Similitud entre nodos

FES Acatlán - Datos Masivos II

Guzmán Domínguez Andrés Manuel

Sánchez Sánchez Kevin

Similitud Estructural a Gran Escala

La **similitud estructural** mide cuánto se parecen dos nodos según su posición y conexiones dentro del grafo, sin usar atributos adicionales.

En grandes grafos (como redes sociales globales, redes web o de transporte), las métricas tradicionales **no son computacionalmente viables**.
Comparar todos los pares de nodos tiene complejidad $O(n2)$, por lo que se reemplazan por **métodos aproximados o paralelizados**.

Medidas

Comunes

- Common neighbors (CN)
- Jaccard Coefficient
- Adamic–Adar Index
- Preferential Attachment

Datos Masivos

- MinHash + LSH (Locality Sensitive Hashing)
- Sketching y Sampling estructural
- Similitud de caminos o proximidad difusiva
- Procesamiento distribuido de grafos

MinHash + LSH (Locality Sensitive Hashing)

- En lugar de calcular Jaccard exacto, se generan *hashes* que preservan similitud.
- Permite identificar pares de nodos similares sin comparar todos contra todos.
- Escalable con *MapReduce* o *Spark*.

Sketching y Sampling estructural

- Se crean representaciones compactas (“sketches”) de los vecinos de cada nodo.
- Se usan muestras aleatorias para estimar similitud.

- **Similitud de caminos o proximidad difusiva**
- Métodos como *Personalized PageRank* o *SimRank* se adaptan mediante:
  - *Monte Carlo random walks*,
  - *Distributed PageRank computation*, o
  - *Approximate neighborhood propagation*.

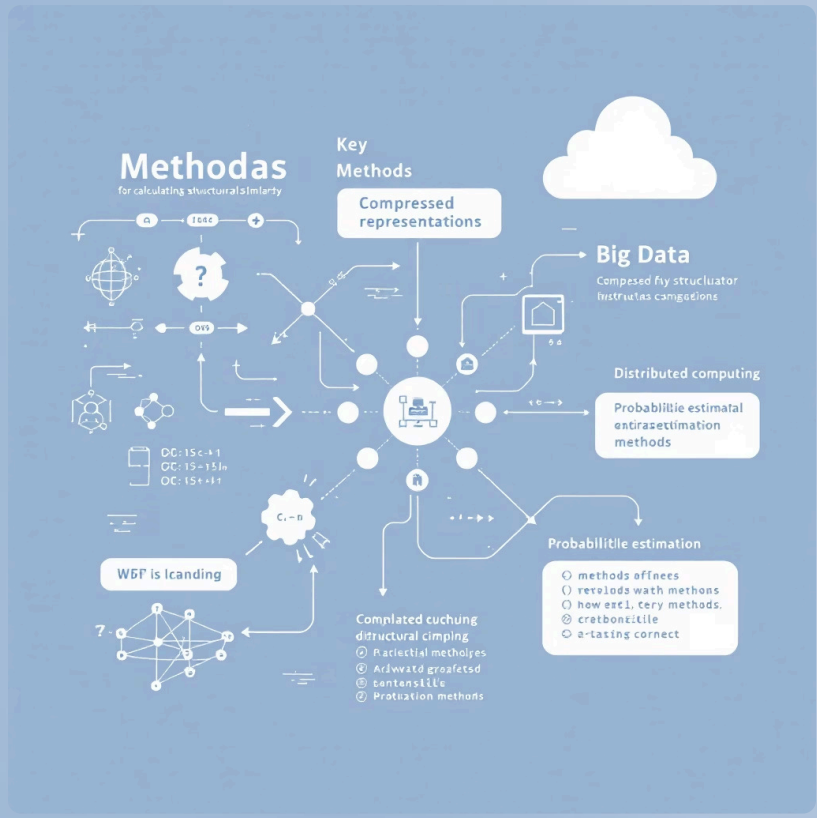
- **Procesamiento distribuido de grafos**
- Herramientas: *Apache Giraph*, *GraphX (Spark)*, *GraphFrames*, *Pregel*.
- Permiten paralelizar el cálculo de similitud y dividir el grafo entre nodos del cluster.

La similitud estructural en Big Data ya no se calcula nodo a nodo, sino mediante:

- **Representaciones comprimidas** (sketches, hashes).
- **Cómputo distribuido**.
- **Métodos de estimación probabilística**.

Ejemplo

En lugar de comparar todos los vecinos de un usuario con millones de otros, se crean firmas MinHash de sus conexiones y se agrupan los pares de alta similitud usando LSH.



Similitud Basada en Atributos en Redes Masivas

Cuando los nodos tienen muchos atributos (por ejemplo, usuarios con miles de características), calcular distancias exactas entre todos los pares es **inviable**.
Se aplican **técnicas de reducción de dimensionalidad y clustering distribuido**.

Técnicas modernas

Reducción de dimensionalidad masiva

- *PCA*, *SVD*, *Autoencoders*, *Random Projection* para condensar atributos sin perder relaciones clave.
- Implementaciones distribuidas: *Spark MLlib PCA*, *TensorFlow PCA*.

Locality Sensitive Hashing (LSH) para atributos vectoriales

- Se generan "hashes" que preservan la similitud coseno o euclidiana.
- Permite agrupar nodos similares sin comparar todo el espacio.

Modelos de embedding de atributos

- *Graph embeddings híbridos* que combinan estructura + atributos (por ejemplo, *GraphSAGE*, *Attri2Vec*).
- Escalan con GPUs o entornos distribuidos.

Búsqueda aproximada de vecinos más cercanos (ANN)

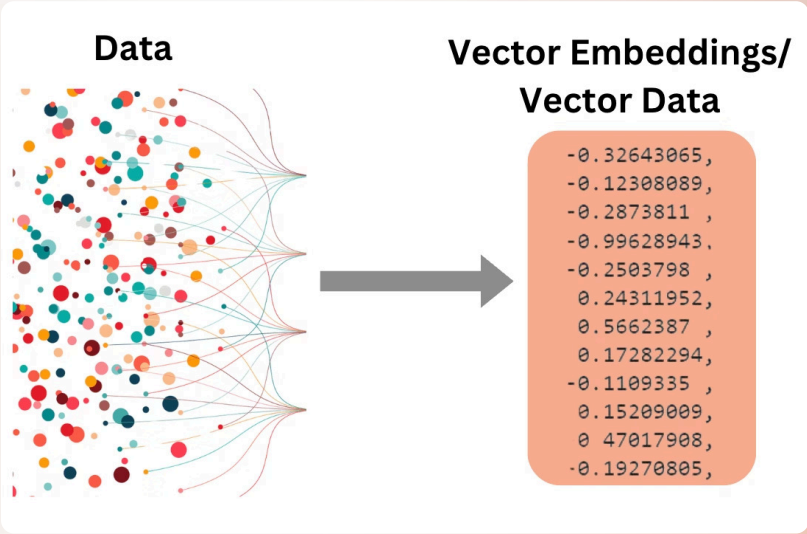
- Algoritmos como *FAISS* (Facebook AI Similarity Search) o *Annoy* permiten buscar nodos similares en millones de vectores.
- Usan estructuras jerárquicas o hashing binario.

Infraestructura Big Data

- Se aprovechan **Spark, Dask, TensorFlow** o **GPUs paralelas**.
- Los cálculos se dividen en **microbatches** o flujos de datos (*stream processing*).

Ejemplo

Si una red de productos contiene millones de ítems con vectores de características, FAISS puede encontrar los 10 más similares a un ítem en milisegundos, sin comparar contra todos.



Similitud mediante Embeddings en Grafos (Graph Representation Learning)

El aprendizaje de representaciones en grafos busca *codificar cada nodo como un vector de baja dimensión* (embedding), de forma que los nodos similares tengan vectores cercanos. Esto permite calcular similitud fácilmente con operaciones vectoriales.

Métodos populares

Métodos basados en Random Walks (Paseos Aleatorios)

- **Idea Central:** Generar secuencias de nodos (simulando "sentencias" de texto) y luego aplicar modelos de *word embedding* (como Word2Vec) para aprender las representaciones.
- **Técnica Clave: Node2Vec**
  - **Funcionamiento:** Equilibra la exploración entre **BFS (Broad First Search - Proximidad Local)** y **DFS (Depth First Search - Equivalencia Estructural)**.

Métodos basados en Redes Neuronales Gráficas (GNNs)

- **Idea Central:** Utilizar arquitecturas de Deep Learning para **aprender la representación de un nodo** agregando iterativamente información de sus vecinos.
- **Técnicas Clave:**
  - **Graph Convolutional Networks (GCNs):** Adaptación de las CNNs (Redes Convolucionales) a datos de grafo, propagando características a través de las aristas.
  - **Graph Attention Networks (GATs):** Permiten que la agregación de vecinos sea más inteligente, asignando **pesos de atención** diferentes a los vecinos más relevantes.

Cálculo de Similitud

Una vez que los nodos del grafo masivo se han proyectado en vectores ($\mathbf{v}_x, \mathbf{v}_y$), la clave para medir la relación es la fórmula que has indicado: la **Similitud del Coseno** .

$$\text{sim}(x, y) = \frac{\mathbf{v}_x \cdot \mathbf{v}_y}{||\mathbf{v}_x|| \cdot ||\mathbf{v}_y||}$$

Esta métrica es la elección fundamental en el aprendizaje de representaciones debido a que se centra exclusivamente en la **orientación angular** de los vectores, ignorando su magnitud.

Embeddings en Datos Masivos

El manejo de grafos con miles de millones de nodos y aristas requiere estrategias que van más allá de un simple modelo de entrenamiento.

Entrenamiento Distribuido y Paralelo:

Para gestionar el volumen de Big Data, el entrenamiento se realiza en *clusters* de alto rendimiento utilizando *frameworks* diseñados para la paralelización (como PyTorch Geometric o DGL).

Esto implica la **partición inteligente del grafo** (p. ej., dividiéndolo por subgrafos o aristas) para que el trabajo se distribuya eficientemente en múltiples **GPUs**, permitiendo el procesamiento simultáneo de inmensas porciones de la red.

Muestreo de Vecinos (*Neighbor Sampling*):

Esta es la técnica de escalabilidad más importante para las Redes Neuronales Gráficas (GNNs). En lugar de calcular el estado de un nodo propagando la información de **todos** sus vecinos (lo cual colapsaría la memoria para los nodos populares de alto grado), el algoritmo **muestrea aleatoriamente un número fijo de vecinos** en cada capa.

Manejo de Grafos Dinámicos (Streaming):

En entornos donde los datos llegan continuamente (como redes sociales o transacciones financieras), se emplean **métodos incrementales**. Estos evitan el costoso reentrenamiento de todo el modelo con cada nueva arista o nodo, optando por la **reinicialización parcial** o modelos de **Graph Stream Processing (GSP)** que actualizan los *embeddings* a medida que la información fluye.

Aplicaciones

La capacidad de vectorizar la estructura de datos relacionales desbloquea soluciones de alto valor en entornos de Big Data:

- **Sistemas de Recomendación y Predicción de Enlaces:** Al calcular la similitud coseno entre el *embedding* de un usuario y el de millones de ítems, se facilita la recomendación de **"cola larga"** (productos o contenidos nicho) que los sistemas basados en popularidad ignoran. Esto es clave en plataformas como Amazon o Netflix, que manejan catálogos masivos.
- **Detección de Fraude y Anomalías:** La técnica permite identificar rápidamente **grupos de nodos con baja similitud** con su comunidad o con el resto de la red. Esto se aplica en tiempo real en redes financieras gigantescas para señalar patrones de **blanqueo de dinero** o estructuras de **bots coordinados** en redes sociales, donde la anomalía está definida por la desviación en la conectividad.
- **Análisis de Redes Biológicas y Financieras:** Los *embeddings* ayudan a descubrir comunidades o *clusters* con alta similitud estructural, permitiendo el **descubrimiento de nuevas comunidades genéticas** a partir de bases de datos proteicas o la **clasificación precisa** de clientes o documentos en bases de conocimiento masivas.

Escalabilidad y Desafíos en Datos Masivos

El cálculo de similitud entre nodos en grafos con millones o miles de millones de nodos presenta retos técnicos y teóricos. La prioridad es lograr *eficiencia sin perder precisión*.

Escalamiento: El Camino hacia la Eficiencia

- **Muestreo de Vecinos (Sampling):** La estrategia principal es evitar el colapso de la memoria. En lugar de procesar la vecindad completa de un nodo (lo cual es inviable para nodos con alto grado), los algoritmos como GraphSAGE **muestrean solo un subconjunto aleatorio** de vecinos . Esto mantiene el costo computacional bajo y permite que las GNNs operen en grafos a escala de la web.
- **Computación Distribuida:** Se requiere la orquestación de *clusters* (ej. Apache Spark, DGL) para distribuir el grafo y el cálculo de *embeddings* entre múltiples **GPUs**. Esto es fundamental para la paralelización de los Random Walks (Node2Vec) o la propagación de mensajes (GNNs).

Desafíos Críticos

- **Grafos Dinámicos y *Streaming*:** Los grafos en Big Data (redes sociales, transacciones) cambian constantemente. El principal desafío es mantener los *embeddings* actualizados **sin tener que reentrenar el modelo completo** cada vez. Esto exige métodos incrementales o *online* que solo actualicen las representaciones de los nodos afectados por las nuevas interacciones.
- **Heterogeneidad:** Los grafos reales suelen tener diferentes tipos de nodos y aristas (ej. usuarios, productos, publicaciones). Integrar esta información en un único espacio vectorial de manera coherente (a través de Heterogeneous GNNs) sin simplificar demasiado las complejas relaciones es un desafío técnico constante.
- **Memoria:** A pesar del muestreo, el almacenamiento de miles de millones de vectores de *embedding* sigue siendo una preocupación. Se utilizan técnicas de **cuantificación** y **compresión** para reducir el tamaño del modelo en memoria y disco.

Conclusión

El desafío actual no es definir qué es la similitud, sino **cómo escalarla**:

- Se busca **balancear precisión y costo computacional**.
- Las técnicas modernas se apoyan en **hashing, embeddings distribuidos, y aprendizaje incremental.**
- La combinación de **infraestructura + algoritmos probabilísticos** es la clave del análisis de grafos en big data.