Apuntes

Un ejemplo de proyecto real donde se usaría Hadoop sería el almacenamiento y análisis de grandes volúmenes de datos de registros de transacciones de una compañía de tarjetas de crédito. En este caso, se utilizaría Hadoop para almacenar los datos en un sistema distribuido de almacenamiento HDFS y para procesar los datos utilizando MapReduce para generar informes y estadísticas sobre las transacciones.

Por otro lado, un ejemplo de proyecto real donde se usaría Spark sería el procesamiento en tiempo real de datos de sensores para el mantenimiento predictivo en una fábrica. En este caso, se utilizaría Spark para recolectar y analizar datos de los sensores en tiempo real, utilizando operaciones iterativas y análisis estadísticos para detectar patrones y fallos en los equipos, y generar alertas para intervenciones preventivas.

—

Un ejemplo de un proyecto donde se usaría Hadoop sería el almacenamiento y análisis de grandes volúmenes de datos de registros de transacciones de una empresa de comercio electrónico. Los datos se almacenarían en un clúster Hadoop y se utilizarían herramientas como MapReduce para procesar y analizar los datos para obtener información valiosa sobre patrones de compra y tendencias de los clientes.

Por otro lado, un ejemplo de un proyecto donde se usaría Spark sería en un sistema de detección de fraude en tiempo real en una entidad bancaria. Se usaría Spark Streaming para procesar los datos de transacciones en tiempo real y detectar patrones sospechosos de actividad de tarjetas de crédito. Luego, se utilizarían las librerías de aprendizaje automático de Spark para entrenar modelos predictivos para detectar transacciones fraudulentas.

LSH

Locality Sensitive Hashing (LSH) es una técnica de aprendizaje automático que se utiliza para agrupar elementos similares en un conjunto de datos. En el contexto de la recuperación de información, hay varias aplicaciones posibles de LSH:

1. Búsqueda de imágenes similares: Se puede utilizar LSH para encontrar imágenes similares en un gran conjunto de imágenes almacenadas. Esto se hace mediante la creación de una "firma" para cada imagen, basada en características como el contenido, el color y la textura, y luego agrupando las imágenes con firmas similares.

2. Recuperación de documentos similares: Se puede utilizar LSH para encontrar documentos similares en una colección de documentos almacenados, basándose en características como el contenido del texto, las palabras clave y las frases similares. Esto es útil para mejorar la precisión de los resultados de búsqueda y para agrupar documentos relacionados.

- En Locality Sensitive Hashing (LSH), el uso de varias funciones hash y la técnica de apilamiento (stacking) se utiliza para evitar colisiones entre elementos dissimilares.

  La idea básica es que si dos elementos son similares en el espacio de características, entonces tienen una alta probabilidad de ser asignados al mismo bucket en al menos una de las funciones hash. Al usar varias funciones hash y apilarlas, se aumenta la probabilidad de que los elementos similares caigan en el mismo bucket, reduciendo así el número de colisiones entre elementos dissimilares.

  La técnica de apilado de varias funciones hash se utiliza para aumentar la precisión del algoritmo LSH, ya que reduce el número de colisiones entre elementos dissimilares y aumenta la probabilidad de que los elementos similares caigan en el mismo bucket.

- En Locality Sensitive Hashing (LSH), la técnica de repetición no se utiliza para mejorar la eficiencia y el rendimiento.

  La técnica de repetición se refiere a la práctica de aplicar varias veces el algoritmo LSH sobre el mismo conjunto de datos para mejorar la precisión de los resultados. Es posible que se utilice esta técnica para mejorar la precisión de los resultados obtenidos, pero no tiene relación con el rendimiento y eficiencia del algoritmo LSH.

  La eficiencia y el rendimiento del algoritmo LSH se pueden mejorar mediante técnicas como el uso de varias funciones hash y la técnica de apilamiento, así como el uso de estructuras de datos eficientes para almacenar y buscar los elementos en los buckets.

CB & CF

- En general, los sistemas de recomendación basados en contenido requieren un proceso de personalización más complejo en comparación con los sistemas de recomendación basados en filtrado colaborativo. Esto se debe a que los sistemas basados en contenido requieren una mayor cantidad de información sobre los elementos que se están recomendando, incluyendo la información semántica, visual y auditiva, así como la información sobre el contexto. Esto puede requerir un mayor tiempo para recolectar, procesar y analizar esta información.

  Sin embargo, en algunos casos, los sistemas de recomendación basados en contenido pueden ser más fáciles de personalizar que los sistemas de recomendación basados en filtrado colaborativo en nuevos escenarios de aplicación. Por ejemplo, si los datos están disponibles y estructurados, la personalización puede ser más rápida. En otros casos, si el sistema de recomendación basado en contenido es simple y no requiere de una gran cantidad de datos, también puede tardar menos tiempo en ser personalizado.

  En resumen, la afirmación es parcialmente verdadera, ya que depende del contexto y de la complejidad del sistema de recomendación.

- Es cierto que los métodos de factorización de matrices basados en SVD se pueden utilizar para resolver el problema de "cold start" en los sistemas de recomendación de filtrado colaborativo. La factorización de matrices se utiliza a menudo en sistemas de recomendación para descomponer una matriz de calificaciones en dos matrices más pequeñas, una que representa a los usuarios y otra que representa a los artículos. La SVD es una técnica específica de factorización de matrices que se utiliza para analizar la estructura subyacente de una matriz. Al utilizar SVD para descomponer una matriz de calificaciones, se pueden obtener características latentes de los usuarios y los artículos, que luego se pueden utilizar para hacer recomendaciones a usuarios nuevos o a usuarios que no han proporcionado suficientes calificaciones.

Betweenness y clustering local

El coeficiente de agrupamiento local (local clustering coefficient) mide la tendencia de los vecinos de un nodo a conectarse entre sí. Se calcula dividiendo el número de arcos entre los vecinos de un nodo entre el número total de posibles arcos entre esos vecinos.

La betweenness de un nodo mide cuán a menudo pasa por ese nodo el camino más corto entre otros dos nodos. Es una medida de centralidad que indica la importancia de un nodo en relación con el flujo de información en el grafo.

Para calcular la betweenness de cada nodo en proporción, primero se calcula la betweenness de cada nodo de manera absoluta, tal como lo hice en mi respuesta anterior. Luego se divide el valor de betweenness de cada nodo entre el número total de parejas de vértices en el grafo (n(n-1)/2 para grafos no dirigidos) para obtener la proporción.

Exercici 1 (2.5 puntos). Considera el siguiente grafo no dirigido con la siguiente matriz de adyacencia:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Calcula el coeficiente de clustering local y la "betweenness" de cada nodo.

- El coeficiente de agrupamiento local del nodo 0 es 0, ya que no tiene arcos entre sus vecinos.
- El coeficiente de agrupamiento local del nodo 1 es 1, ya que tiene 3 arcos entre sus vecinos de 4 posibles (1-2, 1-4, 4-1)
- El coeficiente de agrupamiento local del nodo 2 es 0, ya que no tiene arcos entre sus vecinos.
- El coeficiente de agrupamiento local del nodo 3 es 0, ya que no tiene arcos entre sus vecinos.
- El coeficiente de agrupamiento local del nodo 4 es 1, ya que tiene 2 arcos entre sus vecinos de 3 posibles (1-4, 4-1)
-
- La betweenness del nodo 0 es 1, ya que aparece en un camino corto (0-1-4)

- La betweenness del nodo 1 es 4, ya que aparece en cuatro caminos cortos (1-0-4, 1-2-3, 1-4-0, 1-4-1)
- La betweenness del nodo 2 es 0, ya que no aparece en ningún camino corto
- La betweenness del nodo 3 es 1, ya que aparece en un camino corto (3-1-4)
- La betweenness del nodo 4 es 2, ya que aparece en dos caminos cortos (4-0-1, 4-1-0)