

Primera Entrega

Investigación algoritmos de recomendaciones

Un algoritmo de recomendación es una subclase de un sistema de filtración de información que busca predecir la preferencia que un usuario podría tener sobre un objeto.

Sistema de recomendación fuera de línea

- Familia y amigos que recomiendan
- Profesores recomendando algún libro

Es un hecho que estos recomendadores fuera de línea saben algo sobre la persona a la que recomiendan un objeto y por ende basan la decisión en estos datos con el fin del beneficio de la persona a la que recomiendan.

Sistema de recomendación en línea

- Facebook utiliza un sistema de recomendación para recomendar amigos que podrías conocer fuera de línea. El sistema utiliza información personal como la escuela, lugares de trabajo u otros pasatiempos del individuo.
- Netflix utiliza la información de los gustos de un usuario para recomendar programas de televisión e incluso incluye los géneros de las películas, el historial y las valoraciones dadas a películas.

(Dataconomy grupo editor, 2020)

Filtros colaborativos

Los sistemas de recomendación colaborativos se basan en recopilar y analizar una gran cantidad de información sobre los comportamientos, actividades o preferencias de los usuarios y predecir lo que les gustará a los usuarios en función de su similitud con otros usuarios. Una ventaja clave del enfoque de filtrado colaborativo es que no se basa en contenido analizable por máquina y, por lo tanto, es capaz de recomendar con precisión elementos complejos como películas sin requerir una "comprensión" del elemento en sí. Se han utilizado muchos algoritmos para medir la similitud del usuario o la similitud de elementos en los sistemas de recomendación. Por ejemplo, el enfoque de k vecino más cercano (k-NN) y la correlación de Pearson. (Dataconomy grupo editor, 2020)

Filtros basados en contenido

Los métodos de filtrado basados en contenido se basan en una descripción del elemento y un perfil de las preferencias del usuario. En un sistema de recomendación basado en contenido, se utilizan palabras clave para describir los elementos; al lado, se crea un perfil de usuario para indicar el tipo de elemento que le gusta a este usuario. En otras palabras, estos algoritmos

intentan recomendar elementos que son similares a los que le gustaban a un usuario en el pasado (o que está examinando en el presente). En particular, se comparan varios elementos candidatos con elementos previamente calificados por el usuario y se recomiendan los elementos que mejor coincidan. Este enfoque tiene sus raíces en la recuperación de información y la investigación de filtrado de información. (Dataconomy grupo editor, 2020)

Filtrado híbrido

Investigaciones recientes han demostrado que un enfoque híbrido, que combine el filtrado colaborativo y el filtrado basado en contenido, podría ser más eficaz en algunos casos. Los enfoques híbridos se pueden implementar de varias maneras, haciendo predicciones basadas en contenido y basadas en la colaboración por separado y luego combinándolas, agregando capacidades basadas en el contenido a un enfoque basado en la colaboración (y viceversa), o unificando los enfoques en uno. modelo. Varios estudios comparan empíricamente el rendimiento del híbrido con los métodos puramente colaborativos y basados en contenido y demuestran que los métodos híbridos pueden proporcionar recomendaciones más precisas que los enfoques puros. Estos métodos también se pueden utilizar para superar algunos de los problemas comunes en los sistemas de recomendación, como el arranque en frío y el problema de escasez. (Dataconomy grupo editor, 2020)

Algoritmos de grafos

Algoritmo Dijkstra

Algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959. (Grupo editor ecured, 2020)

Sus aplicaciones incluyen la distribución de productos a una red de establecimientos comerciales, distribución de correos postales entre otros. Entre sus características encontramos que es un algoritmo greedy, que trabaja por etapas y toma cada etapa la mejor solución sin considerar las consecuencias y por último tenemos que el óptimo encontrado en una etapa puede modificarse posteriormente si surge una mejor solución. (Grupo editor ecured, 2020)

El algoritmo de Dijkstra funciona visitando vértices en el gráfico comenzando con el punto de partida del objeto. Luego examina repetidamente el vértice más cercano aún no examinado, agregando sus vértices al conjunto de vértices que se van a examinar. Se expande hacia afuera desde el punto de partida hasta llegar a la meta. Se garantiza que el algoritmo de Dijkstra encontrará el camino más corto desde el punto de partida hasta la meta, siempre que ninguno de los bordes tenga un costo negativo. (Introduction to A*, 2020)

Greedy Best First Search

El algoritmo Greedy Best-First-Search funciona de manera similar al Dijkstra, excepto que tiene una estimación (llamada heurística) de qué tan lejos del objetivo está cualquier vértice. En lugar de seleccionar el vértice más cercano al punto de partida, selecciona el vértice más cercano a la meta. No se garantiza que Greedy Best-First-Search encuentre el camino más corto. Sin embargo, se ejecuta mucho más rápido que el algoritmo de Dijkstra porque utiliza la función heurística para guiar su camino hacia la meta muy rápidamente. Finalmente es necesario mencionar que a pesar de todo esto el algoritmo no produce rutas óptimas en general, ya que todo lo que utiliza es la heurística, por lo que no se utiliza ninguna información de los costes reales. (Introduction to A*, 2020)

A*

Es una de las mejores técnicas y más populares para búsqueda de rutas y recorrido de grafos. (Rachit, 2021)

El algoritmo A* tiene en cuenta tanto la distancia real desde la fuente a un nodo, y la distancia estimada desde el nodo hasta la meta. A diferencia del algoritmo de Dijkstra, que funciona para pesos positivos, incluido el cero, A* solo funciona con pesos estrictamente positivos. Fue descrito por primera vez en 1968 por Peter Hart, Nils Nilsson y Bertram Raphael. (Monzonis, 2019)

El secreto de su éxito es que combina la información que utiliza el algoritmo de Dijkstra (favoreciendo los vértices que están cerca del punto de partida) y la información que utiliza Greedy Best-First-Search (favoreciendo los vértices que están cerca del objetivo). En la terminología estándar utilizada cuando se habla de A*, $g(n)$ representa el costo exacto de la ruta desde el punto de partida hasta cualquier vértice n , y $h(n)$ representa el costo estimado heurístico desde el vértice n hasta la meta. En los diagramas anteriores, el amarillo (h) representa los vértices lejos de la meta y el verde azulado (g) representa los vértices lejos del punto de partida. A* equilibra los dos a medida que avanza desde el punto de partida hasta la meta. Cada vez que pasa por el bucle principal, examina el vértice n que tiene la menor $f(n) = g(n) + h(n)$. (Monzonis, 2019)

Algoritmo Floyd Warshall

El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. Este es uno de los problemas más interesantes que se pueden resolver con algoritmos de grafos.

Existen varias soluciones a este problema y los algoritmos a aplicar dependen también de la existencia de arcos con pesos o costes negativos en el grafo. En el caso de no existir pesos negativos, sería posible ejecutar V veces el algoritmo de Dijkstra para el cálculo del camino mínimo, donde V es el número de vértices o nodos del grafo. Esto conllevaría un tiempo de ejecución de $O(V^3)$ (aunque se puede reducir). Si existen arcos con pesos negativos, se puede ejecutar también V veces el Algoritmo de Bellman-Ford, una vez para cada nodo del

grafo. Para grafos densos (con muchas conexiones o arcos) esto conllevaría un tiempo de ejecución de $O(V^4)$.

El algoritmo de Floyd-Warshall ('All-Pairs-Shortest-Path' – Todos los caminos mínimos) ideado por Floyd en 1962 basándose en un teorema de Warshall también de 1962, usa la metodología de Programación Dinámica para resolver el problema. Éste puede resolver el problema con pesos negativos y tiempos de ejecución iguales a $O(V^3)$; sin embargo, para ciclos de peso negativo el algoritmo tiene problemas.

(grupo editor Estructura Site, 2020)

Algoritmo de Kruskal

Joseph B. Kruskal investigador del Math Center (Bell-Labs), que en 1956 descubrió su algoritmo para la resolución del problema del Árbol de coste total mínimo (minimum spanning tree – MST) también llamado árbol recubridor euclídeo mínimo. Este problema es un problema típico de optimización combinatoria, que fue considerado originalmente por Otakar Boruvka (1926) mientras estudiaba la necesidad de electrificación rural en el sur de Moravia en Checoslovaquia.

El objetivo del algoritmo de Kruskal es construir un árbol (subgrafo sin ciclos) formado por arcos sucesivamente seleccionados de mínimo peso a partir de un grafo con pesos en los arcos.

Un árbol (spanning tree) de un grafo es un subgrafo que contiene todos sus vértices o nodos. Un grafo puede tener múltiples árboles. Por ejemplo, un grafo completo de cuatro nodos (todos relacionados con todos) tendría 16 árboles.

La aplicación típica de este problema es el diseño de redes telefónicas. Una empresa con diferentes oficinas trata de trazar líneas de teléfono para conectarlas unas con otras. La compañía telefónica le ofrece esta interconexión, pero ofrece tarifas diferentes o costes por conectar cada par de oficinas. Cómo conectar entonces las oficinas al mínimo coste total.

La formulación del MST también ha sido aplicada para hallar soluciones en diversas áreas (diseño de redes de transporte, diseño de redes de telecomunicaciones – TV por cable, sistemas distribuidos, interpretación de datos climatológicos, visión artificial – análisis de imágenes – extracción de rasgos de parentesco, análisis de clusters y búsqueda de superestructuras de quasar, plegamiento de proteínas, reconocimiento de células cancerosas, y otros).

(grupo editor Estructura Site, 2020)

Algoritmo de Prim

“El algoritmo de Prim es tal vez el algoritmo de MST (Árboles Generadores Mínimos) más sencillo de implementar y el mejor método para grafos densos. Este algoritmo puede encontrar el MST de cualquier grafo conexo pesado.

Sea V el conjunto de nodos de un grafo pesado no dirigido. El algoritmo de Prim comienza cuando se asigna a un conjunto U de nodos un nodo inicial Perteneciente a V , en el cual “crece” un árbol de expansión, arista por arista. En cada paso se localiza la arista más corta (u, v) que conecta a U con $V-U$, y después se agrega v , el vértice en $V-U$, a U . Este paso se repite hasta que $V=U$. El algoritmo de Prim es de $O(N^2)$, donde $|V| = N$.

(grupo editor Estructura Site, 2020)

Desing Thinking

Empatía

Necesidades encontradas

- Lectura significativa
- Podcast interesantes e innovadores
- Emprendimientos innovadores y que cubran una necesidad
- Pasatiempos durante la cuarentena
- Animes y mangas
- Artistas emergentes en Guatemala

Resumen de entrevistas:

Se realizo una entrevista al estudiante de la UNIS Kai Escobar y nos comento que no estaba tan interesado en un sistema de recomendación de lectura ni de pasatiempos, ni de emprendimientos. Sin embargo, se mostró interesado en un sistema un sistema de podcast, artistas emergentes y pasatiempos. También recomendó un sistema de recomendación de free lancers. El entrevistado aun que no supo al principio que era un sistema de recomendación se le explico con ejemplos y por ello pudo responder las preguntas y a portar al proyecto con respuestas de calidad y concisas.

Definición

Como ya existen algoritmos de recomendación de podcast y de libros no resolvería ninguna necesidad antropológica de la sociedad. Por otro lado, se pensó que hacer uno de los emprendimientos, pasatiempos o artistas emergentes, pero surgió el problema de como generar una base de datos para ello y cuál iba a ser el criterio para la recomendación. Por último, se tomó la decisión entre free lancers, podcasts regionales y animes y el elegido fue una mezcla entre podcast, libros y animes.

Ideación

En este caso se decidió trabajar con la plataforma Neo4j que es una base de datos nativa donde se puede crear no solo data si no también relaciones entre esta misma. Neo4j conecta la data y se guarda permitiendo la existencia de queries y tablas a una velocidad considerable. Por otro lado, se decidió utilizar Python como el lenguaje para programar debido a su gran popularidad en el mundo de manejo de datos y fácil entendimiento. Es necesario mencionar que el tema principal de nuestro proyecto girara alrededor de entretenimiento donde no solo se recomendaran libros, podcast y anime para el tiempo libre si no que tengan un impacto en la educación de los individuos.

Prototipo

El prototipo de baja fidelidad se realizó en canva y se pensó como una página web que recomienda libros, podcast y anime. Sin embargo, como se programará en Python se deberá utilizar una librería GUI como Tkinter para que se vea lo mas fiel posible al prototipo. Evidentemente no se verá como una pagina web si no más como una aplicación Windows. El prototipo se encuentra en la carpeta del proyecto y se realizaron por tiempo solo dos iteraciones de este pensamos hacer una entrevista ya con el interfaz en Tkinter diseñado para ver si cumple con lo que nuestros usuarios desean.

Testing

Se realizo una entrevista del primer prototipo de baja fidelidad y a partir de este se hicieron modificaciones al original.

Resumen de Entrevista:

Al usuario le gustó la página principal, pero critico la pagina que describe el proyecto ya que las imágenes no eran coherentes con lo que se busca recomendar en el sistema. Luego mencionó que la letra de la pagina era muy pequeña en la parte de las preguntas para conocer al usuario. Finalmente le gustó la interfaz de las recomendaciones.

Aun que realizamos estos cambios nuestro prototipo no enseña la parte de iniciar sesión para tener datos importantes de las personas, tampoco muestra una forma muy linear de como vamos a mostrar la información contenida en la base de datos para explicar al usuario como funciona el algoritmo de recomendación.

Bibliografía

Dataconomy grupo editor. (2020). *AN INTRODUCTION TO RECOMMENDATION ENGINES*. Retrieved from Dataconomy: <https://dataconomy.com/2015/03/an-introduction-to-recommendation-engines/>

Grupo editor ecured. (2020). *Algoritmo de Dijkstra*. Retrieved from Ecured: https://www.ecured.cu/Algoritmo_de_Dijkstra

Algoritmos y Estructura de Datos

Juan Fernando Ramirez 20666

Elder Guzman 19628

grupo editor Estructura Site. (2020). *Estructura de datos II*. Retrieved from Estructura Site:
<https://estructurasite.wordpress.com/algortimo-de-floyd-warshall/>

*Introduction to A**. (2020). Retrieved from Stanford:

[http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html#:~:te
xt=A*%20is%20the%20most%20popular,a%20heuristic%20to%20guide%20itself](http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html#:~:text=A*%20is%20the%20most%20popular,a%20heuristic%20to%20guide%20itself).

Monzonis, D. (2019, enero 15). *PATHFINDING ALGORITHMS*. Retrieved from
Universitat de Barcelona:

<http://diposit.ub.edu/dspace/bitstream/2445/140466/1/memoria.pdf>

Rachit, B. (2021, febrero 1). *A* Search Algorithm*. Retrieved from Geeks for Geeks:
<https://www.geeksforgeeks.org/a-search-algorithm/>