FACULTAD DE INGENIERÍA Y COMPUTACIÓN

Escuela Profesional de Ciencia de la Computación

Una Estructura de Datos para la Consulta Visual Interactiva por Similitud de Grandes Volúmenes de Datos Multidimensionales Mixtos Georeferenciados

Tesis

Presentado por el bachiller:

Victor Raul Stanilao Reyna Garcia

Para Optar por el Título profesional de:

Licenciado en Ciencia de la Computación

Asesor: Dr. Erick Mauricio Gomez Nieto

Arequipa, Diciembre 2021

Esta tesis va dedicado a Emidelda, Donata, Victor y Ani por ser parte de mil y un batallas para lograrlo.

Agradecimientos

Agradezco a Dios y a la Virgencita, por siempre guiar mis pasos.

A mi familia, por su amor y apoyo inconmensurable que siempre me brindan.

Agradezco a la universidad, por seguir apostando por una carrera sólida en Computación.

Mi sincero agradecimiento y la de mi familia, al Dr. Erick Gómez Nieto por creer en mi, y velar por mi formación profesional y humana.

Deseo agradecer de forma muy especial al Rector Germán Chavez, por brindarnos todo su apoyo, más aún durante el año de la pandemia del 2020.

Agradezco a todos mis profesores, especialmente a Alex, Ernesto, Yván, Gustavo, Álvaro, Yessenia, Regina, por tomarse el tiempo de vivenciar y compartir experiencias muy gratas: Peru vs Costa Rica, actividades navideñas, consejos, apoyo y sobretodo sus muestras de amistad.

Resumen

Actualmente, la visualización de *big data* representa un gran desafío a los analistas. La mixtura y el alto número de dimensiones en los datos, dificulta realizar consultas en tiempo real. Frente a ello, se propone una nueva forma de indizar grandes volúmenes de datos estáticos multidimensionales de tipos mixtos. Con el propósito de ejecutar simples consultas visuales de similitud sobre estos conjuntos de datos, las cuales estarán asociadas a una ubicación geográfica. Nuestra propuesta presenta una versión inicial donde se utilizó métodos de reducción de dimensionalidad con el propósito de mapear nuestro conjunto de datos a un espacio unidimensional, para luego organizarlos por su similitud. Posteriormente, se desarrolló una siguiente versión, mejorando las desventajas de la anterior. El cual mantiene el cálculo de la matriz de distancias de Gower, pero a partir de este se construye un árbol de similitud, que está conectado a una estructura espacial conocida como QuadTree. Adicionalmente, se presenta un prototipo de análisis exploratorio, que nos permite explotar al máximo las capacidades a ser incluidas en la estructura de datos.

Palabras Claves— Estructura de Datos, Similitud, visualización de *big data*, consultas interactivas, datos de tipo mixto, multidimensional y georeferenciado

Abstract

Currently, visualising big data represents a major challenge for analysts. The mixed and high number of dimensions in the data makes it difficult to perform queries in real time. Against this background, a new way of indexing large volumes of static multidimensional mixed-typed data is proposed. With the purpose of executing simple visual similarity queries on these datasets, which will be associated to a geographic location. Our proposal presents an initial version where dimensionality reduction methods were used with the purpose of mapping our dataset to a one-dimensional space, and then organising it by similarity. Subsequently, a next version was developed, improving the disadvantages of the previous one. It maintains the calculation of the Gower distance matrix, but from this a similarity tree is constructed, which is connected to a spatial structure known as QuadTree. Additionally, an exploratory analysis prototype is presented, which allows us to exploit to the maximum the capabilities to be included in the data structure.

Keywords— Data Structure, Similarity, big data visualization, interactive query, mixed, georeferenced and multidimensional data

Abreviaturas

QDS *Quantile Datacube Structure*

PMQ *Packet-Memory Quadtree*

STIG *Spatio-Temporal Indexing using GPUs*

TPFlow *Tensor Partition Flow*

UMAP *Uniform Manifold Approximation and Projection for Dimension Reduction*

PCA *Principal Component Analysis*

t-SNE *t-distributed Stochastic Neighbor Embedding*

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La coyuntura actual por la propagación del virus Covid-19, viene caracterizándose por una crisis e inestabilidad en todos los ámbitos de la sociedad, cambiando la manera de realizar nuestras actividades diarias. Las medidas de confinamiento aplicadas por algunos países, además del proceso acelerado de digitalización, vienen provocando un mayor uso del internet, registrándose en el Perú, un crecimiento del 22 % en el mes de junio con respecto a febrero en el último año, según el diario *"El Peruano"*. Este crecimiento favorece a una mayor disponibilidad de datos, siendo de nuestro interés, aquellas que incluyan la información espacial. Resultando útiles para responder a estas preguntas: "¿Cuáles son los distritos más similares en la provincia de Huaral con respecto a los pacientes Covid en los hospitales del MINSA?" o "¿Cuáles son las calles de la Provincia de Arequipa más similares en cuanto a crímenes?".

1.1. Motivación y Contexto

Para explorar grandes volúmenes de datos de forma visual e interactiva se requiere de un uso eficiente de los recursos computacionales, reflejados en una baja latencia durante las consultas, de tal manera que los analistas puedan realizar sus tareas sin inconvenientes. Frente a ello, diversas soluciones se han propuesto, las más predominantes se basan en Data Cubes, tales como [Lins et al., 2013], [Pahins et al., 2016], [Miranda et al., 2017], [Wang et al., 2016]; el cual es una estructura de datos que permite aplicar funciones de agregación sobre los registros, tales como suma, conteo, promedio [Gray et al., 1997]. Sin embargo, por el alto número de dimensiones que hoy presentan los datos, resulta sumamente costoso utilizar este tipo de métodos por el comportamiento exponencial que llegan a presentar.
Por otro lado, determinar la similitud entre los datos viene ligada a las métricas tales como la distancia euclideana (datos númericos), jaccard (dato categóricos), gower (datos mixtos). El desarrollo de este tipo de herramientas, representa una gran oportunidad para contribuir a la sociedad en la toma de mejores decisiones a partir de los *insights* que se puedan visualizar.

1.2. Planteamiento del Problema

Alto costo computacional durante las consultas interactivas para conjuntos de datos masivos y georeferenciados. Falta de mecanismos para el tratamiento de datos multidimensionales mixtos. Escasez de métodos para visualización por similitud de Big Data.

1.3. Objetivos

En esta sección describiremos nuestro objetivo general, seguido de los ojetivos más puntuales que se plantearon alcanzar en este trabajo de investigación.

1.3.1. Objetivo General

Indizar grandes conjuntos de datos estáticos multidimensionales de tipos mixtos para soportar consultas de similitud de forma interactiva.

1.3.2. Objetivos Específicos

- Generar una estructura de datos que nos permita la consulta interactiva por similitud de grandes volúmenes de datos georeferenciados y multidimensionales mixtos.
- Desarrollar prototipo de visualización que nos permita el análisis exploratorio de los datos almacenados en la estructuras de datos, explotando las capacidades de consulta por similitud.
- Verificar la utilidad de nuestra propuesta con múltiples estudios de caso, con datos del mundo real.

Capítulo 2

Fundamentos Teóricos

En este capítulo describiremos una serie de términos y conceptos que son utilizados en el presente trabajo, en favor de facilitar un buen entendimiento. Empezaremos definiendo un *Data Cube*, luego en qué consiste una consulta por similitud, se detallarán los diferentes tipos de datos, junto a los métodos de proyección multidimensional. Finalmente, se encontrarán las diferetes distancias usados para lidiar con datos mixtos.

2.1. Data Cubes

Data Cube es una estructura que permite realizar operaciones de agregación sobre las dimensiones de una relación. Donde una agregación, consiste en agrupar un conjunto de registros y aplicarle una función de conteo, suma, entre otros. Presenta un gran problema relacionado con el uso de memoria, ya que este aumenta a medida que se incrementa el número de dimensiones, comportándose de manera exponencial [Gray et al., 1997]. En la Fig. 2.1 se pueden reflejar estos conceptos, donde se visualiza data cubes con agregaciones hasta 3 dimensiones. Los métodos de la literatura, en su mayoría adoptan el clásico concepto de Data Cubes, On-Line Analytical Processing, con ciertas variaciones, pero en esencia aprovechando la recuperación de un número de agregaciones a partir de un conjunto base, siendo codificados en una estructura esparza basada en punteros con el fin de representar un Data Cube eficientemente.

2.2. Consulta por similitud

Las consultas por similitud encuentran objetos similares a un objeto de consulta dado bajo un cierto criterio [Yang et al., 2019].

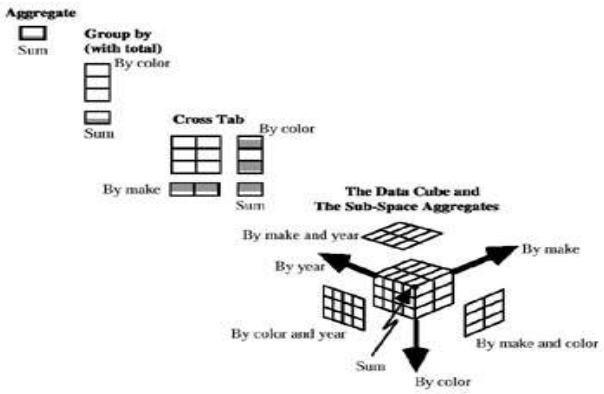


Figura 2.1: La representación para un Data cube de 0 dimensiones es un punto;1 dimensión, una línea con un punto; 2 dimensiones, una tabla, plano, dos líneas y un punto; 3 dimensiones, un cubo con 3 tablas interceptadas. Figura extraída de [Gray et al., 1997].

2.3. Tipos de datos

Existen diversos tipos de datos, de las que se derivan las siguientes:

- **Datos multidimensionales**: antes de estudiar qué son los datos multidimensionales, es importante entender dos términos claves: **objeto** y **características**. Un objeto puede abarcar varias cosas: personas, alumnos universitarios, autos, laptops, etc. Asimismo, este objeto presenta varias cualidades. En cuanto al alumno universitario, podemos describirlo por las siguientes características: nombre, edad, sexo, escuela, carrera profesional y promedio ponderado. Los objetos también reciben el nombre de *items*, instancias, muestras y observaciones; mientras que las características, atributos, parámetros, propiedades, variables y dimensiones [Dzemyda and Kurasova, 2015]. Un conjunto de datos estará conformado por objetos que sean descritos por las mismas características $x_1, x_2, ..., x_n$. Supongamos que cualquier característica puede tomar valores numéricos. Entonces, una combinación de estos valores caracteriza a un objeto en particular:

  $$Xi = (x_{i1}, x_{i2}, ..., x_{in}), i \in 1, ..., m,$$

  Donde i es el número de orden del objeto; n, el número de características y m, el número de objetos. Si estos objetos presentan más de una característica, los datos de estos objetos son llamados datos multidimensionales.

- **Datos Mixtos**: Conjunto de datos en los que se describen tanto por variables cuantitativas como cualitativas. Además, el término mixto se refiere a la presencia simultánea de estos tipos de variables.

- **Datos Espaciales**: Los datos espaciales contienen información geográfica relativa a la tierra y sus características. A través de un par de coordenadas: latitud y longitud, se define una ubicación específica en la tierra.

  Pueden clasificarse en dos tipos según la técnica de almacenamiento: datos ráster y datos vectoriales. Los datos ráster, que pueden ser imágenes satelitales o fotografías,

se representan por una malla cuadriculada donde cada celda es identificadas por la fila y columna. El área geográfica se divide en grupos de células individuales, que representan una imagen. Por otro lado, los datos vectoriales están compuestos por puntos, polilíneas y polígonos [Janipella et al., 2019].

- **Datos Temporales**: El término temporal según Jensen et al. [1997] se aplica a datos que incorporan el tiempo durante el cual es válido o se está en vigor. El propósito principal de estos datos es proporcionar información sobre los estados anteriores. En otras palabras, la información histórica de todas las transacciones de la base de datos están determinada por algún elemento temporal. Con bases de datos temporales, un atributo puede recibir más de un valor. Por ejemplo, el estado civil de una persona puede ser tanto casado como soltero en diferentes momentos de la base de datos.

2.4. Métodos de proyección multidimensional

En este apartado encontraremos las definiciones y sus características más importante de los métodos más recientes del estado del arte tales como *t-SNE, UMAP, PCA*. Además, encontraremos las ventajas y desventajas que ofrece una sobre otra.

2.4.1. t-SNE

t-distributed Stochastic Neighbor Embedding **(t-SNE)** [van der Maaten and Hinton, 2008], es una técnica de aprendizaje de máquina para la reducción de la dimensionalidad que permite identificar patrones relevantes en un conjunto de datos. Su principal ventaja es la preservación de la estructura local. Esto significa, aproximadamente, que los puntos que están cerca unos de otros en el conjunto de datos de alta dimensión tenderán a estar cerca unos de otros en el gráfico.

Este algoritmo modela la distribución de probabilidad de los vecinos alrededor de cada punto. En el espacio original, de muchas dimensiones, se modela como una distribución Gaussiana. Mientras que en el espacio nuevo, bidimensional, se modela como una distribución *t*. El objetivo es encontrar un mapeo en el espacio bidimensional que minimice las diferencias entre estas dos distribuciones en todos los puntos. El principal parámetro que controla el ajuste se llama perplejidad. La perplejidad es aproximadamente equivalente al número de vecinos más cercanos que se consideran al hacer coincidir las distribuciones original y ajustada de cada punto. Una baja perplejidad significa que nos preocupamos por la escala local y nos centramos en los otros puntos más cercanos. Una alta perplejidad requiere más de un enfoque de çuadro general".

Debido a que las distribuciones están basadas en la distancia, todos los datos deben ser numéricos. Debes convertir las variables categóricas en numéricas mediante una codificación binaria o un método similar. También suele ser útil normalizar los datos, de modo que cada variable esté en la misma escala. Esto evita que las variables con un rango numérico mayor dominen el análisis.

Es importante señalar que t-SNE sólo funciona con datos estáticos. Es decir, no produce un modelo que pueda aplicarse a nuevos datos.

2.4.2. UMAP

Uniform Manifold Approximation and Projection for Dimension Reduction **(UMAP)** [McInnes et al., 2020], en esencia es muy similar al t-SNE, ya que ambos utilizan algoritmos de disposición de gráficos para organizar los datos en el espacio de baja dimensión. Es decir, UMAP construye una representación gráfica de alta dimensión de los datos y luego optimiza un gráfico de baja dimensión para que sea estructuralmente, lo más similar posible.

UMAP para construir el gráfico inicial de alta dimensión, construye una representación de un gráfico ponderado, con pesos de borde que representan la probabilidad de que dos puntos estén conectados, que será denominado como *fuzzy simplicial complex*. Para determinar la conexión, UMAP extiende un radio hacia afuera de cada punto, conectando los puntos cuando estos radios se superponen. Elegir este radio es crítico, ya que una elección demasiado pequeña conducirá a pequeños cúmulos aislados, mientras que una elección demasiado grande conectará de forma conjunta todos los puntos. Frente a ello, UMAP supera este problema eligiendo un radio localmente, basado en la distancia al vecino más cercano de cada punto. UMAP entonces hace el gráfico *fuzzy* disminuyendo la probabilidad de conexión a medida que el radio crece. Finalmente, al estipular que cada punto debe estar conectado al menos con su vecino más cercano, UMAP asegura que la estructura local se preserve en equilibrio con la estructura global. A continuación mostraremos la Fig. 2.2. [1]

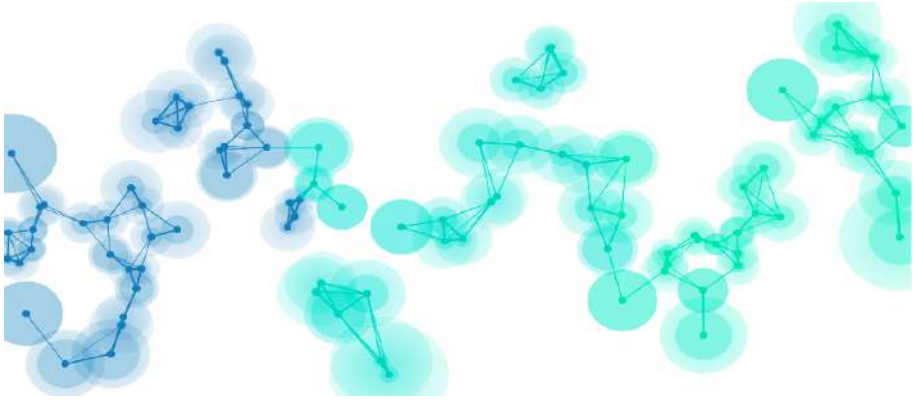


Figura 2.2: Resultado de ajustar un radio hacia afuera de cada punto, calculado por la distancia a su vecino más cercano al n-ésimo. Note que más allá de la intersección con el primer vecino, el radio comienza a ser difuso, con las conexiones subsiguientes apareciendo con menos peso. Figura extraída del portal *https://pair-code.github.io*.

[1] https://pair-code.github.io/understanding-umap/

2.4.3. PCA

Principal Component Analysis **(PCA)** [Wold et al., 1987], tiene como idea principal la reducción de dimensiones de un conjunto de datos compuesto por muchas variables correlacionadas entre sí, ya sea en gran medida o ligeramente, conservando al mismo tiempo la variación presente en el conjunto de datos. Lo mismo se hace transformando las variables en un nuevo conjunto de variables, que se conocen como los componentes principales y son ortogonales, ordenadas de tal manera que la retención de la variación presente en las variables originales disminuye a medida que bajamos en el orden. Así, de esta manera, el primer componente principal retiene la máxima variación que estaba presente en los componentes originales. Los componentes principales son los vectores propios de una matriz de covarianza, y por lo tanto son ortogonales.

Intuitivamente, el Análisis de Componentes Principales puede proporcionar al usuario una imagen de menor dimensión, una proyección o "sombra" de este objeto cuando se ve desde su punto de vista más informativo.

2.5. Medidas de similitud para datos Mixtos

Dentro de los diferentes tipos de distancias, hemos tomado las más importantes para datos mixtos, según Muñoz Salas [2019]. Entre ello destaca las distancias de Gower, aquellas basadas en jerarquías y finalmente la de Goodall. A continuación, detallaremos cada una de ellas con sus respectivas fórmulas de cálculo.

2.5.1. Distancia de Gower

La Distancia de Gower [Gower, 1971], es una métrica usada para medir cuán diferentes son dos registros a través del cálculo de una matriz de distancias para un conjunto de datos mixtos. Es decir, los registros pueden contener una combinación de datos lógicos, categóricos, numéricos o de texto. La distancia es siempre un número entre 0 y 1.

En el caso de un atributo categórico k, la distancia de los puntos p y q se obtiene a través de la siguiente Ecuación 2.1.

$$\delta\left(p_k, q_k\right) = \begin{cases} 0, & p_k = q_k \\ 1, & p_k \neq q_k \end{cases} \tag{2.1}$$

Mientras que para un atributo numérico l, la distancia se calcula usando esta Ecuación 2.2:

$$\delta\left(p_l, q_l\right) = \frac{|p_l - q_l|}{R_l} \tag{2.2}$$

donde R_l es el rango del atributo l en todas sus instancias. Tras el cálculo de estas

distancias parciales, las promediamos para obtener la distancia de Gower, obteniendo la fórmula general en la siguiente Ecuación 2.3:

$$D_{\text{Gower}}\left(x_1, x_2\right) = 1 - \left(\frac{1}{n}\sum_{j=1}^{n}\delta_j\left(x_1, x_2\right)\right) \tag{2.3}$$

2.5.2. Distancia basada en jerarquías

Esta distancia consiste en realizar cálculos que permiten encontrar la disimilitud entre los datos preservando la relación semántica de las diferentes categorías que poseen sus atributos categóricos. Se crea un árbol de jerarquía para cada atributo. Para los atributos categóricos nominales, cada nodo hoja representa una categoría del atributo. Mientras que para los datos categóricos ordinales o datos numéricos los puntos pueden encontrarse en las aristas del árbol. Se considerará como valores similares aquellos que se encuentren bajo el mismo nodo padre. La construcción de cada una de las jerarquías puede realizarse de forma manual o a través de técnicas de agrupación jerárquicas [Muñoz Salas, 2019].

El cálculo de la distancia entre dos puntos sigue la siguiente Ecuación 2.4.

$$\delta\left(p_i, q_i\right) = d_{p_i} + d_{q_i} - 2d_{LCP(p_i,q_i)} \tag{2.4}$$

donde d_{pi} y d_{qi} son las distancias de los atributos de los puntos pi y qi a la raíz del árbol respectivo. LCP es el punto en común más cercano entre pi y qi en el árbol de la jerarquía y $d_{LCP}(pi, qi)$ es la distancia del punto LCP a la raíz.

Es importante resaltar que las distancias $D(pi, qi)$ son calculadas independientemente para cada atributo i para cada par de objetos (pi, qi) del conjunto de datos, manteniendo su estructura jerárquica correspondiente. Tras el cálculo de las distancias parciales, aplicamos la siguiente función de agregación, presentada en la Ecuación 2.5:

$$D(p, q) = \left(\sum_{i=1}^{n} w_i\left(\delta\left(p_i, q_i\right)\right)^L\right)^{1/L} \tag{2.5}$$

donde n representa la totalidad de dimensiones; w_i, peso asociado para el atributo i y L, una constante de valor entero.

2.5.3. Distancia de Goodall

Esta distancia fue propuesta inicialmente por Goodall [1966], y años más tarde generalizada por Li and Biswas [2002] para medir la similitud entre objetos de cualquier dominio. Esta técnica a diferencia de otras considera la frecuencia y la unicidad de los valores de los atributos presentes en los datos. En los atributos categóricos, el mayor

aporte a la similitud entre dos objetos con valores comunes, es obtenida por los valores menos frecuentes en en el conjunto de datos. Para el atributo categórico k, la distancia se calcula mediante la Ecuación 2.6:

$$\delta\left(p_k, q_k\right) = \begin{cases} 1, & p_k \neq q_k \\ \sum_{l \in MSFVS(p_k, p_k)} \frac{(f_l)_k \cdot \left((f_l)_k - 1\right)}{m(m-1)}, & p_k = q_k \end{cases} \tag{2.6}$$

donde m es el número total de registros del conjunto de datos. $(f_l)_k$ es la frecuencia de ocurrencia del valor l con respecto al atributo k. Finalmente $MSFVS(p_k, p_k)$ es el *More Similar Feature Value Set*, es decir, el conjunto de todos los pares cuya frecuencia de ocurrencia es igual o menor que el par actual.

Para los atributos numéricos, la similitud radica en la magnitud del intervalo y la distribución de los datos, calculado con la siguiente Ecuación 2.7.

$$D\left(p_u, q_u\right) = \sum_{l \in MSFSS(p_u, q_u)} \left(\alpha_{pq}\right)_u \tag{2.7}$$

donde $MSFSS(p_u, q_u)$ es el *More Similar Feature Segment Set*, es decir, el conjunto de todos los pares con menor magnitud que el par (p_u, q_u), junto con los pares de igual magnitud pero menor número de elementos incluidos en el intervalo del par (p_u, q_u). Por otro lado $\left(\alpha_{pq}\right)_u$ representa la probabilidad de ocurrencia del par (p, q) en los datos con respecto al atributo u. Está dado por la siguiente Ecuación 2.8:

$$\left(\alpha_{pq}\right)_u = \begin{cases} \frac{2 \cdot (f_p)_u (f_q)_u}{m(m-1)}, & \left(\alpha_p\right)_u \neq \left(\alpha_p\right)_u \\ \frac{(f_p)_u \left((f_p) u^{-1}\right)}{m(m-1)}, & \left(\alpha_p\right)_u = \left(\alpha_p\right)_u \end{cases} \tag{2.8}$$

Al igual que las distancias anteriores, las distancias parciales son calculadas para cada atributo con la Ecuación 2.9 y después agregadas con la Ecuación 2.10:

$$X(p, q) = 2 \sum_{i=1}^{t_{CAT}} 1 - \frac{\delta\left(p_i, q_i\right) \cdot \ln\left(\delta\left(p_i, q_i\right)\right) - \left(\delta\left(p_i, q_i\right)\right)' \cdot \ln\left(\delta\left(p_i, q_i\right)\right)'}{\delta\left(p_i, q_i\right) - \delta\left(p_i, q_i\right)'} - 2 \sum_{i=1}^{t_{NUM}} \ln\left(\delta\left(p_i, q_i\right)\right) \tag{2.9}$$

$$D(p, q) = \exp\left(-\frac{X(p, q)}{2}\right) \sum_{i=0}^{t_{NUM} + t_{CAT} - 1} \frac{\left(\frac{1}{2}\right) X(p, q)\Big)^i}{i!} \tag{2.10}$$

donde t_{CAT} y t_{NUM} representan el total de atributos categóricos y numéricos respectivamente. La función $\lambda(p_i, q_i)$ representa la disimilitud entre las instancias p, q en relación con el atributo i. $\lambda(p_i, q_i)'$ es la siguiente diferencia menor con respecto al atributo categórico i.

Capítulo 3

Estado del Arte

En este capítulo describiremos varios métodos de indización del estado del arte que vienen abordando el problema de la exploración visual de grandes volúmenes de datos espacio-temporales. Para ello, hemos realizado una clasificación, donde se obtuvieron cuatro grupos: métodos basados en Data Cubes, métodos basados en Tensores, métodos basados en Redes Neuronales, métodos basados en la sensibilidad semántica, y aquellos acelerados por GPU.

3.1. Métodos basados en Data Cubes

Los siguientes métodos se apoyan en el concepto de un data cube, realizando el precálculo de las agregaciones sobre varias dimensiones.

imMens, propuesto por Liu et al. [2013], es el primer sistema que permitió analizar millones de datos en tiempo real de forma interactiva. Aborda los problemas de escalabilidad perceptual e interactiva, teniendo como principio que la escalabilidad debe estar limitada por la resolución escogida mas no por el número de registros.

Para la escalabilidad perceptual, realiza la reducción de datos a través del método *binned aggregation*, así como también la descripción de un espacio de *binned plots* para diferentes tipos de variables (numéricas, ordinales, temporales y geográficas). En cuanto a la escalabilidad interactiva, precalcula las proyecciones de las vistas materializadas, denominadas *data tiles multivariados*, descomponiendo un data cube en proyecciones de 3 y 4 dimensiones. Además, paraleliza el renderizado y procesamiento de datos usando un esquema de indexación densa.

ImMens obtiene un desempeño de 50 fps (frames por segundo) en las operaciones de *brushing* y *linking*, permitiendo interacciones sobre millones de registros en tiempo real. Sin embargo, no soporta operaciones compuestas y específicas en más de 4 dimensiones. No hay garantía de una buena calidad de las visualizaciones por el uso de WebGL, dado que aspectos como la textura varían entre diferentes máquinas y tarjetas gráficas. Precalcula y

almacena todos los posibles data tiles en el servidor, lo cual no será viable para conjuntos de datos con alta dimensionalidad por el alto costo que demandaría.

Nanocubes, propuesto por Lins et al. [2013], permite consultar valores agregados sobre datos espacio temporales de varias dimensiones que fueron precalculados. Aprovecha la dispersidad que pueden presentar los Data Cubes creando solo los necesarios, y estableciendo un compartimiento de nodos mediante enlaces compartidos.

Su recorrido siempre empieza con la dimensión espacial, indexada por un quadtree tradicional, donde cada nodo a través de un puntero adicional puede pasar a la siguiente dimensión. Luego, la dimensión categórica, donde se utiliza un árbol plano cuyo nodo raíz puede llegar a tener un número de hijos igual a los diferentes valores presentes en la categoría. Finalmente sigue la dimensión temporal, utilizando una estructura basada en summed-area tables, que resulta equivalente a un histograma de sumas acumuladas. Por ello, para consultas de agregación sobre cualquier periodo de tiempo se realizará dos búsquedas binarias para encontrar los elementos delimitadores del intervalo seguido de una diferencia entre estos elementos. Podemos observar la construcción de los Nanocubes en la Fig. 3.1.

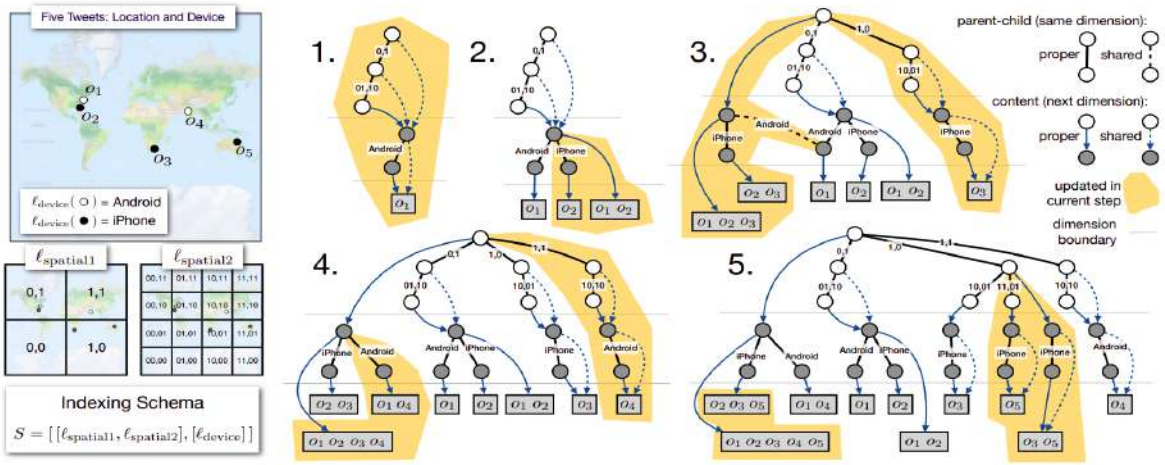


Figura 3.1: Ilustrando la construcción de un nanocube para cinco puntos $[o1, ..., o5]$ bajo el esquema S. Figura extraída de [Lins et al., 2013].

Por la existencia de enlaces compartidos entre los nodos se logra reducir la redundancia de información y por ende ahorrar memoria. La forma de almacenar las series temporales resulta óptima y simple, ya que no se requiere otra estructura de árbol a indexar, además se preservan diferentes niveles de información. Los tiempos de consultas para construir las codificaciones visuales son a lo mucho proporcional al tamaño del número de píxeles de la pantalla. Permite combinar agregaciones de diferentes dimensiones en niveles de detalle independientes.

Presenta el fenómeno de saturación de llaves, haciendo que las nuevas ocurrencias de claves ya insertadas no requieran espacio adicional de almacenamiento. Nanocubes solo permite añadir objetos, por ello, no pueden ser actualizados si un registro es eliminado de la

relación base. La implementación actual del nanocube permite solo una dimensión espacial y una dimensión temporal. Por otro lado, no ha sido optimizada usando multithreading ni memory pools. No es multidimensional realmente, ya que en uno de sus experimentos al indexar 6 dimensiones se consumió alrededor de 45 GB. Nanocube no permite consultas por registros individuales.

Hashedcubes [Pahins et al., 2016], es una estructura de datos que se caracteriza por usar una jerarquía de pivotes similar a la jerarquía presente en los árboles, que son obtenidos tras realizar ordenamientos sobre las dimensiones de los datos. Esta idea se basa en las siguientes consideraciones: El tamaño de un arreglo no se altera al realizarse ordenaciones en sus elementos. Los subconjuntos de datos de un arreglo pueden representarse por dos pivotes que indican el comienzo y final. Finalmente, si respetamos el ordenamiento de estos subconjuntos, podemos realizar ordenamientos dentro de cada uno de ellos.

Hashedcubes inicia su construcción representando el conjunto total de datos con un único pivote raíz. A medida que se van generando las particiones, se almacenarán los pivotes que las delimitan, seguido de ordenamientos para agrupar sus elementos. Para el caso de la dimensión espacial, serán agrupados aquellos que pertenezcan a la misma región; dimensión categórica, bajo un mismo valor específico o rango y para la dimensión temporal en términos de su época de tiempo. El orden de las dimensiones al construir Hashedcubes es libre, y afectará el consumo de memoria y tiempo de respuesta de algunas consultas.

Hashedcubes permite saltarnos subconjuntos enteros de datos, gracias a la presencia de pivotes que contienen información del rango de valores que referencian, por ello requiere menos cantidad de memoria que Nanocubes e imMens. Este uso de memoria es directamente proporcional al número de pivotes. Al igual que Nanocubes presenta una saturación de llaves. Hashedcubes permite la recuperación de registros específicos asociando los índices de los pivotes con índices externos. Asimismo, permite la visualización de datos que presentan un punto de origen y otro de destino.

A diferencia de Nanocubes e Inmens, HashedCubes no realiza el precálculo de agregaciones de todas las posibles combinaciones, en su lugar, calcula las faltantes por medio de la jerarquía de pivotes durante el tiempo de ejecución. Hashedcubes realiza solo el procesamiento de datos estáticos, aún no se ha probado sobre flujos continuos de datos. Aunque Hashedcubes permite ahorrar el uso de la memoria, su tiempo de consulta puede ser peor que algunas otras estructuras como Nanocubes, en casos donde se necesite recuperar muchas agregaciones proveniente de particiones pequeñas.

Gaussian Cubes, propuesto por Wang et al. [2016], es una extensión de los Nanocubes, cuya idea principal es almacenar valores estadísticos para calcular los momentos de segundo orden de un subconjunto de variables. Este precálculo de la mejor distribución de los datos permite dotar de capacidades de modelamiento a las exploraciones visualizaciones interactivas. Gaussian Cubes es concebido como un DataCube espacio temporal contando con la información estadística necesaria para modelar distribuciones multivariables. Por otro lado, hereda las principales características de los Nanocubes, tales como tiempos de ejecución, uso de memoria entre otros.

Gaussian Cubes permite realizar gráficos aproximados que resultan útiles en la práctica, ya que resaltan aspectos relevantes de los datos. Gaussian Cubes aún no presenta un respaldo teórico que garantice su buen funcionamiento, solo casos prácticos hasta el momento. Asimismo, carece de una métrica que mide el grado de ajuste de los modelos. Los modelos soportados Gaussian Cubes están limitados por los valores agregados a pre calcular.

Gaussian Cubes consume más memoria que un Nanocube, teniendo un costo cuadrático al indexar las dimensiones. Sin embargo, su tiempo de ejecución puede llegar a ser independiente del tamaño total de datos. Asimismo, mantiene un tiempo de latencia aceptable durante las consultas.

TopKube [Miranda et al., 2017], es una estructura de datos que extiende las funcionalidades de los Nanocubes, calculando las k consultas más relevantes con una latencia necesaria para permitir la exploración visual interactiva. TopKube es similar a los Nanocubes en cuanto al precálculo de múltiples agregaciones permitiendo una respuesta rápida. Sin embargo, agrega una dimensión, denominada la dimensión clave, para contener información del ranking.

La representación de todas las dimensiones de TopKube son similares a las de Nanocubes, excepto la dimensión clave, que está compuesto de arreglos del mismo tamaño, las cuales pueden verse reflejados en la Fig. 3.2 donde en cada hoja se almacena una tabla para calcular el ranking de cada objeto. TopKube a diferencia de imMens y Nanocubes, busca responder consultas más detalladas que simple cantidades, es decir, preguntas tales como *"Cuáles son los proyectos en Github con más commits en la Costa Oeste"*, o *"cuáles fueron las etiquetas de las más imágenes más populares que fueron subidos de París"*.

TopKube es capaz de generar listas rankeadas con un orden de magnitud menor que las técnicas previas. No presenta optimizaciones para el uso de la memoria ni en los tiempos de construcción de la estructura, tales como *multithreading*.

Quantile Datacube Structure **(QDS)** propuesto por Pahins et al. [2019], es una estructura de datos que proporciona buenas aproximaciones de la distribución de los datos espacio-temporales. Similar a las técnicas basadas en DataCubes, realiza precálculos de las agregaciones. QDS almacena un quantile sketch en la forma de arreglo en cada nodo de un Datacube permitiendo una eficiente selección de datos y una consulta sobre las estadísticas de la distribución presente, cuyos resultados devueltos sean precisos.
Entre las diferencias presentes entre QDS y HashedCubes, tenemos que el índice de pivote de QDS soluciona el problema de la fragmentación de pivotes en HashedCubes. Por otro lado, QDS mantiene un arreglo adicional de pivotes secundarios, siendo útiles para recuperar todos los pivotes correspondientes a un valor dado. Además permite que las claves asociadas a los pivotes sean almacenadas solo una vez, logrando ahorrar memoria.

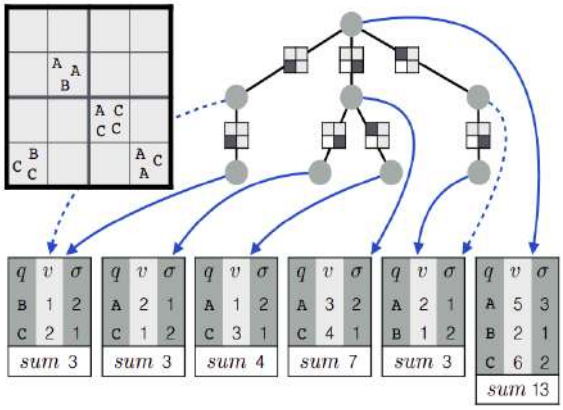


Figura 3.2: Ejemplo de un TopKube. Figura extraída de [Miranda et al., 2017].

La discretización de las dimensiones temporales y espaciales también es necesaria en QDS, similar a los NanoCubes y HashedCubes, sin embargo a diferencia de estas estructuras, QDS puede intercalar o agrupar las dimensiones libremente. El proceso de consulta empieza con la selección de los pivotes que satisfacen dicha consulta para cada dimensión. Luego se da la intersección, donde se combinan los pivotes obtenidos anteriormente para lograr responder a la consulta en todas sus dimensiones. Al final, se realiza la agregación, donde se agrupan aquellos pivotes que al unirse representan el mismo valor.

QDS a partir de sus consultas de cuantiles proporciona dos tipos de mapas de calor, el primero denominado quantile heatmaps, los cuales presentan robustez frente a los outliers; el segundo, cdf heatmaps, muestran las probabilidades que una distribución consiga ser más pequeño que un valor dado para una determinada ubicación. El uso de cuantiles, obtenidos a partir de la distribución de los datos, permite a QDS identificar eventos relevantes y complejos, llegando incluso a rescatar patrones que no fueron descubiertos por herramientas de exploración visual, debido a su capacidad interactiva al recuperar la distribución aproximada de ciertos datos.

Debido a que el ajuste de cuantiles de una distribución paramétrica como los GaussianCubes resulta muy lejano, QDS se convierte en una gran alternativa para el análisis de grandes conjuntos de datos espacio-temporales.

Entre las desventajas del QDS encontramos que solamente trata con distribuciones univariadas. Además, no brinda información al usuario sobre los límites del error en las aproximaciones.

SmartCube, propuesto por Liu et al. [2019], es una variación de los Data Cube para visualizar conjuntos de datos espacio-temporales. Asimismo, se caracteriza por su adaptabilidad a los diferentes patrones de consultas. La idea principal es la actualización realizada sobre su unidad básica, un cuboide, a través de la creación y eliminación de estos de forma dinámica, ya sea cuando las consultas coincidan o fallen.

SmartCube construye los índices para las dimensiones espaciales y categóricas basados en Nanocubes, con la diferencia que cada cuboide es independiente. La dimensión categórica es indizada directamente por su valor mientras que las otras dos requieren discretizaciones y subdivisiones. En la dimensión espacial se construyen capas jerárquicas utilizando el Quadtree. En cuanto a la dimensión temporal, dada la independencia o dependencia que pueden presentarse entre sus niveles, se introducen definiciones como la de orden parcial cuando un nivel cubre totalmente al siguiente y las listas de niveles de orden parcial (POLL), donde los caminos generados incluye un registro del tiempo.

Para realizar las búsquedas de las consultas en SmartCube, primero debemos encontrar el mejor cuboide de apoyo (BSC), es decir aquel cuboide que además de satisfacer la consulta, requiere del menor número de cálculos. Luego, navegar hacia este cuboide para recuperar los resultados. Finalmente, obtener las agregaciones solicitadas.

Una característica importante de SmartCube, es que además de soportar consultas sobre la dimensión espacial y categórica, permite realizar consultas sobre series temporales dada la resolución y rangos de tiempo especificados. Frente al problema de *"cold-start"*, SmartCube emplea un modelo de aprendizaje de máquina logrando predecir los cuboides más importantes para añadirlos a la estructura de datos. Su entrada estará comprendida por la información de los datos y la salida, por los cuboides predecidos a agregar.

Una peculiaridad de SmartCube frente a otras técnicas similares, es la variabilidad de la ruta de una misma consulta debido a los cambios que pueden presentar los cuboides en la estructura. La eficiencia conseguida por SmartCube se debe a que conserva los cuboides más importantes y elimina a los demás. A continuación, en la Fig. 3.3 se muestra el flujo principal de SmartCube. Según el conjunto de datos espacio-temporales original, el algoritmo construye la estructura inicial con el cuboide más fino, es decir, el que contiene los agregados más finos.

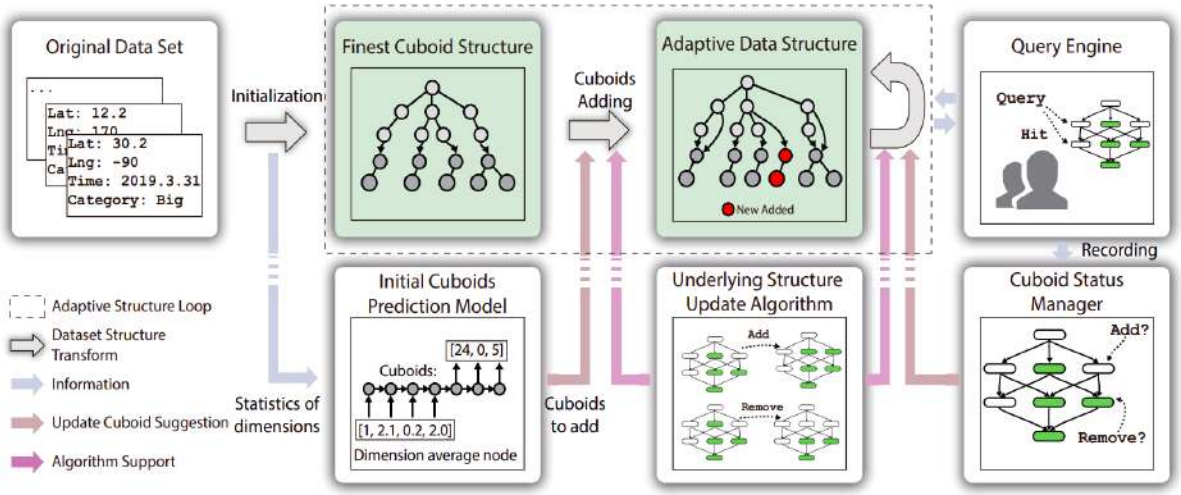


Figura 3.3: Flujo de trabajo de Smart Cubes. Figura extraída de [Liu et al., 2019].

Pese a que en un inicio, SmartCube al igual que HashedCube requiere de una cantidad significativa de cálculos para dar respuesta a una consulta, esta pronto desaparece ya que gracias a su flexibilidad consigue obtener los resultados directamente.

SmartCube logra igualar el gran desempeño de NanoCubes evitando que se necesite calcular ciertos cuboides que presenten consultas similares. Una limitación de esta estructura es que solo trabaja sobre un conjunto de datos estáticos.

ConcaveCubes propuesot por Li et al. [2018], es una estructura de datos en forma de árbol, donde el resultado de las agrupaciones jerárquicas se organizan en data cubes. Utilizan visualizaciones del tipo cluster maps basados en las delimitaciones, a través del cálculo de los polígonos cóncavos para preservar la información geográfica.

La construcción del ConcaveCubes empieza realizando un preprocesamiento de los datos. Para cada dimensión geo-independiente y geo-semántico, se realiza un ordenamiento. Luego, se dividen los valores de las dimensiones geo-independientes, en valores específicos o rangos, para convertirlos en valores categóricos. Esta etapa finaliza mapeando cada punto de dato a su región geo-semántica respectiva.

Tras el preprocesamiento, se realiza un ordenamiento multi-criterio similar a Hashedcubes propuesto por Pahins et al. [2016]. Sin embargo, se diferencia que las características geográficas no se procesan con el QuadTree. Sino más bien, diseñan un nivel geográfico basado en la jerarquía del mundo real, por ejemplo: país, estado, ciudad, etc. Luego, se realiza una clusterización jerárquica para descubrir nuevos subgrupos. Para ello utilizan DBScan, disminuyendo el valor del *threshold* en cada nivel inferior de la estructura.

Finalmente, se realiza el cálculo del polígono cóncavo para soportar los *cluster maps* basados en limitaciones. Se realiza la triangulación global sobre un conjunto de *clusters* que tengan el mismo nodo padre. La información estadística de cada agrupamiento que presente un polígono cóncavo asociado, será almacenado como *views* en la base de datos. En la Fig. 3.4, se muestra esta estructura datos para un caso de estudio con datos de propiedades.

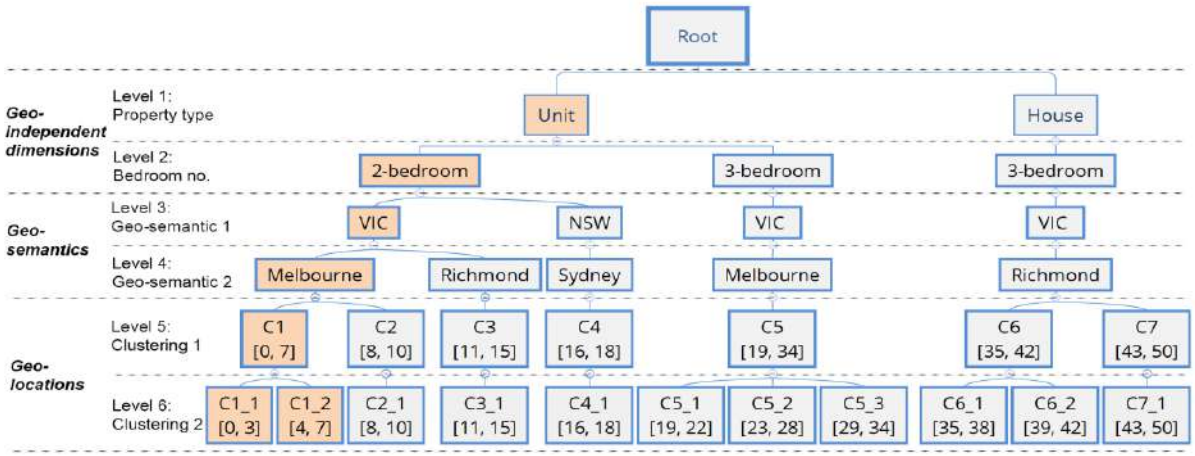


Figura 3.4: Un ejemplo de ConcaveCubes con datos de propiedades de Australia. Figura extraída de Li et al. [2018].

Sus principales ventajas son las siguientes: Soporta visualizaciones de tipo cluster maps a diferencia de las técnicas del estado del arte. Utiliza adecuadamente la información geográfica del mundo real en lugar de las agregaciones basadas en grillas. Puede soportar datos dinámicos, ya que permite insertar, actualizar y eliminar los datos modificando los nodos afectados en la estructura. ConcaveCubes utiliza cálculo y resultados de visualizaciones pre-existentes para soportar diferentes interacciones tales como zooming, panning, y filtrado.

Presenta la siguientes desventajas: No soporta datos con un número alto de dimensiones, al igual que Nanocubes, y Hashedcubes soporta a lo mucho 5 a 10 dimensiones categóricas. Los clusters maps basados en burbujas presentan información geográfica con baja precisión. Los clusters maps basado en limitaciones, no presentan adecuadamente la información de densidad.

Z-Order Curves + HashedCubes, propuesto por Peralta-Aranibar et al. [2019b], se indizan millones de elementos con el objetivo de realizar simples y múltiples consultas por similitud sobre datos multidimensionales georeferenciados. Para ello, mapean los datos del espacio N-dimensional a 1-D, usando el algoritmo de Z-Curve, para conseguir una representación más compacta y fácil de ordenar por similitud.

Como paso previo a la indización, se realiza la proyección a 1D de los datos comparando los siguientes métodos: Fiedler vector, Z-Order Curve y Hilbert Curve. Donde, el segundo obtuvo mejores resultados al medir la preservación de vecindad y costo computacional en el nuevo espacio. El valor z de un punto se obtiene realizando desplazamientos de bits.

A continuación, con las nuevas representaciones de los datos, se toma el concepto de HashedCubes, para indizar los datos en un único array. Primero, se ordenan los datos por los atributos espaciales. Se mantiene la idea de usar el *Quadtree*, obteniendo un par de pivotes que delimiten los datos para cada nodo. Luego, se realiza un ordenamiento en función de sus valores de similitud, es decir los *z-values*. De esta manera, dada una consulta por rangos, se consigue obtener que los datos más similares se ubiquen contiguos dentro del array. En la Fig. 3.5 se muestra una vista general como la estructura fue construída.

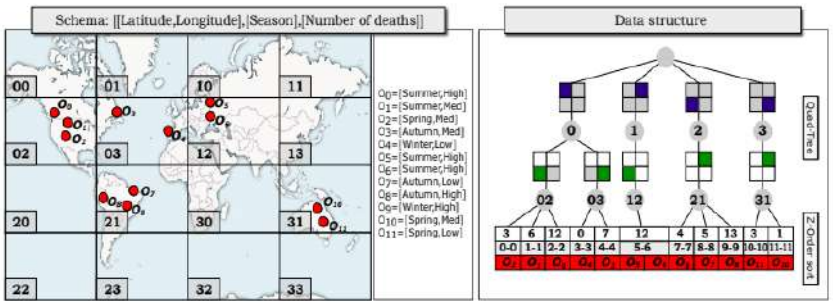


Figura 3.5: Construcción de la estructura presentado en este apartado. Figura extraída de Peralta-Aranibar et al. [2019b].

Presenta ventajas tales como la facilidad de graficar las distribuciones de similitud de un conjunto de datos dado una serie de puntos de consultas sobre un mapa geográfico.

Presenta las ventajas de Hashedcubes, en cuanto al uso eficiente de memoria y tiempo. La reducción de dimensionalidad evita lidiar con la complejidad de muchas dimensiones.

Una de sus desventajas es que solo puede emplearse sobre atributos categóricos si es que presentan un orden natural, ya que se requiere calcular una distancia entre cada par de elementos. Los *z-values* presentan un rango que va desde 0 hasta un número proporcional a la cantidad de atributos categóricos y los posibles valores de cada uno de ellos. Llegando a situaciones donde estos valores sean muy altos, impactando negativamente en el uso de memoria y tiempo por el gran número de pivotes generados por HashedCubes. Finalmente, al reducirse los datos a una sola dimensión, se pierde mucha información.

3.2. Método basado en Tensores

***Tensor Partition Flow* (TPFlow)** propuesto por Liu et al. [2018], es un sistema que modela los data cubes del conjunto de datos como tensores, lo que permitió la creación de un nuevo algoritmo denominado "piece wise rank-one tensor decomposition" para lograr la partición homogénea de los datos, y luego extraer los patrones presentes en cada una de estas particiones haciendo uso de la información contenido en cada una de las dimensiones.

TPFlow automatiza la búsqueda del "slice" más óptimo así como la detección de los patrones ocultos en los subconjunto de los datos. Además, permite un análisis exploratorio detallado, si es requerido por el analista, debido a que particiona los datos en grupos con similares patrones cruzando información de las diferentes dimensiones. A diferencia de otras técnicas similares, se preocupa por entender las necesidades del usuario.

El algoritmo de descomposición presenta problemas de desempeño cuando el tamaño de los tensores se incrementa. Asimismo, debido a su naturaleza, muchos de sus cálculos tensoriales pueden ser paralelizados usando GPU.

No explota la información de la vecindad de horas y regiones para realizar agrupamientos. Esta técnica para visualizar significativamente los patrones encontrados requiere de más recursos frente a aquellos que obtienen interpretaciones a partir de los valores de entropía.

3.3. Método basado en Redes Neuronales

NeuralCubes, propuesto por Wang et al. [2019], son redes neuronales diseñadas para hacer predicciones del resultado de las consultas de agregación a una base de datos. Donde la entrada es una consulta en forma de rangos de atributos; la salida, el resultado de agregar los datos devueltos por dicha consulta. Este modelo comprende dos etapas, el entrenamiento offline y la consulta en tiempo real. En el primero se requiere de un Modelo de Aplicación, que proporciona información de cuántos atributos son usados y cuál es su respectivo formato; en el segundo, un Modelo del Usuario, que indica los tipos de consultas así como la frecuencia usada por el usuario. Gracias a estos modelos se cons-

truye el conjunto de entrenamiento a través de la generación de pares consulta-respuesta, lo que permite realizar el entrenamiento de la red neuronal. Posteriormente, en la etapa de consulta en tiempo real, se predecirá sus correspondientes resultados para las nuevas consultas realizadas.

NeuralCubes al estar basado en redes neuronales, hereda la capacidad de aprender características de alto nivel, convirtiéndose en su principal valor, ya que permite una mejor exploración visual de los datos. Sin embargo, también hereda algunos de sus problemas como la falta de un método automatizado para determinar la arquitectura adecuada de la red, representando su principal limitación, dada la dependencia de su arquitectura con la complejidad de los datos.

NeuralCubes realiza un consumo de memoria principal, que pueden llegar a unos cuantos KBs dependiendo del tamaño del modelo, durante su ejecución, permitiendo que pueda ejecutarse desde el lado del cliente para procesar las consultas en tiempo real.

Otra limitación presente es la falta de exactitud en los resultados devueltos ya que solo son aproximaciones.

3.4. Métodos sensibles a la semántica

SemTree, propuesto por Amato et al. [2015], es un índice basado en el KD-Tree, en su versión distribuida, para lograr la recuperación semántica de documentos. Aprovechando el significado de las palabras claves contenidas en los documentos para realizar las operaciones de indización. La semántica de los documentos son expresados por tripletas <sujeto, predicado, objeto>, siendo mapeados a un espacio vectorial a través de una distancia semántica que usa vocabularios específicos y/o generales. FastMap es el algoritmo usado para realizar el mapeo, permitiendo definir una estructura de indización eficiente bajo un entorno distribuido.

Este índice está distribuido en diferentes particiones administradas por un solo nodo. Soporta inserciones dinámicas de nuevos puntos así como consultas por rangos y los k-vecinos más cercanos. La inserción empieza su recorrido en el nodo raíz correspondiente a la partición raíz hasta encontrar el nodo hoja, donde el nuevo dato debe ser insertado. Por otro lado cuenta con un algoritmo para la construcción de nuevas particiones cuando se alcanza el número máximos de recursos permitidos, obteniendo algunas particiones que serán usados para el enrutamiento y otras para almacenar datos.

Una de las limitantes del SemTree, es que el árbol debe estar bien balanceado para que se realicen inserciones de forma correcta. Por otro lado, el desempeño del algoritmo para construir las particiones está relacionado al número de máquinas usadas. Finalmente, la complejidad de sus algoritmos de búsqueda son iguales a las del KD-tree.

***Packet-Memory Quadtree* (PMQ)**, propuesto por Toss et al. [2018], es una estructura de datos auto organizado del tipo cache-ajena, es decir, aprovecha el recurso de la caché del CPU sin conocer cuál es su tamaño. La idea principal es mantener una tasa de espacios vacíos en el arreglo de tal manera que pueda realizarse a un bajo costo tanto la inserción como la eliminación, ya que deberá desplazar solo algunos elementos del arreglo para realizar estas operaciones.

En la inserción de los elementos, se realiza un rebalanceo de ciertas partes del arreglo que amerite, teniendo en cuenta conservar la densidad además de esconder los datos más antiguos siempre y cuando, se haya alcanzado el uso máximo de la memoria. En cuanto a la división espacial de los datos, se realiza un ordenamiento en función del índice de Morton, donde aquellos que sean cercanos en un espacio bidimensional tendrán el mismo índice permitiendo que sean almacenados de forma cercana en el PMQ. Además de garantizar la localidad espacial, se define un Quadtree implícito que permitirá realizar consultas espacio temporales.

En la Fig. 3.6 podemos apreciar un PMQ para 9 elementos en un dominio 2D. Los elementos se clasifican en base a su índice Z que define recursivamente una subdivisión implícita de cuatro árboles, el cual nunca es almacenada Toss et al. [2018].

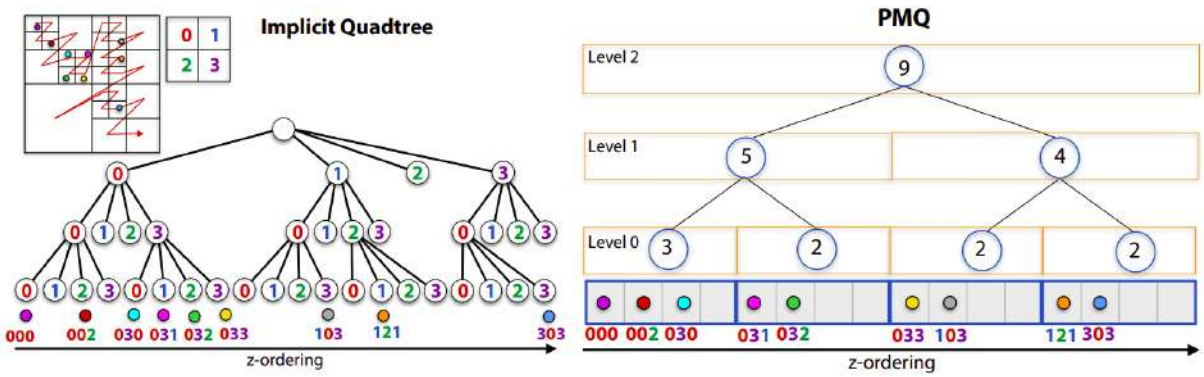


Figura 3.6: Se ilustra el almacenamiento en el PMQ de 9 elementos de índice z que pertenecen a un dominio 2D. Figura extraída de [Toss et al., 2018].

PMQ permite realizar consultas sobre una colección de eventos recientes que carecen de algún tipo de ordenamiento u organización, las cuales serán continuamente actualizadas en tiempo real. En algunas situaciones el tiempo de realizar una inserción podría tomar mucho tiempo cuando es necesario rebalancear completamente la estructura. Asimismo las operaciones se realizan de forma secuencial obligatoriamente, afectando su tiempo de respuesta a las consultas y limitando el número de transacciones a realizarse.

Grafo de Vecindad, frente al problema de los k vecinos más cercanos, en la investigación realizada por Zhou et al. [2013] se plantea una solución aproximada debido al gran volumen y alta dimensionalidad de los datos a procesar. La cual consiste en particionar aleatoriamente el espacio de los datos en subespacios, limitados en tamaño, usando pivotes y diseñar dos capas para la indización distribuida basados en el grafo de vecindad. Estas capas están comprendidas por un grafo de pivotes vecinos y por los subespacios

distribuidos en varias máquinas, respectivamente.

Es importante señalar, que cada nodo en el grafo de vecindad de pivotes, representa un subespacio del conjunto de datos, y que cada punto de dato es asignado al subespacio correspondiente junto a su pivote más cercano. De esta manera, frente a una búsqueda bastará consultar sobre el grafo de vecindad de pivotes por sus N subespacios más cercanos, para luego enviar consultas de los k-vecinos más cercanos a las máquinas correspondientes, donde se encuentran los subespacios retornados previamente. Una vez que todas las respuestas sean recibidas se realizará un ranking de las respuestas, retornando solamente las k más relevantes. En los experimentos realizados, se aprecia un incremento del costo del tiempo originado por el incremento del tamaño de cada subespacio. A medida que se usen más máquinas, el costo se reducirá, presentando un ligero aumento después de cierto punto, que finalmente es aceptable.

3.5. Método acelerado por procesamiento paralelo

Spatio-Temporal Indexing using GPUs **(STIG)**, propuesto por Doraiswamy et al. [2016], es una estrategia de indexación basado en GPU para dar respuesta a consultas espacio temporales de forma interactiva sobre grandes conjuntos de datos históricos. Básicamente, consiste en realizar un filtrado de los atributos espaciales y temporales, realizados al mismo tiempo.

El índice empleado en STIG, denominado Stg-trees. extiende las propiedades del kd-tree, a no solo procesar consultas que puedan ser contenidos en la memoria principal. Stg-trees está compuesto por nodos que se corresponden al KD-tree y una estructura de almacenamiento basado en bloques que contiene la información de un conjunto de registros. Los nodos hojas apuntarán a estos bloques cuya información estará compuesta por los valores de los atributos indexados y un puntero a la ubicación del registro. Bajo este esquema se busca aprovechar eficientemente las GPUs. STIG debido al almacenamiento basado en bloques permite ejecutar consultas fuera de la memoria principal, dejando espacio para almacenar los índices. Asimismo, reduce los costos de transferencia entre CPU y GPU permitiendo eficientes ejecuciones de consultas.

Para aprovechar completamente las capacidades del GPU, los índices deben ser mantenidos una al lado de la otra en la memoria. STIG está limitada a datos que soportan el cálculo de la mediana. Solo funciona sobre máquinas que cuentan con tarjetas Nvidia.

3.6. Análisis Comparativo

En este análisis, que se resume en el cuadro 3.1, se logró identificar que *ConcaveCubes* y *Z-Order Curves+Hashedcubes*, son los únicos que permiten obtener visualizaciones basados en la similitud de los datos. *ConcaveCubes* se caracteriza por su gran simplicidad, ya que a diferencia de los demás métodos realiza agrupamientos con diferentes valores del *threshold* y no utiliza un QuadTree para realizar la organización espacial, sino más bien

la información geo-semántica de los datos (país, estado, región, etc). *Z-order Curves + Hashedcubes*, al basarse en el método propuesto por Pahins et al. [2016] y *Z-Order Curves*, hereda todas sus ventajas pero también sus limitaciones.

Es importante resaltar que el método *HashedCubes* es uno de los métodos con mayores beneficios, siendo caracterizado por su simplicidad y buen desempeño en los tiempos de consulta igualando a *NanoCubes*, pero usando más memoria. Por otro lado, *TopKubes* y *Gaussian Cubes* al ser extensiones de *NanoCubes* heredan sus limitaciones, tales como: no soportar consultas de varias dimensiones espaciales o temporales. *SmartCube* resalta sobre las demás por el dinamismo que ofrece al poder crear, insertar y eliminar cuboides. Sin embargo, al basarse en *Nanocubes*, hereda sus limitaciones tales como no permitir consultas multitemporales, tampoco permite recuperar registros individuales, entre otros. De forma similar, *QDS* puede ser actualizado fácilmente con datos nuevos o recientemente actualizados, al igual que *SemTree*, aunque este último soporta la recuperación semántica de documentos.

PMQ se caracteriza principalmente por trabajar con flujos continuos de datos (*streaming* data), soportando actualizaciones dinámicas. Sin embargo, puede presentar un tiempo de latencia mayor al realizar sus operaciones de rebalanceo sobre toda la estructura. *STIG*, apoyado en su procesamiento paralelo obtiene el mejor rendimiento de tiempo al realizar las consultas. Pero tiene como limitante que los datos deben soportar el cálculo de la mediana. En esta lista, *NeuralCubes* es el único que permite que todas las consultas puedan ser realizadas en el lado del cliente, ya que el tamaño del modelo puede ser lo suficientemente pequeño, sin embargo, sus resultados son solo aproximaciones. *TPFlow* junto con imMens, *HashedCubes* y *QDS* representa el grupo de técnicas que soportan consultas multiespaciales y multitemporales. Por otro lado, junto a *NeuralCubes*, son los métodos que proporcionan información sobre patrones ocultos a los ojos de los analistas. Finalmente, el *Grafo de vecindad*, tiene la peculiaridad de poder responder a consultas de los k-vecinos más cercanos a un punto determinado sobre conjuntos multidimensionales con información asociada a una ubicación geográfico y tiempo.

Tabla 3.1: Cuadro comparativo de los métodos revisados.

Métodos	Multiespacial	Multitemporal	Categ.	Dinámico	Saturac. de llaves	Paralelismo	Recup. de Reg.	Código	*Client-side*	Similitud
imMens	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗
Nanocubes	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
Hashedcubes	✓	✓	✓	✗	✓	✗	✓	✓	✗	✗
Z-O.C. + Hashedcubes	✓	✓	✓	✗	✓	✗	✓	✓	✗	✓
Gaussian Cubes	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗
TopKube	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗
QDS	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗
SmartCube	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
ConcaveCubes	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓
TPFlow	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
NeuralCubes	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗
SemTree	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
PMQ	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗
STIG	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗
Grafo de Vecindad.	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗

Capítulo 4

Una estructura de datos para consultas por similitud sobre datos mixtos

En este capítulo desarrollaremos la propuesta que contempla dos versiones. La primera, se basa en *Hashedcubes*, donde tras un cálculo de la matriz de distancias, seguida de una proyección en 1D, se busca indizar como un nivel adicional dentro de la propuesta de Pahins et al. [2016], esta estrategia es inspirado por el trabajo de Peralta-Aranibar et al. [2019a]. Sin embargo, tras identificar una gran pérdida de información al proyectar los datos, se propone la segunda versión, convirtiéndose en la propuesta final. Para ello, conservamos el cálculo de la matriz de distancias, para luego construir un árbol de similitud y conectarlo estratégicamente con una estructura espacial. A continuación, detallaremos cada una de las versiones en las siguientes secciones.

4.1. Primera Versión - Basado en Hashedcubes

Proponemos una nueva estructura de datos basado en *Hashedcubes* que permitirá realizar consultas, cumpliendo los criterios de espacio, tiempo y similitud en el mismo contexto. La idea principal es transformar los elementos de un espacio N-dimensional a un espacio de una dimensión, conservando la vecindad entre los datos, para luego ser indizadas en la estructura bajo el siguiente orden: espacial y similitud.

Por otro lado, crearemos una representación visual que nos permita explorar todos los datos de manera resumida e interactiva, y realizar consultas interactivamente usando recursos visuales que introduciremos. Adicionalmente, construiremos un prototipo en entorno web que nos permita implementar nuestra metodología.

Siendo Hashedcubes, la estructura base de la propuesta por todos los beneficios que fueron resaltados en la Sección 3.6, pasaremos a explicarlo con más detalle en la siguiente sección.

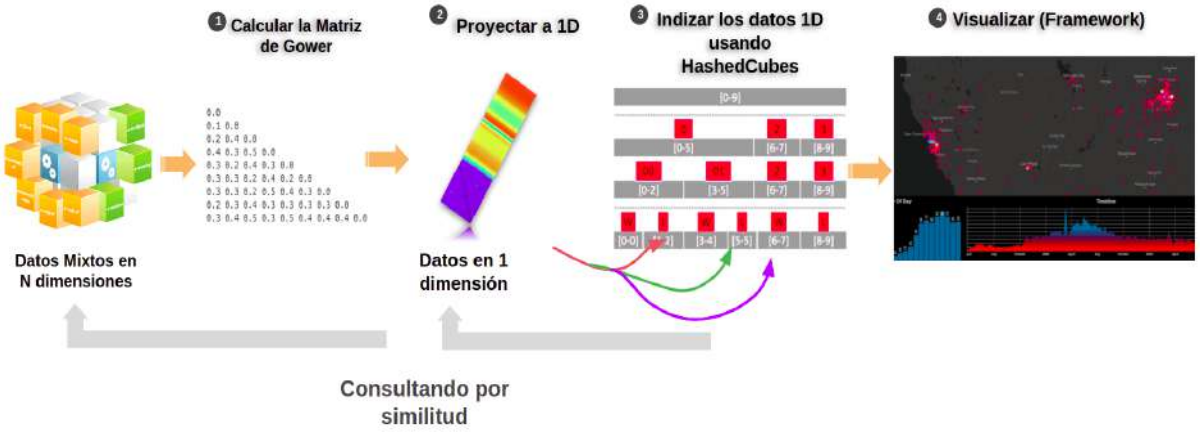


Figura 4.1: Una vista panorámica del flujo que se llevará a cabo en la primera versión de nuestra propuesta. Figura creada por el autor.

4.1.1. Partiendo desde Hashedcubes

La construcción de los Hashedcubes requiere que las dimensiones de los datos sigan un orden, pudiendo ser primero espacial, luego categórico y temporal. Tras ello, debemos asociar un arreglo, que llamaremos *Hash*, con un pivote raíz $[0, n-1]$, que representará la partición inicial de los n elementos del universo. Cada elemento del arreglo es un entero que apunta a un registro en el conjunto de datos. Para cada una de las dimensiones del esquema de indexación, se indexarán las particiones de los objetos utilizando pivotes. Estas particiones, serán denominadas en adelante como *bin*, presentan diferentes interpretaciones para cada dimensión. Representando regiones para una dimensión espacial; valores o rangos específicos, dimensión categórica; o intervalos de tiempo, dimensión temporal.

Teniendo como entrada un arreglo de n elementos, todos sus elementos pertenecen a la misma ubicación, siendo representado por un pivote $[i_0, i_1]$. Cada dimensión recibe como *input* una lista de pivotes y como *output* una lista de pivotes. Por ello, la primera dimensión recibe como entrada el pivote de la raíz. Luego, los pivotes que se vayan creando serán recibidos por las dimensiones subsiguientes. Después de procesar cada dimensión se genera una nueva lista de pivotes, creaándose una jerarquía de pivotes qe permiten conectar los *bins* creados en cada dimensión. Dado que un bin en una dimensión dada es un subconjunto de un bin de la dimensión anterior, una lista de pivotes consigue representar subconjuntos para todas las dimensiones que fueron definidas previamente. Esto es conveniente para una buena administración de la memoria, ya que permite eliminar algunas dimensiones de la representación.

La jerarquía de pivotes está inspirada en la jerarquía de un árbol, debido a que cada pivote representa un conjunto que puede dividirse a su vez en otros subconjuntos. Los pivotes hermanos se almacenan como listas, ya que cada dimensión almacena colecciones de pivotes. Las dimensiones pueden ser tratadas independientemente unas de otras. Esto permite cierta fliexbilidad al momento de ejecutar consultas ya que que saltarse dimensiones que no se solicitan. Además, la cardinalidad de un subconjunto representado por un pivote se puede obtener directamente de los índices del pivote. En la siguiente Fig. 4.2 observaremos los aspectos contemplados en Hashedcubes. Tenemos como entrada 10 puntos

de datos usando el esquema [[Latitud, Longuitud], [Dispositivo], [Tiempo]]. En la Figura 4.2b paso 1, el arreglo se reordena a lo largo del primer nivel del cuadrante y se crean tres particiones asociadas a los cuadrantes 0, 2 y 3 , que vendrían a ser los cuadrantes que contienen puntos. Luego, se crean tres pivotes [0-5], [6-7], [8-9] para delimitar estas particiones. En el paso 2 se reordena el arreglo a lo largo del segundo nivel del cuadrante. Es importante señalar que en este paso solo se está subdividiendo el primer cuadrante del QuadTree. Por lo tanto se actualiza la partición afectada, es decir la que está asociada al pivote [0..5]. Ello ocasiona que se creen dos nuevos pivotes ([0..2] y [3..5]). Lo mismo se realiza en los pasos 3 y 4, pero usando las dimensiones categóricas y temporales para crear más particiones. En la Figura 4.2c comparamos los valores de entrada del arreglo con el reordenamiento final obtenido después de las sucesivas particiones de los datos. En Hashedcubes solo es necesario mantener el arreglo final a lo largo de los pivotes creados en cada paso para recuperar las particiones que se originen. Finalmente se muestra la lista de pivotes creados en cada paso y las almacenados por Hashedcubes. Cada lista de pivotes se corresponde con las particiones inducidas en el primer y segundo nivel del QuadTree, y la partición categórica, en este caso si el dispositivo utilizado era Windows (W) o Linux (L) [Pahins et al., 2016].

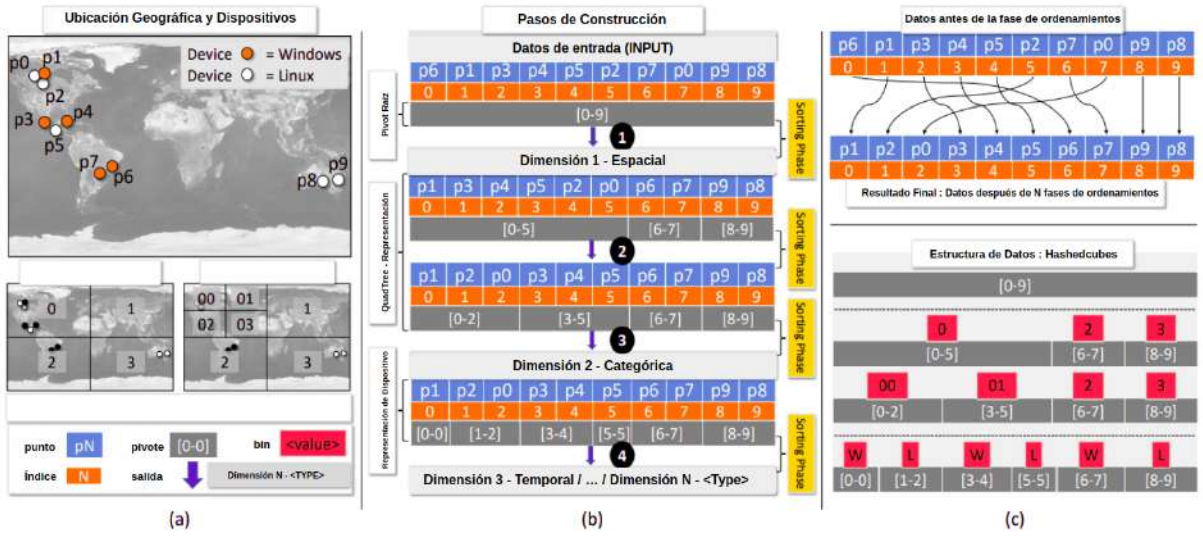


Figura 4.2: Vista general de la construcción de Hashedcubes. En (a) se presenta un pequeño conjunto de datos $[p_0, ..., p_9]$ bajo un esquema espacio-categórico-temporal. El proceso se completo se observa en (b) donde se muestra el paso a paso de la construcción de las particiones ordenadas. Finalmente en (c) Los datos se ingresan en cualquier orden a un arreglo secuencial y cada registro se asocia con un índice, representado por un rectángulo anaranajado. Figura extraída de [Pahins et al., 2016].

4.1.2. Calcular la Matriz de Gower

Como paso previo a nuestra proyección utilizaremos el cálculo de la matriz de distancias usando la métrica de Gower, el cual no requiere realizar transformaciones sobre los datos. Es decir, calcula de forma directa sobre cada par de registros, aplicando una función de agregación simple para cada una de sus dimensiones. Tiene un mejor rendimiento y

precisión que las distancias de jerarquías y de Goodall sobre datos mixtos, [Muñoz Salas, 2019].

4.1.3. Proyectar a 1D

Para realizar la proyección a un espacio unidimensional consideramos las siguientes ténicas del estado del Arte:

- UMAP, método reciente para la redución de dimensiones, caracterizado por su bajo costo en tiempo de ejecución y preservar la estructura global de los datos. Además, no presenta restricciones computacionales en cuanto al número de dimensiones del nuevo espacio que se desea originar [McInnes et al., 2020]. Sin embargo aún no se tienen evidencia de buenos resultados sobre datos mixtos.
- t-SNE, es uno de los métodos de proyección ampliamente usado en la última década, presentando buenos resultados de visualización de datos multidimensionales, la cuales son proyectados a un nuevo espacio 2D o 3D. Muñoz Salas [2019] realizó experimentos sobre datos mixtos, obteniendo buenos resultados al combinarlo con la matriz de distancias de Gower.
- PCA, es un método tradicional, caracterizado por reducir el espacio original a los primeros k componentes principales, es decir, no está limitado solo a dos o tres como los métodos previos [Wold et al., 1987].

Los resultados obtenidos en la Fig. 5.2, evidencian una gran pérdida de información al realizar todas estas proyecciones. Que afectaría fuertemente a nuestra interacción con la herramienta, ya que los resultados retornados para nuestras *query*s serían imprecisos, conllevando a realizar toma de decisiones incorrectas.

En vías de trabajar sobre el espacio original, se plantea otro abordaje que hace uso de toda la información disponible. Para ello, se construye un árbol de similitud a partir de nuestros conjuntos de datos, donde cada uno de sus nodos estará conectado estratégicamente con los nodos de un QuadTree 2D. Todo este nuevo proceso se describirá con mayor detalle en la siguiente sección.

4.2. Segunda Versión - Arból de Similitud 2D

El presente abordaje, cuyo flujo de trabajo se muestra en la Fig. 4.5, se compone de un árbol de Similitud y un QuadTree, conectados estratégicamente para soportar consultas de similitud eficientemente. Por la presencia de la estructura espacial logramos preservar el principio de realizar cálculos solo sobre los datos que pertenecen a la región espacial que el usuario está visualizando [Liu et al., 2013].

Dado que el DBSCAN es utilizado por Li et al. [2018], aprovecharemos este método para realizar diversos agrupamientos sobre nuestra matriz de Gower precalculado, a

través de varias iteraciones con diferentes valores del radio (*epsilon*). Donde para cada valor del ϵ obtendremos una colección de grupos que llamaremos *Cluster Nodos*, las cuales son indizados en cada nivel de nuestro Árbol de Similitud. Tras obtener esta representación jerárquica de similitud, construiremos nuestro árbol del Quadtree basado en una profundidad predefinida. La forma de enlazar ambas estructuras, se realiza almacenando en cada nodo del QuadTree, punteros a los *Cluster Nodos* pertenecientes a un mismo nivel del Árbol de Similitud. Es importante señalar que realizaremos un cálculo para hallar el *BestLevel*, que contendrá las agrupaciones compactas de todo el árbol.

El *bestLevel*, representado en la Fig. 4.3, es el nivel del árbol de similitud que tiene nodos de agrupamientos muy compactos y dispersos entre sí, correspondientes a cada nivel del QuadTree. Esta sincronización se logra porque los nodos para un nivel dado, por ejemplo: dos (Q:2) en el QuadTree, solo apuntarán a nodos de un nivel específico en el árbol de similitud, que vendría a ser el *bestLevel*. Con esto aseguramos que dos nodos hermanos del QuadTree no apunten a diferentes niveles en el árbol de similitud, evitando así inconsistencias en los agrupamientos. El cálculo del *bestLevel* consiste inicialmente en calcular la densidad que tiene un *Cluster Nodo* para cada nodo del QuadTree, es decir, el cociente entre la cantidad de puntos del nodo de similitud que pertenecen a la región espacial del *quadnodo*, dividido entre el tamaño del nodo de similitud. Luego, calculamos el promedio de estas densidades para cada uno de los niveles del árbol de similitud, con respecto a un nivel del QuadTree (Q:2). Por lo tanto, el nivel que tenga el mayor promedio, es el nivel de similitud o *bestLevel* asignado a dicho nivel del QuadTree.

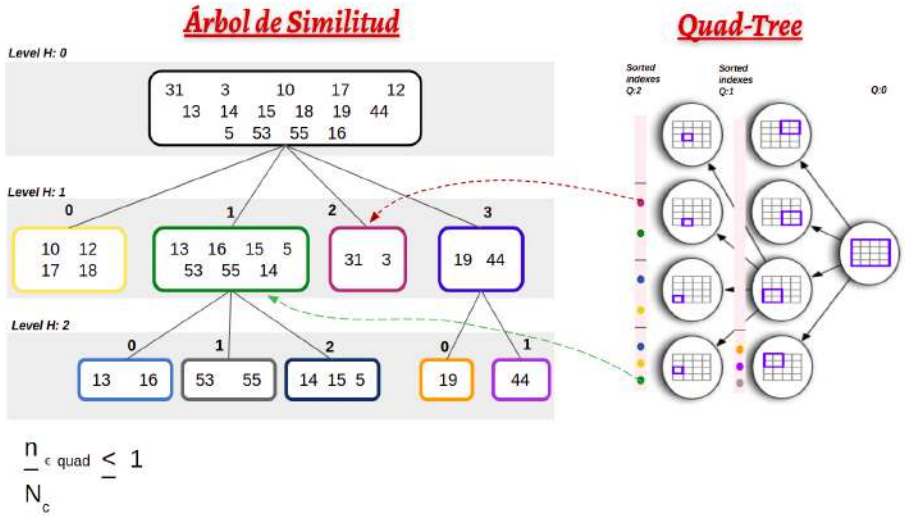


Figura 4.3: Representación del *bestLevel*. Figura creada por el autor.

A diferencia de la técnica del ConcaveCube propuesto por Li et al. [2018], que también realiza agrupaciones, nuestra propuesta soporta datos mixtos, asímismo soporta datos multidimensionales gracias al procesamiento que realizamos con la matriz de gower, ya que solo depende de la cantidad más no de la dimensionalidad. Por otro lado, el usuario será capazar de obtener información detallada tanto a nivel espacial como de similitud. Una desventaja de nuestra estructura, se encuentra en el almacenamiento de la matriz de gower, el cual puede llegar a requerir una cantidad considerable de memoria. Pero que gracias al abaratamiento de harware que cada vez es más exponencial, podría no afectar tanto.

Nuestra herramienta mostrada en la Fig. 4.4 presenta dos vistas, es decir, una para los agrupamientos (i) y otra para la similitud (ii). La primera se establece por defecto, mientras que para cambiar de vista, pulsamos sobre el componente selector indicado por **(a)**. Nuestro componente central es el mapa de la librería *leaflet* indicado por **(m)**, el cual carga todos nuestros datos agregados en *bounding boxes* coloreados con el color asignado al *cluster* que pertenecen. En los componentes **(b)** y **(c)** tenemos dos *sliders*, el primero para recorrer los niveles del árbol de similitud, y el segundo para los del QuadTree, respectivamente. En **(d)** tenemos una tabla de centroides que contiene el dato (registro) más relevante de su respectivo *cluster* y que se actualiza igual que **(m)**, cada vez que interactuamos con el mapa (*zoom in/out*, arrastrar o soltar) y en cada consulta. Donde además de visualizar todos los atributos originales, se observa una figura rectangular coloreada. Su color es único solo si nos encontramos en la vista de agrupamientos. Asimismo, contiene una barra horizontal que contiene un botón para mostrar y ocultar **(s)** la tabla de centroides **(d)**. Cuando nos situemos en la vista de similitud (ii), luego de haber seleccionado la región de interés cuyos *bounding box* servirán de consulta en nuestra estructura, aparecerá una leyenda de similitud **(e)**. El cual presenta una escala de colores que va del amarillo intenso hacia el más oscuro. Asimismo, están compañados de un valor entero que indica la distancia entre los agrupamientos, donde a menor distancia, mayor similitud, y por ende, datos más similares.

i) Vista de agrupamiento ii) Vista de similitud.

Figura 4.4: En esta gráfica presentamos las dos vistas que contempla nuestra herramienta, así como los diferentes componentes visuales que ofrece al usuario: (a) selector para la consulta de similitud, (b) *slider* del árbol de similitud, (c) *slider* del árbol espacial QuadTree, (d) tabla de centroides, (e) leyenda de similitud, (s) botón para mostrar/ocultar la tabla de centroides y finalmente (m) el mapa interactivo disponible en la librería *leaflet.js*. Indicar que estos componentes en su mayoría fueron implementados usando javascript, mientras que para la leyenda se utilizó *d3.js*. Figura creada por el autor.

Figura 4.5: Flujo de Trabajo para la construcción de la estructura de datos que permite realizar consultas por similitud de forma interactiva. Conformado por un árbol de similitud el cual está enlazado estratégicamente con una estrucutura espacial 2D, llamado QuadTree. Figura creada por el autor.

Capítulo 5

Pruebas y Resultados

En esta sección presentaremos diversos experimentos realizados para las dos versiones de nuestra propuesta. En cuanto a la primera, mostraremos los tiempos obtenidos para el cálculo de la matriz de gower y las proyecciones multidimensionales, en la Tabla 5.2 y el gráfico de barras de la Fig. 5.1. Seguido de la Figura 5.2 acerca de los resultados de la métrica de preservación de vecindad. Mientras que para nuestra versión final, se llevaron a cabo varios casos de estudios sobre grandes conjuntos de datos, con la finalidad de demostrar su aplicabilidad en cualquier escenario real.

5.1. Primera versión - Basado en Hashedcubes

Para esta versión, se utilizaron doce conjuntos de datos de naturaleza mixta, que previamente fueron procesados, eliminando los registros que presentan campos nulos o vacíos, y quitando aquellos atributos que no aportaban información relevante o resultaban redudante con otros, por fines de simplicidad. En las siguientes sub secciones describiremos los resultados obtenidos sobre los siguientes conjuntos de datos, que se encuentran listados en la Tabla 5.1.

5.1.1. Cálculo de la matriz de Gower

En la siguiente Figura 5.1, podemos apreciar que los mayores tiempos de ejecución se obtuvo cuando el número de registros del conjunto de datos fue mayor en comparación a los demás, teniendo en cuenta la cantidad de atributos que presentaron. En este sentido, *DT10* quien presenta la mayor cantidad de registros junto con *DT5* y *DT3* , obtienen los mayores tiempos de ejecución, incluso así alcanza solo unos cuantos segundos. Mientras que *DT8* junto con *DT9* obtienen los mejores tiempos. Un hecho interesante sucede con *DT4* que presenta un menor tiempo que *DT2* y *DT7*, a pesar que estos poseen un menor número de registros, *DT4* posee solo un atributo númerico. Otro evento interesante sucede con *DT1* y *DT3* que presentan la misma cantidad de registros, sin embargo *DT1* obtiene un menor tiempo debido a que *DT3* presenta un atributo categórico adicional.

Conjunto de Datos		Original		Proc. Simple	
		Reg.	**Atribut.**	**Reg.**	**Atribut.**
DT1	Australian Credits [Bache, 2013]	690	14	690	14
DT2	Automobile [Bache, 2013]	205	25	***159***	25
DT3	Credit approval [Bache, 2013]	690	15	690	15
DT4	Dermatology [Bache, 2013]	366	34	***358***	34
DT5	German Credits [Bache, 2013]	1 000	20	1 000	20
DT6	Heart [Bache, 2013]	270	13	270	13
DT7	Hepatitis [Bache, 2013]	155	19	***80***	19
DT8	Lymphography [Bache, 2013]	148	18	148	18
DT9	Marathon [Bache, 2013]	88	9	***81***	***5***
DT10	Shooting [Bache, 2013]	528 234	17	**9 765**	17
DT11	Crashes1 [Bache, 2013]	73 486	63	**10 802**	63
DT12	Crashes2 [Bache, 2013]	73 486	63	**14 443**	63

Tabla 5.1: Descripción de los conjuntos de datos utilizados en esta investigación. Cada uno de ellos está asociado a un ID, que permitirá identificarlos en el presente documento. Por otro lado, se indica la cantidad original de registros y atributos, así como las cantidades obtenida tras un simple preprocesamiento.

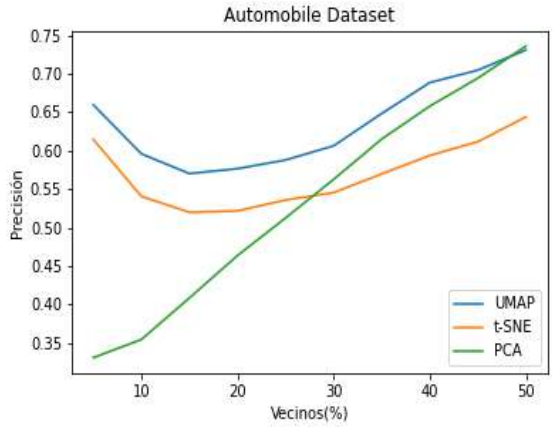
Figura 5.1: Resultados del tiempo(ms) que Gower tarda en calcular la matriz de distancias para nuestros conjuntos de datos mixtos.

A continuación presentaremos la Tabla 5.2 con los tiempos obtenidos al calcular la matriz de distancias de Gower, al proyectar los datos a nuevos espacios, tanto unidimensional como bidimensional a través de UMAP, t-SNE y PCA. Los mejores tiempos son obtenidos por PCA, sin embargo, es conocido de esta técnica su falta de preservar la vecindad de los datos en el nuevo espacio con respecto al original. Al proyector los datos en el espacio 1D, UMAP tarda tanto como t-SNE e incluso más. Frente a ello se optó por medir los tiempos cuando se proyecta a un espacio 2D. En esta ocasión los resultados fueron los esperados, en su mayoría UMAP, registró un mejor tiempo que t-SNE.

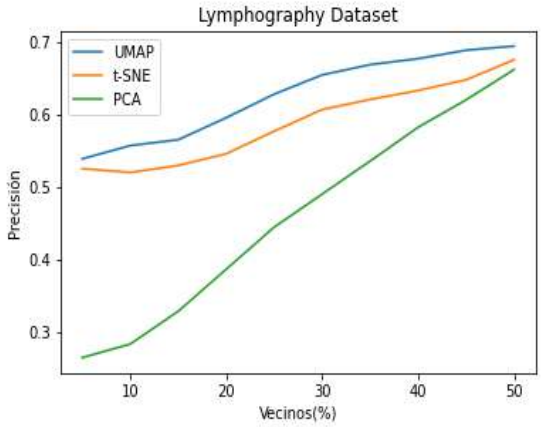
Para realizar la proyección a 1D, UMAP podría no ser la mejor opción en cuanto a tiempo. Sin embargo, no es el único criterio a tomar en cuenta. Por consiguiente, es necesario medir la preservación de la vecindad. Esta métrica indica qué porcentaje de los k-vecinos más cercanos con respecto a un punto, se siguen manteniendo en el nuevo espacio reducido.

5.1.2. Proyectando a 1D:

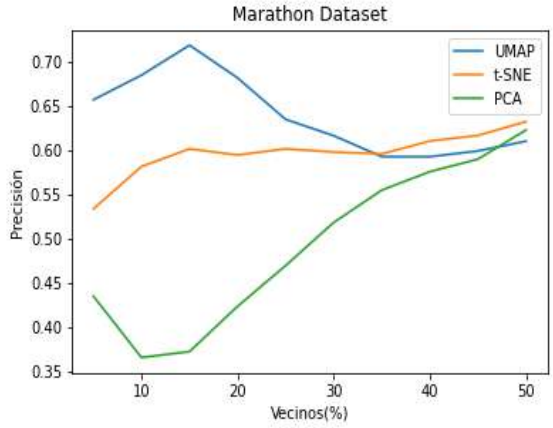
En la Figura 5.2, observamos diferentes gráficas con la precisión obtenida (eje "Y") por UMAP, t-SNE y PCA para varias cantidades de vecinos más cercanos (eje "X"). Se ha dispuesto colocar en la parte superior de la grilla, casos donde UMAP obtiene los mejores resultados, mientras que en la parte media e inferior, aquellos donde se obtuvieron resultados no tan buenos. Claramente se observa en las Figuras 5.2(a) y 5.2(b) que UMAP obtiene el mejor desempeño sobre t-SNE y PCA en todos los porcentajes de vecinos más cercanos. En 5.2(c) se sigue manteniendo esta superioridad pero solo hasta el 35 % de vecinos, siendo superado por t-SNE. En todos estos casos, el número de registros no superó la cifra de doscientos. Por otro lado, en los casos 5.2(eh), se observan las curvas de UMAP y t-SNE muy cercanas, lo que sugiere el uso de cualquiera de ellos para obtener buenos resultados. Finalmente, para un número alto de registros como en el caso DT11 y DT12, t-SNE obtiene la mejor precisión para un gran porcentaje de vecinos cercanos que llega hasta los 40 %.



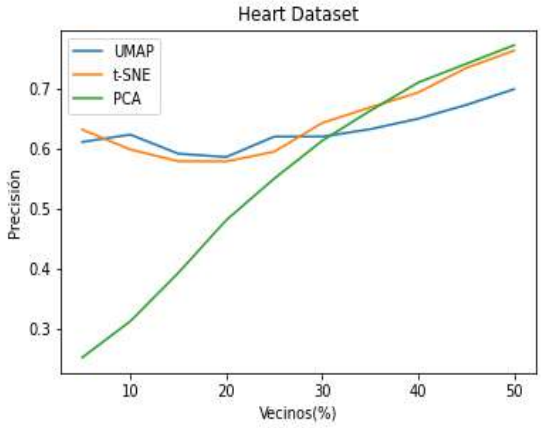
(a) DT2



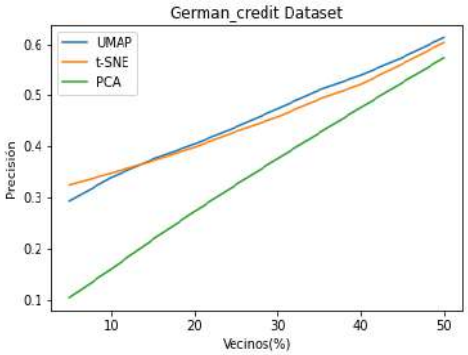
(b) DT8



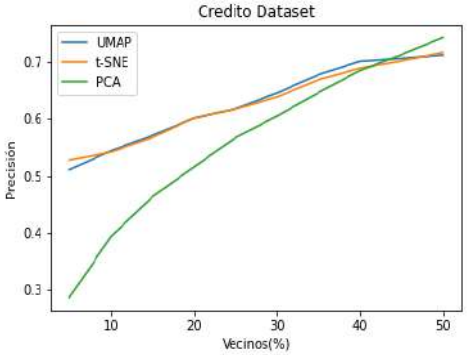
(c) DT9



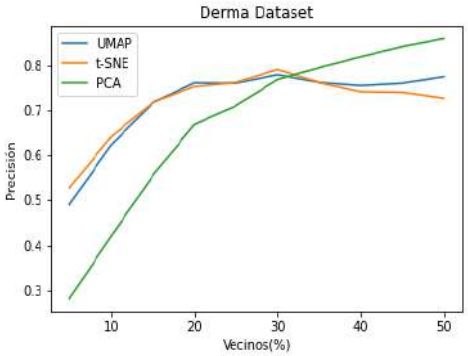
(d) DT6



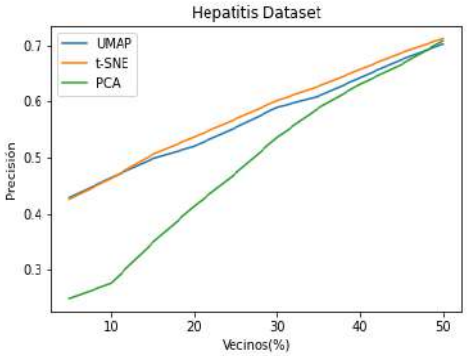
(e) DT5



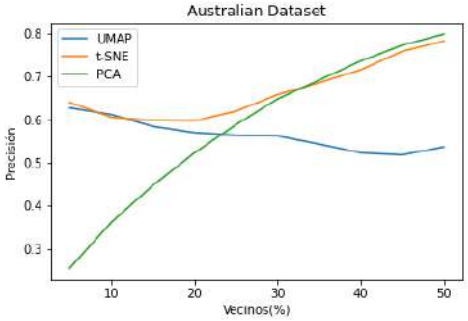
(f) DT3



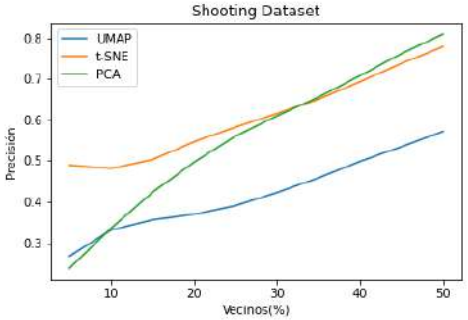
(g) DT4



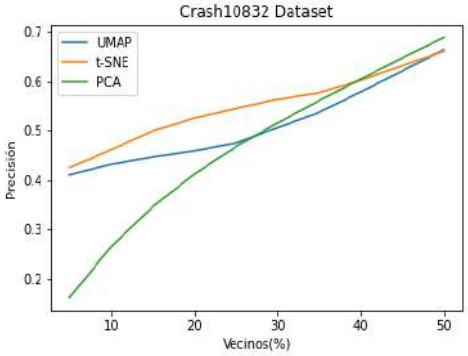
(h) DT7



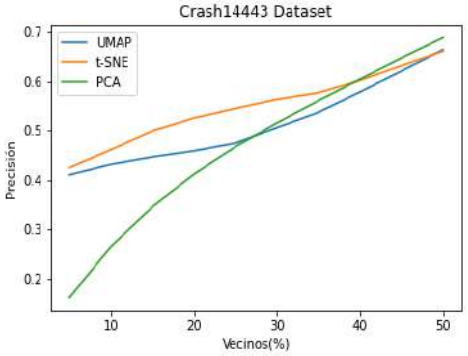
(i) DT1



(j) DT10



(k) DT11



(l) DT12

Figura 5.2: Medición de la preservación de vecindad de las proyecciones a un espacio 1D realizadas por *UMAP* (—), *t-SNE* (—) y *PCA* (—) sobre los conjuntos de datos descritos en la Tabla 5.1. Figuras creadas por el autor.

Datasets	TIEMPO						
	Gower (ms)	UMAP (s)		t-SNE (s)		PCA (s)	
		1D	2D	1D	2D	1D	2D
DT1	**75.6 ms**	2.37 s	2.45 s	2.450 s	3.930 s	0.0628 s	0.0466 s
DT2	**63.3 ms**	0.97 s	1.01 s	0.546 s	0.869 s	0.0243 s	0.0292 s
DT3	**527 ms**	2.57 s	2.61 s	2.540 s	4.220 s	0.0575 s	0.0392 s
DT4	**40.4 ms**	1.60 s	1.74 s	1.360 s	1.840 s	0.0466 s	0.0599 s
DT5	**1 350 ms**	3.44 s	3.47 s	3.940 s	7.460 s	0.0715 s	0.0847 s
DT6	**35.1 ms**	1.49 s	1.51 s	0.891 s	1.450 s	0.0475 s	0.0482 s
DT7	**61.2 ms**	1.18 s	1.20 s	0.573 s	0.709 s	0.0253 s	0.0264 s
DT8	**18.9 ms**	0.98 s	1.18 s	0.563 s	0.644 s	0.0162 s	0.0231 s
DT9	**20.9 ms**	0.99 s	0.98 s	0.343 s	0.426 s	0.0147 s	0.0138 s
DT10	**1 min 43 s**	37.90 s	34.10 s	47.4 s	90.0 s	4.29 s	3.65 s
DT11	**8 min 44 s**	1 min 6 s	1 min 7s	1 min 54 s	3 min 32 s	17.2 s	21.2 s
DT12	**5min 14s**	2min 23s	2min 6s	2min 39s	4min 47s	33.2 s	27.4 s

Tabla 5.2: Mediciones del tiempo de ejecución para realizar el cálculo de la matriz de Gower, realizar la proyección a un nuevo espacio de 1D y 2D, usando UMAP, t-SNE y PCA.

5.2. Segunda versión - Basado en el Árbol de Similitud y QuadTree 2D

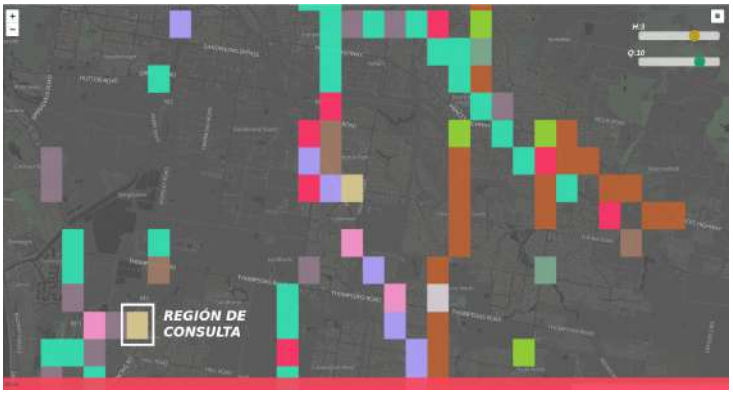
En vías de de demostrar el correcto funcionamiento y aplicabilidad de nuestra propuesta en un escenario real, presentaremos diversos estudios y análisis. Por ello, se utilizarán diferentes conjuntos de datos, lo que implica diferentes tamaños tanto en volumen como en cantidad de dimensiones o atributos. Para cada una de ellas se presentarán los resultados obtenidos de los diferentes agrupamientos y de las consultas por similitud llevadas a cabo. Como primer escenario tenemos accidentes vehículares en la ciudad de Victoria en Australia, luego aquellos ocurridos en California, USA. Finalmente, venta de viviendas en la Costa Este Norte de USA.

5.2.1. Caso de Estudio: Accidentes de tránsito en Victoria

En este caso de estudio usaremos el siguiente *dataset* que contiene información de los accidentes de tránsito en el estado de Victoria, Australia. Conteniendo 51867 registros y 65 atributos, que van desde el 2012 hasta el 2017. Como parte del preprocesamiento de los datos, se eliminaron registros cuyos atributos eran nulas, y se tomaron en cuenta 14 de ellos para nuestro análisis, de los cuales 7 son categóricos, tales como: ACCIDENT_STATUS, ALCOHOLTIME, ACCIDENT_TYPE; y los 7 restantes son numéricos: SPEED_ZONE, FATALITY, SERIOUSINJURY, OTHERINJURY, etc. El tiempo para calcular la matriz de gower fue de 4 segundos, mientras que la indización en nuestra propuesta fue de 1000 segundos, alrededor de 16 minutos.

En la Fig. 5.3 (a-b), podemos observar un total de 37 agrupamientos, con sus respectivos registros relevantes contenidos en la tabla de centroides. Estos *clusters* están representados por un color distintivo para la interacción, donde el nivel en el árbol de similitud es igual a tres (H: 3), y en el Quadtree nueve (Q: 10). En la Fig. 5.3 (a) realizamos una consulta sobre una una región en la avenida Frankston - Dandenong Rd, ubicada en el suburbio de Carrum Downs. Observamos los resultados en la Fig. 5.3 (b), que existe una alta similitud con el cruce de Lynbrook BLVD y S. Gippsland Highway. Los accidentes que se registran en ambos lugares se caracterizan principalmente por la presencia de alcohol y la falta de atención policial. Mientras que es menos similar que los accidentes ocurridos en la Hallam Bypass Trail, representados con el color anaranjado, donde no se registró presncial de alcohol, pero si atención policial, además de presentar la misma severidad. Por otro lado, son aún menos similares con aquellos accidentes del cruce de las avenidas Thompsons Road y Narre Warren RD, donde presentan otro nivel de severidad, pero el mismo valor de "C" para el atributo SRNS_ALL, y la misma cantidad de mujeres heridas.

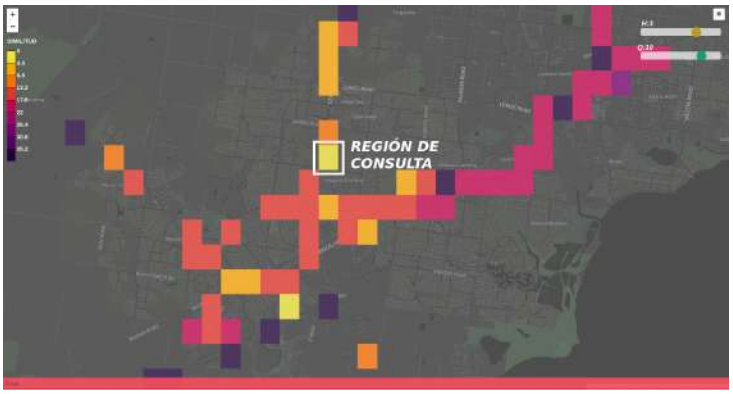
En la Fig. 5.3 (c-d), presentamos los resultados de nuestra segunda consulta, en la intersección de las avenidas Hogans Road y Derrimut Road. En la Fig. 5.3 (c) se puede identificar 16 *clusters* encontrados para el actual espacio visual del mapa. Por otro lado, observamos la gran similitud de nuesta región de consulta con los accidentes ocurridos en Duncan Roads, mostrados en la Fig. 5.3 (d), donde se caracterizan por ser accidentes del tipo Colisión con otro vehículo, de haber recibido atención policial, de no presentar heridos, en una zona donde la velocidad permitida es de 60 km/h. Mientras que son menos similares con las regiones de color anaranjado donde los accidentes no son del tipo colisión con otros vehículos, sino más bien por volcadura, o colisiones con objetos fijos. Asimismo se presentan intervención policial y presencia de alcohol.



a) Primera visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:10

b) Resultado de la primera consulta por similitud.

c) Segunda visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:10



d) Resultado de la segunda consulta por similitud.

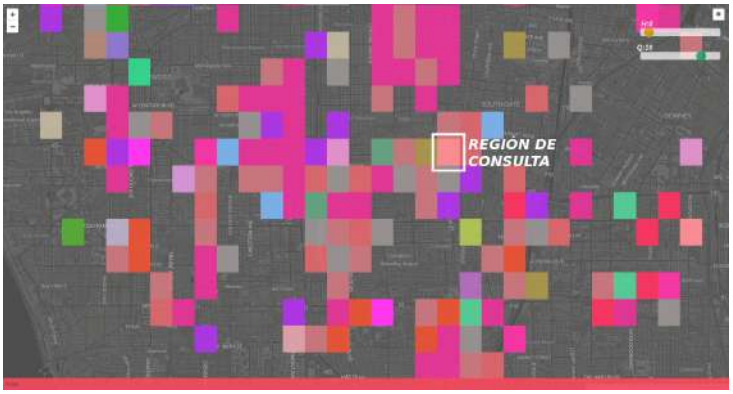
Figura 5.3: En esta gráfica mostramos los resultados para dos consultas. Figuras creadas por el autor.

5.2.2. Caso de Estudio: Accidentes vehiculares en California - USA 2019

Para este análisis hemos tomado todos los datos correspondientes del año 2019 para el estado de California del conjunto original de accidentes de vehículos en USA. El cual contenía alrededor de 2 millones de registros, que son captados por diversas instituciones, por un lado los departamentos de transporte, los diferentes estados, las fuerzas de seguridad, las cámaras de tráfico y los sensores de tráfico de las redes de carreteras [Moosavi et al., 2019a], [Moosavi et al., 2019b]. Tras realizar una limpieza, que consistió en eliminar los registros con campos nulos, se obtuvo 25329 registros, y de un total de 35 atributos, se extrayeron 28 de las cuales 20 son categóricas y 8 numéricas. El tiempo que se tómo para calcular la matriz de Gower fue de 3 segundos, mientras que la indización 942 segundos, alrededor de 15 minutos.

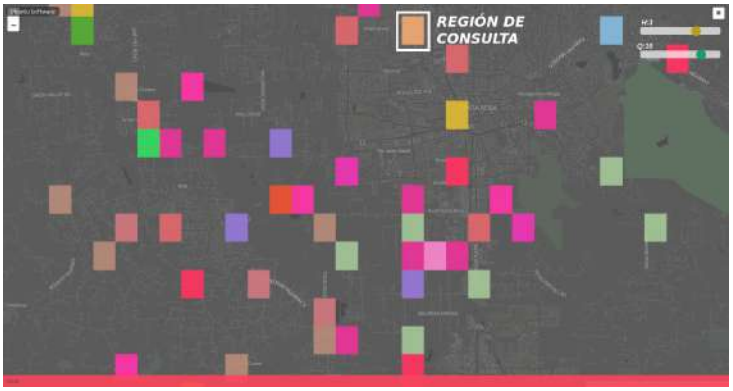
En la Fig. 5.4 conseguimos visualizaciones tanto para la exploración Fig. 5.4 (a) como el resultado de la consulta por similitud en Fig. 5.4 (b). En el primero, se consiguió obtener alrededor de 34 agrupamientos, representadas por un único color. En el segundo, se muestra los resultados, luego de trazar la región de consulta, enmarcada con un borde blanco, sobre la intersección de la Alameda Street, el Imperial Highway y Fernwood Avenue. El cual nos mostró basado en el color intenso de amarillo, que era similar a otros lugares ubicados como Crenshaw BLDV, así como en el cruce de Aviation BLDV con W. Century BLDV. Las cuales se asemejan principalmente porque las condiciones ambientales estuvieron parcialmente nublados, teniendo una visibilidad de los vehículos de 10 mi, sin presencia de precipationes. Luego, tenemos zonas cuyos colores son de color rojo, siendo menos similares debido a que presentan clima moderado, adecuado para actividades al aire libre. temperatura, sol. Sin embargo, presentan la misma visibilidad de 10 mi, no presentan precipationes, compartiendo la misma presión alrededor de 29 in. Finalmente, están las regiones coloreadas de morado oscuro que son las menos similares, tales como el cruce de Manhattan Beach BLVD con Van Ness Avenue, la intersección de Central Avenue con W. Compton BLVD.

En la Fig. 5.4 (c-d) realizamos otra exploración hacia el norte de California, observando en la Fig. 5.4 (c) un total de 28 agrupamientos con sus distintivos colores, de donde eligiremos como región de consulta, la intersección entre Piner Road y Coffey Lane. A partir del cual, se observa los resultados de la consulta por similitud en la Fig. 5.4 (d), muy similares a los accidentes ocurridos en S. Wright RD., Llano Road, Petalum Hill RD, caracterizándose por presentar una visibilidad de 10 mi, con una presión del viento de 29 in, con una dirección del viento hacia el Sur y Suroeste teniendo como velocidad de 9 mph. Por otro lado, es menos similar que las regiones coloreadas de rojo, que presentan una humedad del 38 %, y un clima despejado. Mientras que las zonas de color morado, son las más lejanas en similitud ya que contemplan una humedad del 50 %.



a) Primera visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:10

b) Resultado de la primera consulta por similitud.



c) Segunda visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:10

d) Resultado de la segunda consulta por similitud.

Figura 5.4: Esta gráficas fueron generados por el autor. El conjunto de datos fue extraído de un conjunto de todos los accidentes registrados en USA durante los últimos años.

5.2.3. Caso de Estudio: Venta de Viviendas en la Costa Este Norte de USA

Para estos experimentos se empleó los datos de venta de viviendas de Estados Unidos [Craigslist, 2019]. Como primer paso se realizó una limpieza, elimiando todos los registros con campos nulos. Posteriormente, eliminamos las columnas cuya información era redundante, es decir, de 16 atributos nos quedamos con 13, de los cuales 9 son categóricos y 4 numéricos. Tras ello, filtramos los datos basado en los estados, empleando solo aquellos que se encontraban en la costa este norte (12) tales como: New York, Massachusetts, Maryland, Rhode Island, Connecticut, entre otros. El tiempo que se tómo para calcular la matriz de Gower fue de 3.5 segundos, mientras que la indización 1400 segundos, alrededor de 23 minutos.

En la Fig. 5.5 (a-b) presentamos los resultados tanto de la exploración como de la consulta por similitud. En la Fig. 5.5 (a) podemos apreciar que para el nivel en el árbol de similitud $H : 3$, y $Q : 9$, se visualizan alrededor de 29 *clusters* en nuestra ventana de visualización. La mayor cantidad de datos se encuentra en el litoral, las cuales presentan un mayor precio que aquellas ubicadas en zonas más alejadas.

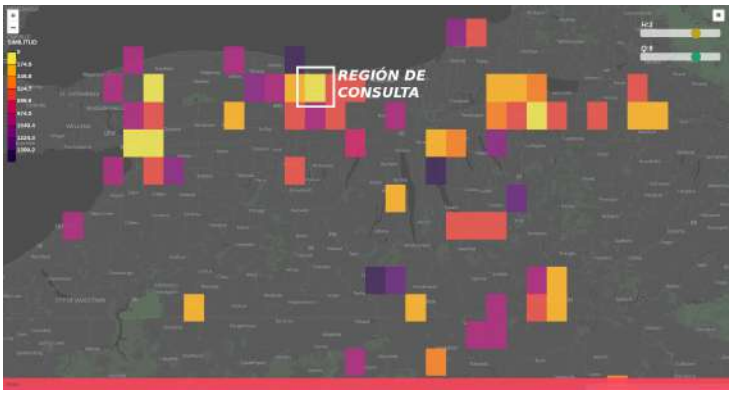
Tras realizar la consulta sobre Lido Boulevard en Long Beach, observamos que presenta una gran similitud con las regiones de color amarillo tales como las avenidas de Linden BLVD, S. Conduit Avenue, así como tambien Normandy Road, y el cruce de New Amwell Road con Auten Road. Caracterizándose por presentar precios entre 1600 a 1800 dólares, de alrededor de 1600 pies cuadrados, con 3 y 4 camas, sin acceso a sillas de rudas. Las zonas menos similares de color rojo, tales como las que se ubican en Newark, donde se tienen departamentos, de entre 700 y 100 pies cuadrados, de 1 a 2 dormitorios, presentando acceso a sillas de ruedas. Por otro lado, las que se encuentran aún más distantes, de color morado, como las de South Amboy, Twin Rivers, Bensalem que tienen precios de alrededor de 1400 dólares americanos.

En la Fig. 5.5 (c-d) visualizamos los resultados para nuestra segunda exploración y consulta, respetivamente. En Fig. 5.5 (c) observamos alrededor de 30 agrupamientos para el nivel tres en el árbol de similitud (H:3) y nivel nueve en el QuadTree (Q:9). En la Fig.5.5 (d) establecemos como región de consulta a Rochester, observando que tiene una similitud muy grande con el centro de Buffalo, Dewitt, y LockPort, caracterizándose estos lugares por el alquiler de departamentos, de 790 pies cuadrados, donde gatos y perros está permitido, así como la presencia de 2 dormitorios. Aquellas regiones en color crema, tales como Marbletown, Olean, Batavia; que preservan una gran similitud por tener acceso a sillas de ruedas y contar con un baño en cada uno de sus departamentos de alquiler. Y aquellas que se encuentran de color morado a oscuro, son los que menos similitud tienen con respecto a nuestra consulta, ya que se trata de casas de pueblos, tales como las ubicadas en la ciudad de East Aurora, Poplar Beach, Riverside.

a) Primera visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:10

b) Resultado de la primera consulta por similitud.

c) Segunda visualización de los datos para $\boldsymbol{H}$:3, $\boldsymbol{Q}$:9



d) Resultado de la segunda consulta por similitud.

Figura 5.5: En esta figura presentamos los experimentos realizados sobre el conjunto de datos de viviendas en USA [Craigslist, 2019]. Estas gráficas fueron generados por el autor.

Capítulo 6

Conclusiones, Limitaciones y Trabajos Futuros

Los métodos actuales que permiten explorar de forma visual e interactiva grandes volúmenes de datos, no soportan consultas por similitud, representando una oportunidad para abordar esta problemática. La alta mixtura que presentan los datos del mundo real reflejan un gran reto al poder determinar la similaridad entre ellos. Dado que al calcular sus distancias, usando Gower, un atributo adicional como el categórico puede marcar una diferencia. Asi como también, la presencia o no de atributos numéricos.

La combinación entre Gower y t-SNE, obtiene los mejores resultados sobre miles de registros, al proyectar los datos mixtos a 1D, contraponiéndose a nuestra hipótesis sobre la superioridad de UMAP. Nuestros resultados indicaron que t-SNE logra superarlo tanto en precisión como el tiempo en ejecución en conjuntos de datos mixtos más grandes y diversos.

Al trabajar sobre las proyecciones de los datos, perdemos información valiosa. En ese sentido, realizamos agrupamientos sobre la matriz de gower, que al ser indizadas estratégicamente en forma de árbol, siendo conectados a una estructura espacial, nos permitió realizar nuestras consultas de forma interactiva.

La principal limitación de nuestro trabajo es la demanda de recursos de memoria RAM, debido al cálculo de la matriz de Gower, y la generación de agrupamientos a través del DBScan cuyo parámetro de entrada es la misma matriz. Llegando a procesar hasta 32 mil registros (es decir, una matriz de 32000 x 32000) en un computador estándard de 16 GB de RAM. También es importante señalar que nuestra primera versión basado en proyecciones, depende fuertemente de la precisión del método de proyección. Asímismo, nuestra segunda versión, del método de agrupamiento.

Frente a las consultas puntuales, nuestras consultas por regiones permiten visualizar otras regiones cuyos agrupamientos son los más parecidos. Logrando responder: ¿Cuáles son los avenidas o carreteras cuyos accidentes sean los más y menos similares que las nuestra región de consulta? o ¿Cuáles son las calles con características de vivienda similares a nuestra consulta?.

Como parte de los trabajos a futuro, consideramos la implementación óptima del método DBScan en C++. Asimismo, la agregación de nuevos componentes visuales que permitan identificar los atributos que más se correlacionan dentro de la tabla de centroides. Finalmente la indización y visualización de la información temporal dentro de la herramienta.

Bibliografía

Flora Amato, Aniello De Santo, Francesco Gargiulo, Vincenzo Moscato, Fabio Persia, Antonio Picariello, and Silvestro Roberto Poccia. Semtree: An index for supporting semantic retrieval of documents. 2015:62–67, 06 2015. doi: 10.1109/ICDEW.2015.7129546.

K. Bache. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Craigslist. Sold housing options. 2019. doi: https://www.kaggle.com/austinreese/usa-housing-listings.

Harish Doraiswamy, Vo Huy, Claudio Silva, and Juliana Freire. A gpu-based index to support interactive spatio-temporal queries over historical data. pages 1086–1097, 05 2016. doi: 10.1109/ICDE.2016.7498315.

Gintautas Dzemyda and Olga Kurasova. *Multidimensional data visualization. Methods and applications*, volume 75. 05 2015. doi: 10.1007/978-1-4419-0236-8.

David W. Goodall. A new similarity index based on probability. *Biometrics*, 22(4):882–907, 1966. ISSN 0006341X, 15410420. URL `http://www.jstor.org/stable/2528080`.

J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971. ISSN 0006341X, 15410420. URL `http://www.jstor.org/stable/2528823`.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, January 1997. ISSN 1384-5810. doi: 10.1023/A:1009726021843. URL `https://doi.org/10.1023/A:1009726021843`.

Ramesh Janipella, Vikash Gupta, and Rucha Moharir. *Application of Geographic Information System in Energy Utilization*, pages 143–161. 01 2019. ISBN 9780444640833. doi: 10.1016/B978-0-444-64083-3.00008-7.

Christian Jensen, Copyright Christian, S. Jensen, Richard Snodgrass, Christian (codirector, Michael Böhlen, Renato Busatto, Heidi Gregersen, Kristian Torp, Richard (codirector, Anindya Datta, and Sudha Ram. Temporal data management. 07 1997.

C. Li and G. Biswas. Unsupervised learning with mixed numeric and nominal data. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):673–690, 2002. doi: 10.1109/TKDE.2002.1019208.

Mingzhao Li, Farhana Choudhury, Zhifeng Bao, Hanan Samet, and Timos Sellis. Concavecubes: Supporting cluster-based geographical visualization in large data scale. 06 2018. doi: 10.13140/RG.2.2.36113.53608.

Lauro Lins, James Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE transactions on visualization and computer graphics*, 19: 2456–65, 12 2013. doi: 10.1109/TVCG.2013.179.

Can Liu, Cong Wu, Hanning Shao, and Xiaoru Yuan. Smartcube: An adaptive data management architecture for the real-time visualization of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2019. doi: 10.1109/TVCG.2019.2934434.

Dongyu Liu, Panpan Xu, and Liu Ren. Tpflow: Progressive partition and multidimensional pattern extraction for large-scale spatio-temporal data analysis. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2018. doi: 10.1109/TVCG.2018.2865018.

Zhicheng Liu, Biye Jiang, and Jeffrey Heer. immens: Real-time visual querying of big data. *Computer Graphics Forum*, 32, 06 2013. doi: 10.1111/cgf.12129.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

Fabio Miranda, Lauro Lins, James Klosowski, and Claudio Silva. Topkube: A rank-aware data cube for real-time exploration of spatiotemporal data. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 02 2017. doi: 10.1109/TVCG.2017.2671341.

Sobhan Moosavi, Mohammad Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. A countrywide traffic accident dataset. 06 2019a.

Sobhan Moosavi, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. Accident risk prediction based on heterogeneous sparse data. *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Nov 2019b. doi: 10.1145/3347146.3359078. URL `http://dx.doi.org/10.1145/3347146.3359078`.

Gina Lucia Muñoz Salas. Estudio de distancias para datos mixtos para análisis visual de datos multidimensionales. Master's thesis, Universidad Católica San Pablo, Arequipa, Perú, 2019.

Cícero Pahins, Sean Stephens, Carlos Scheidegger, and João Comba. Hashedcubes: Simple, low memory, real-time visual exploration of big data. *IEEE Transactions on Visualization and Computer Graphics*, 23:1–1, 01 2016. doi: 10.1109/TVCG.2016.2598624.

Cícero Pahins, Nivan Ferreira, and Joao Comba. Real-time exploration of large spatiotemporal datasets based on order statistics. *IEEE transactions on visualization and computer graphics*, PP, 05 2019. doi: 10.1109/TVCG.2019.2914446.

Roger Peralta-Aranibar, Cícero Pahins, João Comba, and Erick Gomez-Nieto. Similarity-based visual exploration of very large georeferenced multidimensional datasets. 04 2019a. doi: 10.1145/3297280.3297556.

Roger Peralta-Aranibar, Cícero Pahins, João Comba, and Erick Gomez-Nieto. Similarity-based visual exploration of very large georeferenced multidimensional datasets. 04 2019b. doi: 10.1145/3297280.3297556.

Julio Toss, Cícero Pahins, Bruno Raffin, and João Comba. Packed-memory quadtree: A cache-oblivious data structure for visual exploration of streaming spatiotemporal big data. *Computers Graphics*, 76, 09 2018. doi: 10.1016/j.cag.2018.09.005.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

Zhe Wang, Nivan Ferreira, Youhao Wei, Aarthy Bhaskar, and Carlos Scheidegger. Gaussian cubes: Real-time modeling for visual exploration of large multidimensional datasets. *IEEE Transactions on Visualization and Computer Graphics*, 23:1–1, 01 2016. doi: 10.1109/TVCG.2016.2598694.

Zhe Wang, Dylan Cashman, Mingwei Li, Jixian Li, Matthew Berger, Joshua A. Levine, Remco Chang, and Carlos Scheidegger. Neuralcubes: Deep representations for visual data exploration, 2019.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37 – 52, 1987. ISSN 0169-7439. doi: https://doi.org/10.1016/0169-7439(87)80084-9. URL `http://www.sciencedirect.com/science/article/pii/0169743987800849`. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

Keyu Yang, Xin Ding, Yuanliang Zhang, Lu Chen, Baihua Zheng, and Yunjun Gao. Distributed similarity queries in metric spaces. *Data Science and Engineering*, 4:1–16, 06 2019. doi: 10.1007/s41019-019-0095-7.

Wenhui Zhou, Chunfeng Yuan, Rong Gu, and Yihua Huang. Large scale nearest neighbors search based on neighborhood graph. pages 181–186, 12 2013. ISBN 978-1-4799-3261-0. doi: 10.1109/CBD.2013.20.