TECNOLÓGICO NACIONAL DE MÉXICO

TECNM

Tecnológico Nacional de México

Campus Culiacán

Ingeniería en Sistemas computacionales

Inteligencia artificial

09:00 – 10:00

Practica 3 – Sistema de recomendacion

Integrantes:

Caro García Jorge Ariel

Galván González Sebastián

Docente:

ZURIEL DATHAN MORA FELIX

16/02/2025

Como construir un sistema de recomendaciones (¿Qué tecnologías y frameworks se usan?)

Primero quiero que te imagines navegando por plataformas como Amazon, Netflix o Spotify. Seguro que te aparecen muchas sugerencias sobre productos que te pueden interesar, películas o series que quizás te gusten o música de tu estilo. Pues estas recomendaciones no son aleatorias. Forman parte de los denominados sistemas de recomendación con data science, que muchas empresas implementan porque les reportan una gran cantidad de beneficios.

¿Qué son los sistemas de recomendación?

Los sistemas de recomendación son algoritmos que tratan de predecir qué productos o servicios de una tienda online tienen más posibilidades de ser adquiridos por un usuario para, seguidamente, mostrárselos en la web mientras este navega.

¿Cómo funcionan los sistemas de recomendación?

Los sistemas de recomendación se basan en análisis de datos a partir de la información que se ha recopilado de la navegación de los usuarios, como por ejemplo, qué productos han visto o comprado y cómo han interactuado con la plataforma. Para ello, se utilizan algoritmos avanzados capaces de hacer comparaciones detalladas entre distintos perfiles de usuarios y encontrar patrones comunes.

Cómo crear un sistema de recomendación con machine learning

Python es uno de los lenguajes más utilizados para crear herramientas de data science, de machine learning y también a la hora de crear páginas web, principalmente por su código robusto y su optimizada sintaxis. A los programadores que comienzan en este mundo se les recomienda utilizarlo, ya que es uno de los lenguajes más confiables a la hora de crear software.

Sin embargo, también existen otras alternativas como Java, Golang, Node.js, PHP o Ruby.

Java es la mejor alternativa a Python, de hecho, es su principal competidora por decirlo de alguna manera.

Construir un sistema de recomendaciones implica el uso de diversas tecnologías y frameworks que dependen del enfoque del sistema y de los datos disponibles.

1. Tipos de Sistemas de Recomendación

Antes de elegir la tecnología, es importante definir el tipo de recomendador que se quiere construir:

- **Basado en filtrado colaborativo**: Se basa en las preferencias de usuarios similares.
- **Basado en contenido**: Utiliza las características de los productos para hacer recomendaciones.
- **Híbrido**: Combina ambos enfoques para mejorar la precisión.

2. Tecnologías y Frameworks Comunes

Lenguajes de Programación

- **Python** (más popular debido a su ecosistema de machine learning).
- **Java** (utilizado en sistemas escalables y empresariales).
- **Scala** (útil para procesamiento distribuido con Spark).

Librerías y Frameworks de Machine Learning

- **Scikit-Learn**: Ideal para modelos sencillos de filtrado colaborativo.
- **TensorFlow / PyTorch**: Para modelos basados en deep learning.
- **Surprise**: Especializado en filtrado colaborativo y matrices de descomposición.

Big Data y Procesamiento Distribuido

- **Apache Spark MLlib**: Para recomendaciones a gran escala.
- **Hadoop**: Procesamiento distribuido si se tienen grandes volúmenes de datos.

Bases de Datos

- **SQL (PostgreSQL, MySQL)**: Para almacenar información estructurada de usuarios y productos.
- **NoSQL (MongoDB, Cassandra)**: Ideal para manejar grandes volúmenes de datos no estructurados.
- **Neo4j**: Para sistemas basados en grafos.

Herramientas para APIs y Desarrollo Web

- **Flask / FastAPI** (Python): Para construir APIs de recomendación.
- **Django**: Framework completo para aplicaciones web con recomendadores.
- **Node.js**: Para sistemas escalables en tiempo real.

Enfoques y Algoritmos

- **Modelos basados en matrices (SVD, ALS)**: Utilizados en filtrado colaborativo.
- **Modelos de deep learning (Autoencoders, Transformers)**: Para recomendaciones más precisas.
- **Modelos de grafos (Graph Neural Networks)**: Usados en plataformas como Pinterest y LinkedIn.

Implementación en Producción

- **Docker y Kubernetes**: Para escalar servicios de recomendación.
- **Google Cloud AI / AWS SageMaker / Azure ML**: Para entrenar modelos en la nube.
- **Redis**: Para almacenar caché y acelerar recomendaciones en tiempo real.

¿Qué herramientas usa amazon?

El sistema de recomendación de Amazon utiliza el filtrado colaborativo "item to item". Es decir, funciona analizando un producto comprado y/o añadido al carrito, incluso metido en la lista de deseos, mostrándole así al cliente un ranking con productos que puedan ser de su interés. Asimismo, en el algoritmo también intervienen la puntuación dada a los productos por otros consumidores o las compras que han hecho usuarios similares.

Este aprendizaje busca tener como resultado recomendaciones individualizadas que mejoren la experiencia y el consumo de los usuarios en tiempo real. Es por ello, que Amazon cuenta con bases de datos enormes y su departamento de Big Data tiene gran importancia para el correcto funcionamiento del negocio y la satisfacción de los clientes.

Los puntos de obtención de datos de Amazon son múltiples: historial de compra, productos que se consultan, listas de deseos, localización geográfica y origen del tráfico, así como relaciones funcionales entre productos, cestas de compra frecuentes en otros clientes, etc.

¿Qué algoritmos existen y frameworks de desarrollo en la optimización de recursos?

¿Qué es un framework y para qué sirve?

Un framework, o un marco de trabajo, es una plantilla para desarrollar softwares de manera más rápida y eficiente. Solo necesitas escribir el código que se encargue de la lógica de tu app. No tendrás que reinventar la rueda y gastar tiempo en tareas básicas de desarrollo, como crear clases, manejar objetos y definir funciones típicas. Un framework define la estructura de tu futuro proyecto y proporciona las herramientas necesarias que puedes usar como bloques de construcción. Puede incluir:

- Bibliotecas y módulos para tareas específicas. Se encargan de funciones como manipulación de datos, gestión de sesiones, seguridad, autenticación, etc.
- Patrones de diseño recomendados para estructurar el código. Sirven para ejecutar operaciones típicas de desarrollo y crear una arquitectura coherente en la aplicación.
- Herramientas de desarrollo. Permiten generar y depurar el código, realizar pruebas unitarias, manejar bases de datos, etc.
- Estándares de codificación. Aseguran que los archivos y directorios se organicen de la misma manera, mientras que el código se mantenga consistente y legible. Además, establecen el formato unificado para los nombres de las variables.

Los tipos de frameworks varían según el propósito, el lenguaje de programación y el área de aplicación. Los frameworks más populares se usan para crear:

- Apps móviles
    - React Native (JavaScript), permite desarrollar aplicaciones móviles nativas para iOS y Android con una base de código común.
    - Flutter (Dart), es un framework de código abierto elaborado por Google. Ayuda a crear interfaces de usuario en aplicaciones para iOS y Android.
- Aplicaciones de escritorio
    - Electron (JavaScript), usa tecnologías web como HTML, CSS y JavaScript para construir aplicaciones nativas para Windows, macOS y Linux.

    - Qt (C++), permite crear aplicaciones multiplataforma para escritorio y dispositivos móviles. Además, ofrece herramientas y bibliotecas para diseñar interfaces gráficas de usuario.

- Videojuegos
    - Unity (C#), permite desarrollar juegos 2D y 3D para una variedad de plataformas, incluyendo PCs, consolas y dispositivos móviles.
    - Unreal Engine (C++), es un framework avanzado para elaborar juegos realistas de alta calidad.
- Aplicaciones empresariales
    - Spring (Java), ofrece módulos de funcionalidad diversa, como gestión de bases de datos, acceso remoto a servidores, autorización y autentificación.
    - ASP.NET (C#), está elaborado por Microsoft para construir aplicaciones empresariales en la plataforma .NET.
- Aplicaciones web
    - Django (Python), es un framework de alto nivel que sigue el patrón de diseño Modelo-Vista-Controlador (MVC).
    - Ruby on Rails (Ruby), promueve el principio de convención sobre la configuración. Es cuando el desarrollo se basa en las convenciones de programación establecidas en vez de las configuraciones definidas por el programador.
    - Laravel (PHP), cuenta con una sintaxis simple y una gran cantidad de herramientas para el desarrollo de aplicaciones web modernas.
    - Angular (JavaScript), está desarrollado por Google para crear aplicaciones web de una sola página (SPA).
    - Vue.js (JavaScript), se orienta a la construcción de interfaces interactivas y reutilizables.

Descubre los 10 tipos de algoritmos más usados. Explora cómo estos algoritmos han transformado la forma en que procesamos información y tomamos decisiones.

En la era digital actual, la omnipresencia de la tecnología y la explosión de datos han dado lugar a una creciente dependencia de los algoritmos en diversos campos. Estos intrincados conjuntos de instrucciones y reglas definen el núcleo de la inteligencia

artificial y el machine learning, impulsando avances significativos en la resolución de problemas complejos y la toma de decisiones automatizada. En este vasto panorama algorítmico, se despliega una fascinante variedad de enfoques, cada uno diseñado para abordar distintos tipos de problemas.

Este artículo se sumergirá en el fascinante mundo de los algoritmos, explorando sus diferentes categorías y destacando aquellos que han emergido como los más prominentes y utilizados en la actualidad. Desde algoritmos clásicos que han resistido la prueba del tiempo hasta las innovaciones más recientes que aprovechan la potencia computacional moderna, examinaremos cómo estos algoritmos han transformado radicalmente la manera en que procesamos información y tomamos decisiones en un mundo cada vez más digitalizado. Acompáñanos en este viaje exploratorio mientras desentrañamos los misterios de los algoritmos, desde sus fundamentos hasta sus aplicaciones más vanguardistas.

Los 10 tipos de algoritmos más usados

Lo cierto es que existen muchísimos tipos de algoritmos y es imposible familiarizarse con todos ellos. No obstante, a continuación presentamos algunos de los más utilizados.

1. Algoritmos de ordenación

Un algoritmo de ordenación es un conjunto de instrucciones diseñadas para organizar elementos en un conjunto de datos en un orden específico. La ordenación es fundamental en la manipulación eficiente de datos y se aplica en diversas áreas, como bases de datos, algoritmos de búsqueda y procesamiento de información. Los dos algoritmos de ordenación más comunes son:

- Quicksort: Un algoritmo de ordenación eficiente que utiliza la estrategia de "dividir y conquistar". Divide el conjunto de datos en subconjuntos más pequeños, ordena cada subconjunto y luego combina los resultados.
- Mergesort: Un algoritmo de ordenación que también emplea la estrategia "dividir y conquistar". Divide el conjunto en mitades, ordena cada mitad y luego combina las mitades ordenadas para obtener el conjunto ordenado final.

2. Algoritmos de búsqueda

Un algoritmo de búsqueda es un conjunto de instrucciones diseñadas para encontrar la ubicación o determinar la existencia de un elemento específico dentro de un conjunto de datos. Existen varias estrategias de búsqueda, y la elección del algoritmo depende de factores como la estructura de los datos y la eficiencia requerida. No obstante, los dos algoritmos de búsqueda más empleados son:

- Búsqueda Lineal: Examina cada elemento en el conjunto de datos en secuencia hasta encontrar el elemento deseado o llegar al final. Es simple pero puede ser ineficiente en conjuntos de datos grandes.
- Búsqueda Binaria: Aplicable solo en conjuntos de datos ordenados. Divide repetidamente el conjunto a la mitad y compara el elemento deseado con el elemento en el centro, reduciendo así el espacio de búsqueda.

Tanto los algoritmos de ordenación como los de búsqueda son fundamentales y ampliamente usados en la programación y resolución eficiente de problemas relacionados con la organización y búsqueda de datos. La elección del algoritmo adecuado depende de diversos factores, como el tamaño del conjunto de datos, la complejidad del problema y los requisitos de eficiencia.

3. Algoritmo de grafos

Un algoritmo de grafos se basa en reglas diseñadas para resolver problemas relacionados con estructuras de datos llamadas grafos. Los grafos son representaciones abstractas de relaciones entre objetos y constan de nodos (o vértices) y aristas (o arcos) que conectan estos nodos.

Los algoritmos de grafos se utilizan para analizar y manipular estas estructuras con el objetivo de resolver problemas prácticos en diversos campos. De nuevo, dentro del ábanico de algoritmos de grafos existen muchos tipos de algoritmos, de entre los cuales destacan:

- Búsqueda en Anchura (BFS): Un algoritmo que explora todos los nodos a la misma profundidad antes de avanzar a los nodos de la siguiente profundidad. Es útil para encontrar el camino más corto en grafos no ponderados.
- Algoritmo de Dijkstra: Se utiliza para encontrar el camino más corto entre un nodo de origen y todos los demás nodos en un grafo ponderado con pesos no negativos.
- Algoritmo de Bellman-Ford: Similar a Dijkstra pero puede manejar grafos con aristas de peso negativo. Se utiliza para encontrar el camino más corto en grafos con costos variables.
- Algoritmo de Kruskal: Utilizado para encontrar un árbol de expansión mínima en un grafo ponderado. Este árbol abarca todos los nodos del grafo y tiene la suma mínima de los pesos de las aristas.

4. Algoritmos de clasificación

Un algoritmo de clasificación es un conjunto de instrucciones o reglas diseñadas para asignar un objeto o instancia a una categoría o clase específica, basándose en sus características o atributos. Estos algoritmos son ampliamente utilizados en machine learning y minería de datos para organizar y etiquetar datos de manera automática, permitiendo la identificación de patrones y la toma de decisiones.

El proceso de clasificación implica entrenar un modelo utilizando un conjunto de datos de entrenamiento, donde se proporcionan instancias junto con sus respectivas etiquetas de clase. El modelo utiliza esta información para aprender patrones y relaciones entre las características y las clases. Una vez entrenado, el modelo se puede utilizar para clasificar nuevas instancias cuyas etiquetas de clase son desconocidas.

Algunos algoritmos de clasificación comunes incluyen:

- Regresión Logística: A pesar de su nombre, se utiliza para problemas de clasificación binaria, asignando instancias a una de dos clases posibles.
- Máquinas de Soporte Vectorial (SVM): Busca un hiperplano que maximice el margen entre clases en un espacio de características.
- Árboles de Decisión (explorados en mayor profundidad más adelante): Representan decisiones en forma de árbol, dividiendo iterativamente el conjunto de datos en subconjuntos.
- K-Vecinos Más Cercanos (KNN): Clasifica una instancia según la mayoría de las clases de sus k vecinos más cercanos en el espacio de características.
- Naive Bayes: Basado en el teorema de Bayes, asume independencia entre características y asigna probabilidades a las clases.
- Redes Neuronales: Modelos inspirados en el funcionamiento del cerebro, utilizados para problemas de clasificación complejos.
- Random Forest: Ensamble de árboles de decisión que votan para determinar la clase de una instancia.
- Gradient Boosting: Combina múltiples modelos más débiles para mejorar la precisión general del modelo.

La elección del algoritmo de clasificación depende de la naturaleza del problema, el tamaño y la calidad de los datos, así como de los requisitos específicos del contexto de aplicación. Cada algoritmo tiene sus propias características y rendimientos en diferentes situaciones, por lo que la selección adecuada es esencial para lograr resultados óptimos.

- No te pierdas: Algoritmos de clasificación vs. clusterización: Una explicación práctica

5. Algoritmos de regresión

Los algoritmos de regresión son un conjunto de técnicas en machine learning y estadísticas utilizadas para modelar y analizar la relación entre variables. Estos algoritmos se aplican principalmente en problemas de predicción para análisis predictivos, donde el objetivo es prever el valor de una variable de salida (o variable dependiente) basándose en una o más variables de entrada (o variables independientes).

La tarea principal de un modelo de regresión es encontrar una función matemática que describa la relación entre las variables de entrada y la variable de salida. La función resultante se puede utilizar para hacer predicciones sobre valores futuros o desconocidos de la variable de salida.

Algunos algoritmos de regresión comunes incluyen:

- Regresión Lineal Simple y Múltiple: La regresión lineal simple modela la relación entre dos variables, mientras que la regresión lineal múltiple modela la relación entre más de dos variables, considerando múltiples predictores.
- Regresión Polinómica: Utiliza una ecuación polinómica para modelar relaciones no lineales.
- Regresión de Vecinos Más Cercanos (KNN): El algoritmo predice el valor de salida basándose en los valores de salida de los k vecinos más cercanos en el espacio de características.
- Máquinas de Soporte Vectorial (SVM) para Regresión: El algoritmo encuentra un hiperplano en el espacio de características que mejor se ajusta a los datos.
- Árboles de Regresión: El algoritmo utiliza árboles de decisión para modelar relaciones no lineales y segmentar el espacio de características en regiones.
- Regresión Ridge y Lasso: Técnicas de regularización que ayudan a controlar el sobreajuste del modelo.
- Redes Neuronales para Regresión: El algoritmo utiliza arquitecturas de redes neuronales para modelar relaciones complejas.

6. Algoritmo de árbol de decisión

Un algoritmo de árbol de decisión es un método de aprendizaje supervisado utilizado en el ámbito del machine learning y la inteligencia artificial. Está diseñado para tomar

decisiones basadas en múltiples condiciones o características. Los árboles de decisión modelan decisiones en forma de un árbol, donde cada nodo interno representa una prueba en una característica, cada rama representa el resultado de esa prueba, y cada hoja representa la decisión final.

Características clave de los árboles de decisión:

- División Recursiva: El árbol se construye de manera recursiva, dividiendo el conjunto de datos en subconjuntos más pequeños en función de las características relevantes.
- Criterios de Decisión: En cada nodo, se realiza una prueba sobre una característica específica, y la decisión de qué característica y cómo dividir se realiza mediante un criterio que maximiza la pureza de las hojas resultantes.
- Clasificación o Regresión: Los árboles de decisión pueden utilizarse tanto para problemas de clasificación como para problemas de regresión. En clasificación, las hojas representan clases o categorías, mientras que en regresión, las hojas contienen valores numéricos.
- Interpretabilidad: Uno de los beneficios clave de los árboles de decisión es su capacidad para ser interpretados fácilmente por humanos. El árbol proporciona una estructura lógica que puede entenderse visualmente.

7. Algoritmo 'Greedy':

Un algoritmo greedy, o algoritmo voraz, es un enfoque de diseño de algoritmos que toma decisiones locales en cada etapa con la esperanza de llegar a una solución global óptima. En otras palabras, en cada paso, el algoritmo selecciona la opción que parece ser la mejor en ese momento, sin considerar las consecuencias a largo plazo. La idea es tomar decisiones que parecen ser las más beneficiosas en el momento actual, sin preocuparse por su impacto futuro.

Características clave de los algoritmos greedy:

- Elección Greedy: En cada paso, el algoritmo toma la decisión que parece ser la mejor en ese momento.
- Sin Retroceso: Una vez tomada una decisión, no se revisa ni se cambia en etapas posteriores.
- Esperanza de Optimización Global: A pesar de tomar decisiones locales, se espera que la secuencia de decisiones conduzca a una solución global óptima o, al menos, a una solución aceptable.

Los algoritmos greedy son efectivos en situaciones donde la elección local óptima también conduce a una solución global óptima o cercana a la óptima. Sin embargo, no siempre garantizan la mejor solución en todos los casos, por lo que su aplicabilidad depende del problema específico que se esté abordando.

8. Algoritmos de fuerza bruta:

Un algoritmo de fuerza bruta es un algoritmo cuya esencia se basa en probar todas las posibles soluciones y verificar cuál es la correcta. Este método exhaustivo es simple y garantiza encontrar la solución óptima a un problema, pero a menudo es ineficiente debido a la gran cantidad de combinaciones a considerar, especialmente cuando el tamaño del problema aumenta.

Características clave de los algoritmos de fuerza bruta:

- Exploración Exhaustiva: Este tipo de algoritmos consideran todas las combinaciones posibles para encontrar la solución.
- Sin Estrategia de Optimización: No se basan en estrategias inteligentes para reducir el espacio de búsqueda.
- Garantía de Solución Óptima: Debido a que examinan todas las posibilidades, los algoritmos de fuerza bruta garantizan encontrar la solución óptima si existe.

Aunque los algoritmos de fuerza bruta pueden ser eficaces para problemas pequeños o cuando no hay una solución más eficiente conocida, su tiempo de ejecución puede volverse impracticable (complejidad temporal) a medida que aumenta el tamaño del problema. En muchos casos, se buscan algoritmos más especializados y eficientes para abordar problemas específicos.

9. Algoritmo de 'backtracking'

Un algoritmo de 'backtracking' trata de buscar todas las soluciones posibles para un problema mediante la exploración sistemática de todas las opciones disponibles. Este enfoque se basa en el principio de prueba y error, donde el algoritmo avanza hacia adelante para probar una solución, pero retrocede (backtracks) cuando se da cuenta de que la solución parcial no puede llevar a una solución válida.

Características clave de los algoritmos de backtracking:

- Exploración Sistemática: El algoritmo prueba sistemáticamente todas las posibles soluciones, construyendo y deshaciendo soluciones parciales según sea necesario.

- Decisiones Secuenciales: Toma decisiones secuenciales para construir una solución, retrocediendo cuando se encuentra en un estado no válido.
- Sin Garantía de Solución Óptima: Aunque puede encontrar una solución, no garantiza la solución óptima.

El éxito de un algoritmo de backtracking depende de la eficiencia en la toma de decisiones para evitar explorar innecesariamente ciertos caminos. En algunos casos, se utilizan técnicas adicionales, como la poda (pruning), para reducir la búsqueda de soluciones. La implementación eficiente y la elección adecuada de estrategias son esenciales para abordar problemas de manera efectiva con algoritmos de backtracking.

10. Algoritmos de programación dinámica

Un algoritmo de programación dinámica es un tipo de algoritmo diseñado para dividir un problema en subproblemas más pequeños y resolver, en primer lugar, cada uno de los subproblemas. Las soluciones halladas se almacenan para evitar la repetición del trabajo. Esta técnica es especialmente útil cuando un problema puede descomponerse en subproblemas superpuestos o subproblemas que comparten soluciones comunes.

Características clave de la programación dinámica:

- Subestructura Óptima: El problema global puede resolverse mediante la combinación de soluciones óptimas de subproblemas más pequeños.
- Solapamiento de Subproblemas: Los subproblemas comparten soluciones comunes, y la programación dinámica evita recalcular las soluciones para cada subproblema, almacenando y reutilizando los resultados ya calculados.

Pasos fundamentales en un algoritmo de programación dinámica:

- Definición del Problema: Formular el problema en términos de subproblemas más pequeños.
- Identificación de Subproblemas: Identificar subproblemas que se solucionarán y combinarán para resolver el problema general.
- Definición de la Ecuación Recursiva: Desarrollar una ecuación recursiva que relacione la solución del problema original con las soluciones de sus subproblemas.

- Orden de Resolución: Determinar el orden en que se resuelven los subproblemas para asegurar que las soluciones necesarias estén disponibles cuando se requieran.

Almacenamiento de Resultados: Guardar los resultados de los subproblemas resueltos en una estructura de datos (a menudo una tabla o matriz) para su reutilización.

Bibliografía

- https://www.ibm.com/mx-es/think/topics/recommendation-engine
- https://www.cyberclick.es/numerical-blog/data-science-como-crear-un-sistema-de-recomendacion-con-machine-learning
- https://blog.smartupdigital.com/sistemas-de-recomendacion-ecommerce#:~:text=El%20sistema%20de%20recomendaci%C3%B3n%20de%20Amazon%20utiliza%20el%20filtrado%20colaborativo,puedan%20ser%20de%20su%20inter%C3%A9s.
- https://ebac.mx/blog/frameworks
- https://blog.bismart.com/10-algoritmos-mas-usados