Miércoles 15 Diciembre 2021

Funcionalidades Implementadas

versión entrega 3.0

PROP

Universitat Politècnica de Catalunya

2021-2022 Q1

SUBGRUPO PROP 2.1:
ANDRÉS EDUARDO BERCOWSKY RAMA
CARLA CAMPÀS GENÉ
BEATRIZ GOMES DA COSTA
PAU VALLESPÍ MONCLÚS

1 FUNCIONALIDADES OBLIGATORIAS

1.1 Definición de tipos a partir de sus atributos

1.2 Permitir la gestión de usuarios

En nuestro sistema, hemos creado dos clases para facilitar la gestión de los usuarios: *Usuario* y *Usuarios*. Gracias a estas, podemos saber qué ítems ha valorado este usuario, qué valoración les ha dado a cada uno y además, crear nuevos usuarios.

1.3 Permitir la gestión de ítems

Además de gestionar los usuarios, tenemos 2 clases adicionales para controlar la gestión de ítems: *Item* e *Items*. En ellas guardamos información fundamental como su identificador y los atributos que tiene cada ítem. Para guardar los atributos, utilizamos una clase llamada *Atributo* de la cual heredan 6 tipos de atributos, cada uno de ellos encapsulado en una clase: *AtributoNumerico, AtributoCategorico, AtributoCategoricoMultiple, AtributoBooleano, AtributoTexto* y *AtributoFecha.* Con todas estas clases, podemos gestionar de manera adecuada todos los ítems, y como el usuario guarda el identificador de los ítems que ha valorado, podemos obtener los ítems a partir de estos identificadores.

1.4 Permitir la valoración de ítems por parte del usuario

En nuestro sistema se le da la opción al usuario de añadir valoraciones nuevas a ítems que no ha valorado previamente o editar valoraciones que ha realizado a algún ítem. Además, el usuario también puede gestionar los ítems que ha valorado. Para ello, primero puede visualizar a qué ítems les ha dado valoraciones para después valorar nuevos ítems o borrar valoraciones que ha hecho anteriormente. Esto lo puede realizar tantas veces como desee y después, en el momento de recomendar nuevos ítems, el sistema tendrá en cuenta todas las modificaciones que ha hecho el usuario.

1.5 Definir funciones de distancia o similitud dependiendo de los atributos

Esta funcionalidad la llevamos a cabo en la parte de *Content Based Filtering* ya que es aquí donde calculamos las distancias entre ítems. Dependiendo del tipo del atributo del ítem, hemos definido 5 funciones para calcular la distancia entre ellos. Estas funciones devuelven valores entre 0 y 1 donde 0 es que la distancia entre ellos es muy baja y 1 muy alta. Conseguimos esto gracias a un preproceso de los datos donde normalizamos todos los datos numéricos y fechas.

Primero, para calcular la distancia entre dos atributos numéricos o de tipo fecha, utilizamos la misma función. Esta función simplemente devuelve la resta en valor absoluto de ambos atributos. Como los datos están normalizados, el resultado será un valor entre 0 y 1.

Después, para calcular las distancias entre dos atributos booleanos o categóricos, utilizamos una función que nos devuelve 0 si los elementos son diferentes o 1 si son iguales. Para esto tenemos 2 funciones (1 para cada tipo de atributo).

A continuación, para calcular las distancias entre dos atributos de tipo categórico múltiple, utilizamos un método algo más sofisticado que lo visto hasta ahora, y es parte de las funcionalidades opcionales que hemos realizado en este proyecto. Utilizamos una técnica llamada One-hot, en la que usamos la clase *BitSet* de Java para hacer más eficiente estos cálculos de distancia. Esta función consiste en tener para cada atributo categórico múltiple, un BitSet de tantas posiciones como posibles valores tenga ese atributo. Por poner un ejemplo, si los ítems son películas y el género es un atributo categórico múltiple, tendríamos para cada

película, un BitSet del tamaño de todos los géneros encontrados en el dataset. Pondremos a 1 todas las posiciones que correspondan a los géneros a los que pertenece esa película (en este caso). Todo esto se lleva a cabo en el preproceso. Entonces, la función calcula la similitud de cosenos de ámbos BitSets. de esta forma, la distancia entre dos atributos de tipo categórico múltiple es la similitud de coseno de sus BitSets. Como en la similitud de coseno un 1 representa que son iguales y un 0 que no tienen relación, ajustamos la fórmula devolviendo 1 menos lo que nos ha dado el cálculo anterior.

Finalmente, tenemos la función que calcula la distancia entre textos. Esta funcionalidad también es opcional. Para calcular la distancia entre dos atributos de tipo texto utilizamos una técnica llamada Tf-idf, que en inglés significa *Term frequency - Inverse document frequency*. Gracias a esto, los términos que aparecen muchas veces en muchos documentos no tienen tanto peso como términos que aparecen muchas veces en pocos documentos. Así, palabras muy comunes no tendrán peso en el cálculo de las distancias. Finalmente, una vez calculada el Tf-idf, realizamos una similitud de coseno entre ambos resultados para así obtener una distancia entre ambos y devolvemos 1 menos el resultado por la misma razón que hemos explicado anteriormente.

1.6 Guardar datos preprocesados

Otra funcionalidad que hemos implementado es el hecho de guardar los datos preprocesados en archivos de texto para poder seguir con el programa en otro momento si el usuario así lo desea. Lo que hacemos es guardar la información que ha pasado ya por el preproceso (como la normalización de los datos, la cabecera del dataset, los BitSets…) para después fácilmente, leyendo este archivo nos ahorremos volver a preprocesar todos los datos.

1.7 Recomendar un conjunto de ítems a un usuario

Esta es la funcionalidad clave de nuestro sistema. Se basa en que un usuario ha valorado una serie de ítems y después, le pide al sistema que le recomiende ítems que no haya valorado previamente. Para esto, nos basamos en dos técnicas: *Content Based Filterign* y *Collaborative Filtering.* En la primera parte, ordenamos los ítems en función de la distancia que tienen estos con los que ha valorado el usuario. Además de esto, también se tiene en cuenta la valoración que le ha dado el usuario a ese ítem. Por ello, estarán mejor posicionados los ítems similares a ítems que haya valorado el usuario positivamente y, por el contrario, tendrán peores resultados los ítems similares a ítems que el usuario haya valorado negativamente, o ítems muy diferentes a los que ha valorado positivamente el usuario. Para esta parte se usa el algoritmo de *K-nearest neighbors*.

Por otro lado, también usamos la técnica de *Collaborative filtering*, la cual se basa en recomendar ítems que usuarios similares han valorado positivamente. A diferencia del algoritmo anterior, ya no tenemos en cuenta las características de los ítems sino las del usuario. Por ello, buscaremos usuarios con gustos similares, que hayan valorado positivamente ítems similares (o negativamente) con el fin de recomendar otros ítems que le hayan gustado a esos usuarios (y que no haya valorado previamente). Para esta parte de las recomendaciones utilizaremos dos algoritmos: *K-means* y *Slope One.*

En el Slope One hemos optado por el sistema de weighted slope one, ya que con este haremos predicciones más aproximadas a lo que valoraría un ítem el usuario dado. Esto nos dará más constancia en los datos producidos ya que dará más peso a las valoraciones de usuarios con valoraciones más similares.

En el K-Means hemos optado por automatizar el método de coger la k correspondiente al algoritmo. Inicialmente esto estaba hecho con el elbow method, peró finalmente nos decidimos por el silhouette method ya que este proyecto requería más exactitud y el elbow method no nos daba esta. En el Silhouette Method, que usa nuestro proyecto, encontramos la k más óptima y nos permite automatizar este método.

Finalmente, combinamos ambos métodos de recomendación. El método híbrido junta ambas técnicas, donde hacemos la media de la predicción de la valoración que le ha dado cada uno de los métodos mencionados anteriormente. De esta manera, obtenemos una aproximación más precisa de la valoración que le podría llegar a dar el usuario basado en el contenido que ha valorado y en sus gustos comparados con otros usuarios.

1.8 Guardar Recomendaciones en Historial

En cuanto se generan las recomendaciones se guardará está en el historial cada vez que el usuario se cambie o se cierre la sesión. Dado esto el usuario tendrá la opción de ver las recomendaciones previas que se han hecho en el sistema. Podrá ver con qué algoritmo se ha creado y ver las diferentes valoraciones que se han predecido en el sistema. Además el usuario tendrá la opción de gestionar estas recomendaciones, borrando la recomendación que quiera del sistema y borrando diferentes ítems de las valoraciones correspondientes. Recuperando estas cuando sea necesario.

1.9 Evaluación de Recomendaciones

Finalmente, también hemos implementado un algoritmo para valorar la calidad de las recomendaciones. Este se llama DCG, que en inglés significa *Discounted Commulative Gain.* Este algoritmo mide la "utilidad" de un ítem basado en la posición en la que se encuentra después de haber ordenado los ítems por orden de valoración que se predice que le dará el usuario. Le damos la opción al usuario de evaluar las recomendaciones que se le dan utilizando este algoritmo.

1.10 Eliminación de atributos

Para obtener una mejor predicción de los ítems que hay que recomendarle a un usuario, hemos eliminado algunos atributos que no eran interesantes para el cálculo de distancia entre ítems (que como sabemos se basa en la suma de las distancias entre los atributos). Por ello, atributos como el path, url o similares los hemos eliminado y no los tendremos en cuenta, ya que no son características relevantes de esos ítems.

1.11 Asignación de tipos a los atributos

Otra funcionalidad que hemos añadido es el hecho de crear un fichero que contenga el tipo de cada uno de los atributos de los ítems. Esto facilita mucho la tarea del preproceso ya que conociendo esto, podemos clasificar cada atributo con su correspondiente tipo.