris o per etiquetes en conseqüència a l'exploració feta prèviament.

- Provar l'algorisme i afegir-li millores si escau

- Validar l'algorisme fent proves amb diferents usuaris bastant tipats en un tipus de música.

- Afegir una interfície visual poder veure les recomanacions de l'algorisme.

3 ESTAT DE L'ART

L'empresa Omitsis Consulting S.L. creadora del producte *Relatiooon* [1] vol afegir-hi un recomanador de gustos musicals.

Relatiooon consisteix en una solució genèrica per a espais escènics la qual pretén obtindré un millor coneixement sobre el públic de l'espai a la vegada que proporciona una experiència personalitzada als usuaris.

Actualment aquest sistema consisteix en una App per als clients la qual dóna informació als usuaris dels esdeveniments pròxims de l'espai, ofertes personalitzades per a cada usuari i també mitjançant Bluetooth i beacons trackeja a l'usuari dins del recinte, obtenint així informació dels esdeveniments als quals assisteix.

A part l'app també permet a l'usuari indicar els seus gustos musicals i marcar esdeveniments com a favorits.

Relatiooon actualment està implementada com a prova pilot a la Sala Apolo sota el nom de Appolo. Per tant aprofitarem aquesta prova pilot per a fer el treball.

Actualment hi ha empreses que ja implementen un recomanador de gustos musicals, a esmentar està Spotify o Deezer, la primera essent principalment una plataforma de streaming de música amb un component secundari de recomanació de música (té una funció radio, la qual a partir d'una cançó, artista o playlist ens reprodueix cançons similars) i la segona essent més enfocada a la recomanació de música nova en funció de les que l'usuari escolta.

Ambdues utilitzen recomanadors en funció de similitud d'usuaris (usuaris que escolten normalment el mateix s'assemblen bastant) però cap d'elles recomana enfocant-se a esdeveniments o concerts que és el que es pretén aconseguir en aquest projecte.

3.1 Recomanadors

En aquest treball quan parlem de recomanadors ens referim als algorismes "intel·ligents" que permeten mostrar a l'usuari els ítems més rellevants per a ell.
Per a fer això hi ha diferents tipus de recomanadors tenint en compte la metodologia que fan servir per a aconseguir trobar aquests ítems. Els tipus més rellevants són els recomanadors de filtratge col·laboratiu [2] i els de filtratge de contingut[3]:

3.1.1 Recomanadors de filtratge col·laboratiu

Aquests recomanadors basen el seu filtratge o recomanacions en la classificació que fan els usuaris, és a dir, els usuaris indiquen els ítems que els hi "agraden" i aleshores l'algorisme troba similituds entre usuaris.

D'aquesta manera si dos usuaris tenen X ítems en comú el més probable és que els ítems que no tenen en comú també puguin ser del gust de l'altre.

A més aquests algorismes es poden enriquir si el sistema té en compte no només la rel·lació usuari-usuari sinó que implementant una capa social l'algorisme pugui tindre en compte la relació "d'amics" com un afegit a l'hora de trobar similituds.

3.1.2 Recomanadors de filtratge de contingut

Aquests recomanadors es basen en l'etiquetatge d'objectes. Els objectes tenen X atributs (en el cas dels gustos musicals podria ser l'estil de música d'un esdeveniment) i als usuaris els hi agraden Y atributs.

Els atributs que els hi agraden a l'usuari poden aconseguir-se explícitament demanat-li aquests atributs a l'usuari (preguntant-li quin tipus de música prefereix) o implícitament depenent dels ítems que li "agradin".

Aleshores les recomanacions es fan a partir dels atributs que li agrada a cada usuari.

3.1.3 Solució proposada a partir de l'exploració

La solució proposada en aquest treball com es veurà més endavant consisteix en una barreja d'ambdós tipus de recomanadors.
Es dóna prioritat a la recomanació basada en col·laboratiu i posteriorment la de filtratge de contingut, així si l'usuari no ha afegit cap favorit pot obtindré recomanacions igualment.

4 METODOLOGIA

Aquest projecte s'ha realitzat segons les dates previstes de la planificació de l'assignatura i de la proposada en aquest treball mitjançant els informes. S'ha seguit un procés en cascada, en el que primer s'han après les bases per a poder realitzar l'algorisme (mitjançant la cerca a internet i també aprenent eines necessàries per al desenvolupament de la solució com ara el sistema MongoDB).

Posteriorment un cop acabat la fase d'exploració s'ha seguit amb la implantació de la sol·lució la qual ha requerit primer la creació del sistema en si (màquina virtual amb la base de dades, server en python) i posteriorment del desenvolupament de l'algorisme.

Finalment s'han fet unes quantes proves amb usuaris creats per un algorisme per a validar si les recomanacions de l'algorisme eren suficientment ajustades.

4.1 Eines utilitzades

Com a eines utilitzades s'ha fet servir:

- PyCharm[4]: És un IDE per al desenvolupament en python. Utilitzat en aquest treball per a desenvolupar l'algorisme recomanador i el server http que espera les peticions de l'app.

- MongoChef[5]: És un client de MongoDb. Utilitzat en aquest treball per a explorar i fer consultes a la base de dades d'Apolo.

- Android Studio [6]: És un IDE específic per al desenvolupament d'aplicacions natives d'Android en java. S'ha fet servir per adaptar l'app d'Apolo per a permetre rebre recomanacions a l'App.

- Virtual Box [7]: És un entorn per a crear màquines virtuals. Utilitzada per a crear una màquina virtual d'Ubuntu Server per a allotjar la base de dades d'Apolo.

5 INTEGRACIÓ I IMPLEMENTACIÓ DE LA SOLUCIÓ

5.1 Implantació de la solució

Per a què la solució funcionés es va haver primer d'extreure la base de dades d'Apolo amb les dades dissociades.

Per a fer això es van eliminar les dades personals dels usuaris i es van afegir en una màquina virtual amb Ubuntu Server com a sistema operatiu.

Un cop fet això es va crear l'algorisme en python i junt amb l'algorisme un servidor amb python per a poder rebre peticions de recomanacions.

5.2 Servidor desenvolupat

L'algorisme desenvolupat s'ha integrat en un servidor en python per a poder comunicar-se amb l'app mòbil.

Per a fer això el server escolta peticions http que fa l'app. La petició té el següent esquema:

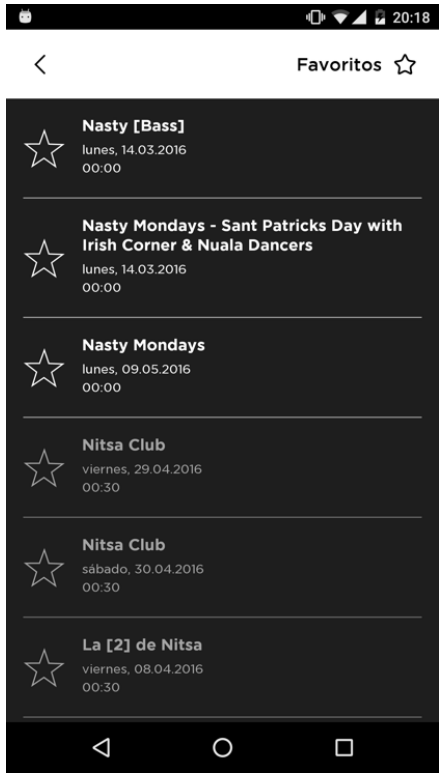
GET /recommend/?user=576060ad59eea47bace6a721 HTTP/1.1

El server parseja peticions d'aquest tipus per a poder parsejar la id de l'usuari, executa l'algorisme explicat a l'apartat 8.1.2. Un cop executat l'algorisme, aquest converteix els esdeveniments a retornar a format JSON [6] i envia una resposta http amb el json com a contingut.

5.3 Integració a l'app del recomanador

Per afegir el recomanador a la app mòbil s'ha adaptat la pantalla de favorits per a poder mostrar també les recomanacions i veure-les en detall des de l'app mòbil.

Per fer això s'han afegit les recomanacions a la llista de favorits amb transparència per a donar a entendre a l'usuari quins són els seus favorits i quines les recomanacions com es pot veure a la imatge d'abaix.



6 ESTRUCTURA INTERNA BASE DE DADES

Per a poder implementar la solució adequada primer s'ha estudiat la base de dades.
La base de dades d'Apolo és una base de dades de tipus orientada a objecte (OO) en llenguatge Mongo. Conté les següents taules:

- Events: Aquesta taula conté tots els esdeveniments que s'han realitzat o es realitzaran a l'Apolo. Com es pot mostrar a la imatge els elements que importen per a l'algorisme són els tipus de música, el nom i un camp afegit per a la realització d'aquest treball que és el número de likes (gent que té l'esdeveniment a favorits).

_id	music_types	name	num_likes
5759d0dd59...	[2 elements]	Pop_event_6	6
5759d0dd59...	[1 elements]	Pop_event_7	3
5759d0dd59...	[1 elements]	Pop_event_8	8
5759d0dd59...	[1 elements]	Pop_event_9	8
5759d0dd59...	[2 elements]	Alternativa_ev...	6
5759d0dd59...	[2 elements]	Alternativa_ev...	3
5759d0dd59...	[2 elements]	Alternativa_ev...	6

- Users: Aquesta taula conté tots els usuaris de la sala Apolo (la versió amb la qual s'ha realitzat aquest treball però ha estat modificada per a no incloure dades personals). Com es pot observar a la imatge els elements que ens importen per a l'algorisme són els tipus de música i el nom.

_id	music_types	name
5759d09859eea44f2496470c	[1 elements]	Blues1
5759d09859eea44f2496470b	[1 elements]	Blues0
5759d09859eea44f2496470a	[1 elements]	Latina9
5759d09859eea44f24964709	[1 elements]	Latina8
57572eda59eea435a79e0758	[1 elements]	Experimental3
57572eda59eea435a79e0757	[1 elements]	Experimental2
57572eda59eea435a79e0756	[2 elements]	Experimental1

- User_events: Aquesta taula conté la relació dels usuaris amb els esdeveniments (els esdeveniments que té cada usuari com a favorit).

_id	event_id	user_id
575d5bdf59e...	575d5bdf59e...	575d5bdf59e...
575d5bdf59e...	575d5bdf59e...	575d5bdf59e...
575d5bdf59e...	575d5bdf59e...	575d5bdf59e...
575d5bdf59e...	575d5bdf59e...	575d5bdf59e...
575d5bdf59e...	575d5bdf59e...	575d5bdf59e...

- Music_types: Aquesta taula conté els diferents tipus de música existents dins la base de dades.

També la base de dades compta amb altres taules que no són d'ús de l'algorisme com les notificacions les promocions i les relacions usuari-promocions i usuari-notificacions.

7 ALGORISME PROPOSAT

L'algorisme proposat com hem esmentat abans és una barreja de filtratge per contingut i filtratge col·laboratiu.
S'aprofiten per això l'atribut music_types de la taula users que indica els tipus de música que li agrada a l'usuari i també la similitud entre dos usuaris per a fer les recomanacions.

7.1 Estructura de l'algorisme

L'algorisme es basa en dues parts, la primera serien mètodes que degut al seu cost computacional s'haurien de cridar de manera periòdica per anar actualitzant les recomanacions de l'algorisme i la segona part l'algorisme que s'executa quan l'app mòbil demana les recomanacions al server.

7.1.1 Metodes per actualitzar les recomanacions

Aquests mètodes com ja hem esmentat abans es criden cada X temps per anar actualitzant les recomanacions, ja que seria inviable executar-los amb cada crida que fes l'app al servidor. Aquests mètodes es dediquen a actualitzar bàsicament les relacions de semblança entre usuaris i la "popularitat" de l'esdeveniment (quantitat d'usuaris que tenen un esdeveniment a favorits).

Update user matrix

Aquest mètode s'encarrega d'actualitzar la matriu de semblança d'usuaris. Aquesta matriu es guarda com a diccionari de diccionaris (un diccionari per a cada usuari amb els usuaris amb els quals guarda similitud com a key i com a value un coeficient de semblança).

Per a actualitzar aquesta matriu la funció recorre cada usuari amb cada usuari i mira a la BD les coincidències de favorits (si li han donat favorit ambdós a un mateix esdeveniment) i les coincidències d'estil musical, i les puntua actualment amb 1 punt per a esdeveniment coincidit i amb 2 punts per a coincidències d'estil musical. D'aquesta manera s'obté una matriu d'usuaris amb usuaris i una puntuació de "semblança".

Aquesta matriu de semblança es guarda doncs a un fitxer en format binari utilitzant la llibreria pickle de python.

En pseudocodi simplificat seria:

```
For usuari in usuaris:
  For usuari2 in usuaris:
    For events in usuari.events:
      For events2 in usuari2.events:
        If event == event2:
          Matriu[usuari][usuari2]+= 1
        FiIf
      FiFor
    FiFor
    For tipus_musica in usuari.tipus_musiques:
      For tipus_musica2 in usuari2.tipus_musiques:
        If tipus_musica == tipus_musica2:
          Matriu[usuari][usuari2]+=2
        FiIf
      FiFor
    FiFor
  FiFor
FiFor
```

Update likes on events

Aquest mètode s'encarrega d'actualitzar el camp user_likes dels esdeveniments a la taula events. L'algorisme recorre per cada esdeveniment totes les interaccions de favorits que conté la base de dades relacionades amb ell (instàncies a user_events amb el seu event_id) i en fa un recompte. Aquest recompte es guarda posteriorment a cada objecte event en un camp anomenat user_likes.

Gràcies a aquesta funció podem ordenar les recomanacions no només per semblança entre usuaris sinó que també per popularitat en general de l'esdeveniment.

7.1.2 Tractament de la sol·licitud de recomanació

Per a enviar les recomanacions a l'app mòbil quan aquesta ho sol·licita mitjançant una crida http, l'algorisme executa una sèrie de mètodes per a poder donar les recomanacions.

Per a donar recomanacions l'algorisme primer mira a la matriu d'usuaris amb quins usuaris guarda alguna similitud i agafa suggeriments d'usuaris més semblants a menys semblants i dintre d'aquests d'esdeveniment més popular a menys. Si tot i així no arriba a cobrir 10 recomanacions, l'algorisme busca als gustos musicals del nostre usuari i recomana esdeveniments que tinguin aquell tipus de música.

Per a fer tot aquest procés comptem amb els següents mètodes:

Find suggestions with user matrix

Aquesta funció utilitza la matriu d'usuaris per a fer recomanacions.

Per fer això primer l'algorisme busca dintre de la matriu l'usuari per al que es volen les recomanacions i se'n agafen els usuaris semblants.

Un cop extrets aquests usuaris (en forma de diccionari amb clau l'usuari i valor el coeficient de semblança) s'ordenen en funció del coeficient de semblança (de coeficient més gran a més petit).

Aleshores mentre l'algorisme no tingui el número de recomanacions demanada (10 per defecte) va buscant usuari a usuari semblant i crida a la funció "Find user likes" la qual busca els likes d'un usuari (en aquest cas de l'usuari semblant per a aconseguir els seus likes), a aquesta funció també li passem els esdeveniments que ja té el nostre usuari a favorits i els que ja estan a la llista de recomanacions que li retornarem per tal de no repetir-nos i també el número de suggeriments restant.

Si no tenim prous usuaris semblants i per tant no podem obtenir suficientes recomanacions l'algorisme cridarà a la funció "Find suggestions for" que buscarà esdeveniments dels mateixos estils musicals que els que l'usuari ha indicat que li agraden a l'hora de registrar-se a l'app (guardats a la base de dades com a "music_types"). A l'igual que amb "Find user likes" aquesta funció permet passar-li els esdeveniments ja recomanats o ja a favorits de l'usuari i les recomanacions que falten.

En pseudocodi seria:

```
Usuaris_semblants = Matriu[usuari]
U_semblants_ordenats = Ordenar (Usuaris_semblants)
i = 0
While Suggerencies.lenght < 10:
  Sugg_left = 10 – Suggerencies.length
  If U_semblants_ordenats.lenght > i:

      Suggerencies.append(find_user_likes(U_semblants_or
      denats[i], usuari.events+Suggerencies, Sugg_left ))
  Else:
      Suggerencies.append(find_suggestions_with_music_
      types(usuari, usuari.events+Suggerencies,Sugg_left ))
  FiElse
FiWhile
```

Find user likes

Aquesta funció busca els favorits d'un usuari passat per paràmetre, també opcionalment se li pot passar per paràmetre els esdeveniments que no es volen trobar i la quantitat màxima d'esdeveniments a retornar.

Per trobar els favorits d'un usuari es fa una consulta a la taula user_events, buscant que la user_id coincideixi amb la id de l'usuari i que l'event_id no estigui dintre de les ids d'esdeveniment passades per paràmetre. Un cop trobades les ids dels esdeveniments es busquen aquests esdeveniments ordenats per la quantitat de gent que el té a favorits, essent els més populars primers.

Find suggestions with music types
Aquesta funció busca esdeveniments que tinguin uns tipus de música concreta, també a l'igual que find user likes opcionalment se li pot passar per paràmetre els esdeveniments que no es volen trobar i la quantitat màxima d'esdeveniments a retornar.

Per trobar els esdeveniments es fa una consulta a la base de dades buscant els esdeveniments que tinguin un o més dels tipus de música especificats per paràmetre i no estiguin continguts en les ids d'esdeveniment passades per paràmetre. Un cop trobades també s'ordenen per la quantitat de gent que els té a favorits, essent els més populars primer.
Si es clica a qualsevol dels esdeveniments (recomanats o ja a favorits) s'obre una pantalla amb tots els detalls de l'esdeveniment.

Aquesta integració és bastant transparent i adaptada a l'app.

7.2 Integració D'Spotify

Per a millorar les recomanacions, sobretot les dels nous usuaris que no tenen favorits, s'ha integrat la possibilitat de trobar estils musicals que li agradin a l'usuari a través d'spotify.

7.2.1 Llibreries utilitzades
Per tal de fer l'autentificació amb Spotify i després la crida a l'api per a aconseguir els artistes preferits s'ha utilitzat una sèrie de llibreries per tal d'estalviar temps i línies de codi:

- Spotify Android SDK[8]: Aquesta llibreria desenvolupada per Spotify ens permet loguejar-nos amb Spotify i aconseguir el token de l'usuari (codi alfanumèric que permet autentificar que es X usuari des de la nostra app i fer les crides posteriors a l'api).

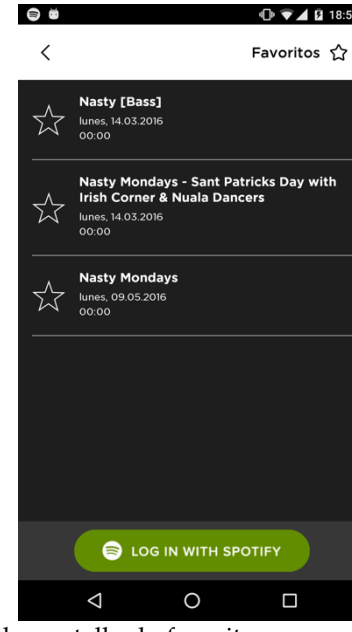
- Spotify Web Api Wrapper [9]: Aquesta llibreria consisteix en un wrapper que ja conté totes les classes necessàries per a fer les crides a l'api (un cop obtenim l'access token) i les classes dels dtos (Data Transfer Objects) de l'api com ara els artistes.

7.2.2 Criteri per a extreure els estils músicals

Per a fer això primer s'ha mirat l'api d'Spotify [10] en la qual es detalla el que se'n pot extreure com a desenvolupadors de la base de dades d'Spotify.

Finalment s'ha decidit que per a mirar els gustos musicals es miraran els generes dels 50 artistes més escoltats per l'usuari, ja que Spotify només permet veure els autors i albums més escoltats i també perquè les pistes d'audio en si no tenen genere, només ho tenen els artistes.

7.2.3 Implantació a l'app i el server
Per a fer que l'usuari sigui capaç de loguejar-se amb Spotify i permetre així a l'app llegir els artistes més escoltats i determinar-ne els gustos musicals de l'usuari s'ha afegit



un botó a la pantalla de favorits que convida a l'usuari a loguejar-se amb Spotify.

Quan l'usuari clica en aquest botó serà dirigit a una pantalla per a autoritzar a l'app d'Apolo per a accedir als seus artistes preferits i finalment se li retornarà a la pantalla de favorits on es fa l'app fa la crida i determina els gustos musicals de l'usuari.

Un cop determinats mostra un missatge en un Dialog [11] el qual informa a l'usuari els gustos que ha determinat a partir d'Spotify i li demana confirmació per a utilitzar-los en les recomanacions.

Si l'usuari accepta finalment es fa una crida al nostre servidor per a afegir els gustos musicals a l'usuari dins la base de dades.

7.2.4 Parsejament dels gustos musicals
Per a determinar quins són els gustos musicals (dintre dels acceptats per la nostra app els qual són 28) de l'usuari d'entre tots els que Spotify conté (994 generes musicals [12]) s'ha hagut de fer un parsejament.

Així tots els generes que continguin pop dintre dels generes d'Spotify es cataloguen com a pop, etc. Per exemple:

If genere conté "pop":
Retornar 16 (id del pop a la base de dades)
Fi If
If genere conté ...

8 PROVES DE VALIDACIÓ

Per a validar que l'algorisme dóna resultats dintre del que cap coherents s'han fet unes proves de validació.

S'ha creat un altre algorisme que crees usuaris i esdeveniments ja amb un tipus bastant definit per a fer les proves de validació.

8.1 Algorisme de validació

Aquest algorisme de validació es dedica a crear usuaris que tenen bastanta "preferència" per un tipus de música i els esdeveniments dels diferents tipus de música. Per no fer usuaris de tots el tipus de música per a la nostra prova hem fet servir 5 tipus de música i hem creat 10 usuaris i 10 esdeveniments per a cada tipus de música (fent un total de 50 usuaris i 50 esdeveniments)

8.1.1 Creació d'usuaris

Per a la creació d'usuaris l'algorisme recorre cada tipus de música i va creant 10 usuaris a la base de dades amb el nom "[Tipus_de_musica][num]" (exemple: Pop2) a l'hora de crear-los se'ls hi assigna el tipus de música dintre dels seus music_types i amb una probabilitat del 30% se'ls hi afegeix un segon tipus de música totalment aleatori als seus estils de música. Les ids de la base de dades de tots els usuaris creats es guarden en un diccionari de llistes, en la que cada llista correspon a les ids dels usuaris d'un tipus de música (i les claus del diccionari per tant són els diferents estils de musica).

8.1.2 Creació d'esdeveniments

A l'hora de crear esdeveniments es fa un procés similar al de crear usuaris, es creen 10 esdeveniments a la base de dades amb el nom "[Tipus_de_musica]_event_[num]" (exemple: Pop_event_2) i se'ls assigna el tipus de música dintre els seus music_types, també amb una probabilitat del 30% se'ls hi afegeix un segon tipus de música totalment aleatori. A l'igual que amb els usuaris els esdeveniments són guardats en un diccionari de llistes en la que cada llista correspon a les ids dels esdeveniments d'un tipus de música.

8.1.3 Creació d'interaccions Usuaris – Esdeveniments

Un cop tenim els usuaris i els esdeveniments creats l'algorisme simula afegir "favorits" dels usuaris creats als esdeveniments.

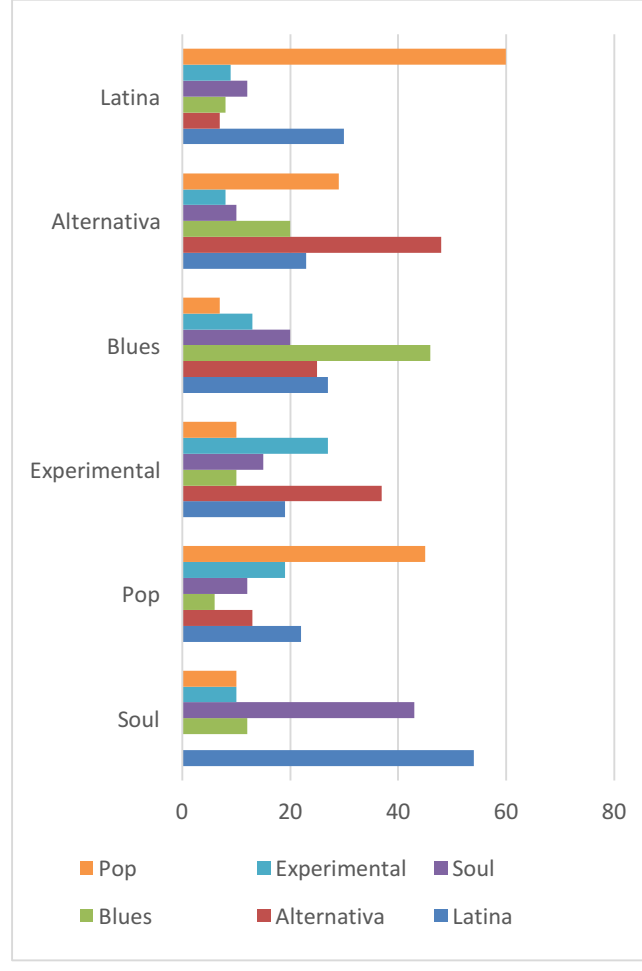
Per fer això recorrem per cada tipus de música els usuaris amb els esdeveniments d'aquell tipus i aleatòriament amb un 50% de probabilitats es fa l'esdeveniment favorit per a l'usuari o no.

Finalment per a cada usuari es creen de 1 a 5 "favorits" completament aleatoris.

8.2 Proves realitzades

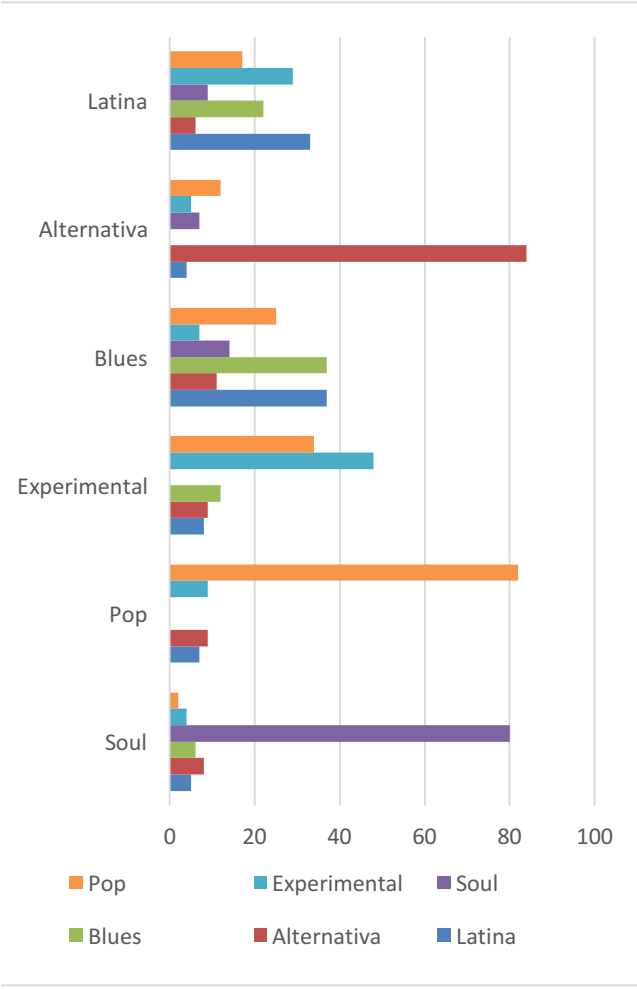
Amb l'algorisme esmentat a l'apartat anterior s'han fet unes proves de validació per tal de corroborar dintre del possible (ja que aquest algorisme es basa en les recomanacions entre usuaris).

Primer s'han fet proves amb 5 tipus de música, creant 10 usuaris de cada tipus i 10 esdeveniments de cada tipus amb una probabilitat del 30% que cada esdeveniment o usuari tingues un segon tipus de música, també a l'hora de crear els favorits de cada usuari amb un 50% de probabilitat per a cada esdeveniment del mateix tipus i també la possibilitat d'afegir fins a 5 favorits completament aleatoris. Els resultats han estat els següents:



En aquest gràfic es mostren de colors els favorits agrupats per tipus d'usuari, en altres paraules les barres agrupats a "Latina" mostren la distribució de favorits dels usuaris creats que tenen "preferència" per la música Llatina.

Podem observar que efectivament predomina la majoria de vegades el tipus de música amb la distribució de favorits, excepte en casos com la música Soul o Latina. Això és degut a que com la majoria d'esdeveniments del mateix tipus de música que el de preferència de l'usuari ja estàn a preferits, l'algorisme intenta recomanar aquells esdeveniments que l'usuari no té ja a favorits, per tant és normal que recomani bastant altres tipus de música en altres ocasions. Si per exemple canviem la probabilitat de fer un esdeveniment del mateix tipus de música del 50% al 30% veiem que al tenir més esdeveniments per a recomanar recomana els del mateix tipus amb més freqüència:



En aquest cas si que hi predomina bastant més notablement el tipus de música amb les recomanacions de l'algorisme.

Si provem de treure la probabilitat que s'afegeixin altres tipus de música o altres "favorits" d'un altre tipus de música diferent del seu preferent trobem que efectivament l'algorisme només recomana esdeveniments del mateix tipus de l'usuari.

8.3 Conclusions envers les proves realitzades

Després d'aquestes tres proves podem veure que l'algorisme efectivament funciona i ens dóna uns resultats coherents depenent del tipus d'usuaris que tinguem a la nostra base de dades i de les seves interaccions (els seus favorits) amb ella.

Ja que quan teníem els usuaris amb la majoria dels esdeveniments del seu tipus de música preferit a favorits aquest intentava recomanar-nos els esdeveniments que algun usuari tenia d'un altre tipus de música (degut a l'aleatorietat que li hem ficat a l'algorisme de validació), que un cop tenia més esdeveniments del mateix tipus de música que poder recomanar ho feia (segona prova) i finalment que si treiem l'aleatorietat a l'hora d'afegir-hi altres tipus de música als esdeveniments, als usuaris i als preferits només recomanava esdeveniments del mateix tipus de música.

9 CONCLUSIONS

Fer un projecte per a empresa m'ha aportat bastants avantatges però també alguns inconvenients. Com a avantatges el suport no només del tutor del TFG, sinó també del tutor de l'empresa, el qual s'ha involucrat i ajudat en el que ha pogut. D'altra banda com a desavantatges el fet que sigui un projecte proposat per mi a través de l'empresa m'ha portat treball extra, ja que no era un projecte amb uns objectius definits, vaig partir d'una idea i l'he anat desenvolupant com he sabut.

També és interessant desenvolupar un projecte que posteriorment podrà ser utilitzat per bastanta gent, ja que es veu una sortida real al projecte que he estat desenvolupant.

D'altra banda com a aprenentatge aquest TFG m'ha proporcionat coneixements de MongoDb, ja que m'ha obligat a aprendre uns mínims d'aquest sistema de base de dades orientat a objectes, també d'aprofundir més en el llenguatge python i descobrir-ne noves llibreries (la de sockets per al server i la de pymongo per a la connexió amb la BD).

Per acabar, s'ha aconseguit l'objectiu de proporcionar un recomanador musical a l'app d'Apolo, que en aquest cas s'ha aconseguit integrant en el visualitzador de recomanacions en la mateixa app, Així com la integració d'Spotify dins del sistema recomanador per tal de poder personalitzar més les recomanacions sobretot als nou vinguts a l' app.

Com a millores es podria seguir investigant per a trobar un algorisme que requerís de menys temps per a processar les dades.

AGRAÏMENTS

Agrair tant al meu tutor Josep Lladós com al meu tutor de pràctiques Òscar Cardona pel suport donat durant el treball. També agrair a les empreses Omitsis i Apolo per a deixar-me utilitzar les apps i l'estructura de la base de dades ja existents com a base del projecte.

BIBLIOGRAFIA

[1] Relatiooon: http://relatiooon.com/

[2] Collaborative Filter Recommender Systems http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.4520&rep=rep1&type=pdf

[3] Content Based Recommender Systems http://facweb.cs.depaul.edu/mobasher/classes/ect584/Papers/ContentBasedRS.pdf

[4] PyCharm IDE https://www.jetbrains.com/pycharm/

[5] MongoChef http://3t.io/mongochef/

[6] Android Studio IDE https://developer.android.com/studio/index.html

[7] Virtual Box https://www.virtualbox.org/

[8] Spotidy Android SDK
https://developer.spotify.com/technologies/spotify-android-sdk/android-sdk-authentication-guide/

[9] Spotify Web Api Wrapper https://github.com/kaaes/spotify-web-api-android

[10] Spotify web api https://developer.spotify.com/web-api/

[11] Android Dialog
https://developer.android.com/guide/topics/ui/dialogs.html

[12] Tots els generes musicals de l'api d'Spotify (944)
https://docs.google.com/spreadsheets/d/1L3F3oKddQxz2v9a_eqchacv4XXqVru1AMwsbVUqqMsU/pub