

# CHOOSE & EAT: Recomendador de restaurantes.

Trabajo Fin de Máster

Jorge Serrano del Cerro

## Índice

## 

Introducción 3

Datos utilizados 4

Metodología 8

Resultados 8

Manual de usuario 8

## Introducción

Un sistema de recomendación es un sistema inteligente que proporciona a los usuarios una serie de sugerencias personalizadas (recomendaciones) sobre un determinado tipo de elementos (items). Los sistemas de recomendación estudian las características de cada usuario y, mediante un procesamiento de los datos, encuentra un subconjunto de items que pueden resultar de interés para el usuario.

Podemos clasificar los sistemas de recomendación de la siguiente forma:

1. Sistema de recomendación con filtrado basado en contenido: las recomendaciones se basan en el conocimiento que se tiene sobre los items que el usuario ha valorado (ya sea de forma implícita o explícita), y se le recomendarán items similares que le puedan gustar o interesar. Un ejemplo de este tipo de sistema es Youtube.
2. Sistema de recomendación con filtrado demográfico: estas recomendaciones se realizan en función de las características de los usuarios (edad, sexo, situación geográfica, profesión, etc).
3. Sistema de recomendación basado en filtrado colaborativo: el ﬁltrado colaborativo consiste en ver qué usuarios son similares al usuario activo y, a continuación, recomendar aquellos items que no han sido votados por el usuario activo y que han resultado bien valorados por los usuarios similares.
4. Sistema de recomendación con métodos de filtrado híbrido: mezcla alguno de los tres filtrados mencionados anteriormente para realizar recomendaciones e incluso lo combinan con alguna otra técnica de inteligencia artificial.

En este trabajo vamos a realizar un sistema de recomendación basado en el filtrado colaborativo. En particular, vamos a usar la técnica de los K-vecinos que se basa en recomendar a un usuario los items que le han gustado a otros usuarios con gustos similares al usuario activo, de ahí el nombre de los k-vecinos. En esta técnica las recomendaciones se hacen siguiendo cuatro pasos:

* Calcular de la similitud entre usuarios: en primer lugar se ha de elegir una métrica para determinar la similaridad/distancia entre el usuario activo y el resto de usuarios.
* Calcular los K-Vecinos: haciendo uso de la métrica seleccionada, se obtienen los k usuarios más similares al usuario activo. A estos usuarios se les denomina como los

k-vecinos.

* Calcular las predicciones de los items: a partir de los k-vecinos del usuario activo, se determinan las posibles valoraciones que el usuario activo haría sobre los items que no ha votado.
* Realizar las recomendaciones: tras el cálculo de las predicciones, se eligen los N items más adecuados para ser recomendados al usuario. De forma opcional, puede incluirse un valor mínimo para evitar que los items con una predicción inferior a dicho umbral sean recomendados.

## Datos utilizados

Los datos los hemos obtenido de Kaggle (https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings#README).

El conjunto de los datos consta de 8 ficheros csv. Veamos el contenido de cada uno de ellos:

* chefmozaccepts.csv: contiene 615 restaurantes y 11 métodos de pago. La estructura de la tabla es:

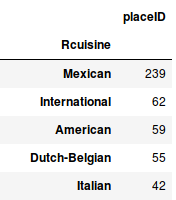
|  |
| --- |
| **chefmozaccepts** |
| placeID |
| Rpayment |

El campo Rpayment tiene los valores: CASH, VISA, MASTERCARD-EUROCARD, AMERICAN\_EXPRESS, BANK\_DEBIT\_CARDS, CHECKS, DISCOVER, CARTE\_BLANCHE, DINERS\_CLUB, JAPAN\_CREDIT\_BUREAU, GIFT\_CERTIFICATES.

* Chefmozcuisine. csv: contiene 769 restaurantes y 59 tipos de comida. Este fichero tiene la siguiente estructura:

|  |
| --- |
| **Chefmozcuisine** |
| placeID |
| Rcuisine |

Donde los valores más comunes del campo Rcuisine son:



* chefmozhours4.csv: contiene 694 restaurantes y sus respectivos horarios. Y la estructura es:

|  |
| --- |
| **chefmozhours4** |
| placeID |
| hours |
| days |

* chefmozparking.csv: contiene 675 restaurantes y 7 tipos de parking. La estructura es:

|  |
| --- |
| **chefmozparking** |
| placeID |
| parking\_lot |

Y los valores del campo parking\_lot son: public, none, yes, valet parking, fee, Street, validated parking.

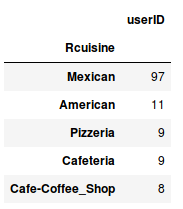
* geoplaces2.csv: contiene toda la información sobre el restaurante desde la localización hasta si venden alcohol o precio de la comida. Veamos la estructura:

|  |
| --- |
| **geoplaces2** |
| placeID |
| latitude |
| longitude |
| the\_geom\_meter |
| name |
| address |
| city |
| state |
| country |
| fax |
| zip |
| alcohol |
| smoking\_area |
| dress\_code |
| accessibility |
| price |
| url |
| Rambience |
| franchise |
| area |
| other\_services |

* usercuisine.csv: contiene la información de 138 usuarios y sus gustos culinarios (103 distintos). La estructura del fichero es:

|  |
| --- |
| **usercuisine** |
| userID |
| Rcuisine |

Los valores más usuales del campo Rcuisine son:



* userpayment.csv: el fichero contiene la preferencia de pago de 133 usuarios. La estructura del fichero es

|  |
| --- |
| **userpayment** |
| userID |
| Upayment |

Y el campo Upayment tiene los valores: cash, bank\_debit\_cards, MasterCard-Eurocard, VISA, American\_Express.

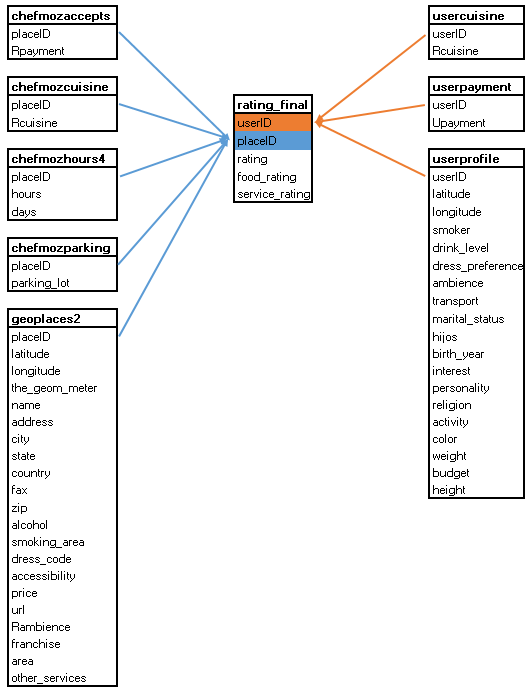
* userprofile.csv: contiene la información de los perfiles de 138 usuarios desde estatura y color favorito a fumador o tipo de transporte. La estructura de este fichero es:

|  |
| --- |
| **userprofile** |
| userID |
| latitude |
| longitude |
| smoker |
| drink\_level |
| dress\_preference |
| ambience |
| transport |
| marital\_status |
| hijos |
| birth\_year |
| interest |
| personality |
| religion |
| activity |
| color |
| weight |
| budget |
| height |

* rating\_final.csv: contiene la relación entre restaurantes y usuarios, con las calificaciones (general, servicio y comida). La estructura es la siguiente:

|  |
| --- |
| **rating\_final** |
| userID |
| placeID |
| rating |
| food\_rating |
| service\_rating |

Con todo ello tenemos la siguiente relación entre los ficheros:



## Metodología

Como hemos comentado en la introducción, en este trabajo hemos realizado un sistema recomendador basado en el filtrado colaborativo usando la técnica de K-vecinos con los datos introducidos en el punto anterior.

Las herramientas usadas han sido:

* Python: lo utilizamos como código base, tanto para tratar los ficheros que contienen los datos como para realizar el sistema recomendador propiamente dicho. Dentro de Python hemos usado las bibliotecas:
  + Numpy y Pandas: para trabajar con los datos.
  + Matplotlib.Pylab: para la visualización de datos.
  + Scipy.Spatial: para calcular las distancias entre usuarios.
* Flask: para crear la aplicación web.
* Html: con la que hemos construido la página donde vamos a volcar los datos.

## Resultados

Como resultado del sistema de recomendación hemos creado una aplicación Web con Flask para la visualización de los resultados. Para ello hemos desarrollado:

* Recommender.py: donde tenemos los métodos para realizar el proceso de recomendador.
* Init.py: se encuentra como puente entre el front-end y el back-end.
* Main.html: donde se muestran los datos para el usuario.

## Manual de usuario

Una vez que nos hemos descargado el repositorio nos situamos en la carpeta web y ejecutamos el fichero init.py (usamos el comando “Python init.py”). Si no se abre el navegador automáticamente, abrimos un navegador y en la barra de direcciones escribimos <http://127.0.0.1:5000>

Nos aparece la web donde elegimos en los menús despegables nuestras preferencias y, pulsando el botón de Submit, la aplicación nos devuelve los 5 restaurantes recomendados:

