UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Diseño e Implementación de un sistema para la Detección de Cortocircuitos mediante técnicas de Data Stream y Web Semántica.

Autor: Luis Ernesto Hurtado González

Tutores: Dr.C. Amed Abel Leiva Mederos
Dr.C. Boris Villazón Terrazas
Consultante: Ing. Rigoberto Acosta González

Santa Clara

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Diseño e Implementación de un sistema para la Detección de Cortocircuitos mediante técnicas de Data Stream y Web Semántica.

Autor: Luis Ernesto Hurtado González

E-mail: luhgonzalez@uclv.cu

Tutores: Dr.C. Amed Abel Leiva Mederos

E-mail: amed@uclv.edu.cu

Dr.C. Boris Villazón Terrazas

E-mail: boris.villazon.terrazas@gmail.com

Santa Clara

Año 2021

"Año 63 de la Revolución "

PENSAMIENTO

"El primer paso es establecer que algo es posible, entonces la probabilidad ocurrirá"

Elon Musk

DEDICATORIA

.

A mis padres, Lisset y Raúl, a mi hermana, mi tía y mis abuelos.

AGRADECIMIENTOS

A mis padres Lisset y Raúl por el sacrificio de tantos años, el apoyo incondicional, por estar siempre y dar lo mejor de sí en todo momento para mi bienestar…a ustedes debo lo que soy

A mi hermana Lisbet, a quien espero sirva de ejemplo…te quiero mucho pequeña

A mi tía Yanet por estar siempre para mí y ser mi amiga, mi otra madre

A mis abuelos Felo y Fefi por su amor, su dedicación, por atesorarme siempre y cuidarme tanto, espero hacerlos sentir orgullos

A mi novia Danay por ser la persona que me ha acompañado en días y noches, por su amor, dedicación, comprensión y apoyo durante esta etapa

A Amed, por guiarme y presentarme este proyecto

TAREA TÉCNICA

Para confeccionar el presente trabajo y alcanzar los resultados esperados, fue necesario elaborar las tareas técnicas siguientes:

- Estudio de las características generales la Web Semántica.
- Precisar las técnicas de Inteligencia Artificial asociadas a data stream y seleccionar una de ellas.
- El diseño de la arquitectura para el Sistema y los modelos de comunicación.
- El trabajo con los softwares seleccionados, con el fin de una posible implementación real.
- Elaborar el informe final del Trabajo de Diploma.

Las tecnologías aplicadas a esta investigación se desarrollaron en el centro de investigaciones de informática de la UCLV, por tanto, es el propietario de la tecnología para el desarrollo de estos Sistemas.

Firma del Autor

Firma del Tutor

RESUMEN

El presente Trabajo de Diploma se enfoca en el diseño de un Sistema para la detección de cortocircuitos mediante técnicas de Web Semántica y Data Stream. Primeramente, se exponen los principales referentes teóricos de la Web Semántica, Inteligencia Artificial y Big Data. Se diseña una arquitectura para el Sistema, así como el modelo de comunicación a utilizar. Se describen los diferentes procesos por los que pasan los datos desde que se generan en los sensores hasta que son visualizados por el usuario. Se aplica el algoritmo de Machine Learning (K-Means) para clasificar los streams. Se mapean en el grafo existente en el triple store y se visualizan en un mapa geográfico los resultados.

Palabras clave: IoT, Sistema Eléctrico de Potencia, Ontología, Data Streams, K-Means

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los Sistemas Eléctricos de Potencia se han convertido en una de las bases fundamentales del desarrollo humano. Se encargan de llevar la energía eléctrica desde las plantas generadoras hasta los grandes centros de consumo tales como ciudades e industrias. Debido a esto, los sistemas de potencia se ven envueltos en un constante cambio, el cual permite brindar un servicio fiable y seguro.

En dicho proceso ocurren fallas que tienen origen por fenómenos naturales como; tormentas, relámpagos, el deterioro de los aislamientos, e incluso son causados por aves, árboles y accidentes humanos. Un sistema eléctrico presenta fallas del tipo serie (ruptura de conductores) y tipo paralelo (cortocircuito a tierra o entre fases).

La falla que más daño causa en un sistema de potencia es el cortocircuito, el mismo es producido cuando entran en contacto entre sí o con tierra conductores energizados de diferentes fases. Encontrar los cortocircuitos en las líneas de transmisión y postes, ha sido un tema preocupante para los operadores de red. Se sabe que las líneas se caracterizan por tener inmensas longitudes y grandes cantidades de postes. Ante la ocurrencia de una falla, es difícil conocer el punto preciso donde se genera. Esto causa grandes incertidumbres e inconvenientes, ya que se vuelve totalmente necesario revisar tramo por tramo para localizar el punto causante de la anomalía.

El desarrollo de la ciencia y la tecnología en diversos campos ha permitido el empleo de herramientas virtuales para el manejo y control de los sistemas eléctricos. Estas son implementadas fundamentalmente en países desarrollados, que constituyen una ventaja significativa para la detección y localización rápida de fallas en sistemas eléctricos y su solución. El Internet de las Cosas (IoT) constituye la base tecnológica para el empleo de dichas herramientas.

La innovación y el desarrollo en empresas líderes en tecnología llevan a trazar una marcada diferencia entre su uso en los países más desarrollados y aquellos que no cuentan con las infraestructuras digitales avanzadas necesarias, como es el caso de América Latina.

En nuestro país ocurren muchas fallas en el sistema eléctrico nacional, provocadas sobre todo por cortocircuitos. No hay métodos eficientes para el monitoreo y localización de estas. Cuando ocurren fallas en el sistema eléctrico nacional el tiempo de respuesta de la Empresa Eléctrica no es el más rápido, afectando a una gran cantidad de usuarios. Por tal razón, se necesita hallar métodos eficaces para dar respuesta a este problema.

Lo anterior constituye la situación problemática de la presente investigación. De ahí que el problema de investigación es: ¿Cómo contribuir mediante el uso de las IoT y Web Semántica a la detección de cortocircuitos en Sistemas Eléctricos de Potencia?

Esta investigación tiene como **objeto** los sistemas para la detección de fallas en el Sistema Eléctrico de Potencia con Web Semántica y stream data. El **campo de estudio** es la detección de cortocircuitos en el Sistema Eléctrico de Potencia mediante stream data y Web Semántica en Cuba.

Para dar solución al problema científico planteado se traza el siguiente **objetivo general**: Desarrollar un sistema para la detección de cortocircuitos en Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica.

Para resolver el problema de investigación y dar cumplimiento al objetivo general, se plantean los siguientes **objetivos específicos**:

- ✓ Determinar los referentes teóricos para la detección de cortocircuito en los Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica.
- ✓ Diseñar el sistema para la detección de cortocircuitos en Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica.
- ✓ Implementar el sistema para la detección de cortocircuitos en Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica.

Para la realización de esta investigación fueron empleados métodos de investigación como: métodos teóricos (análisis documental, inductivo-deductivo, histórico-lógico y el analítico-científico), y la modelación como método empírico.

Los resultados de esta investigación influyen de manera positiva en el tratamiento y manejo del sistema eléctrico nacional. En función del cumplimiento de los objetivos trazados este trabajo de diploma se estructura de la siguiente manera: un resumen en español e inglés, un índice, la introducción, tres capítulos, conclusiones, recomendaciones y anexos.

El capítulo 1 aborda los principales referentes teóricos que soportan la integración semántica de datos, en él se exponen los principales elementos de la Web Semántica y se relatan elementos del sistema que son tomados en consideración para el desarrollo de la investigación.

En el capítulo 2 se identifica el paradigma utilizado en el desarrollo del método propuesto, se realiza la definición del método utilizado a la vez que se describen cada uno de sus componentes. Se detalla el proceso de desarrollo según la metodología seleccionada, así como la arquitectura del sistema utilizada.

El capítulo 3 relata la selección de la estrategia de validación, la preparación del caso de estudio, su diseño con cada una de las etapas que lo componen. Se realiza un análisis de los datos recolectados que permite comprobar la validez de la hipótesis formulada para conducir la investigación.

La tesis consta de 91 páginas, con 2 tablas y 37 figuras incluidas, así como 10 anexos. Se revisaron 103 fuentes bibliográficas.

CAPÍTULO 1. REFERENTES TEÓRICOS SOBRE LOS SISTEMAS PARA LA DETECCIÓN DE FALLAS EN EL SISTEMA ELÉCTRICO DE POTENCIA CON WEB SEMÁNTICA Y STREAM DATA.

En este capítulo se estudian las principales características de la Web Semántica, internet de las cosas y Big Data. Así como las técnicas de Inteligencia Artificial que existen y son útiles para dar cumplimiento a la tarea. Se exponen los diferentes tipos de cortocircuitos y se referencia teóricamente sobre Data Stream.

1.1 Cortocircuitos en Sistemas Eléctricos de Potencia

Se define el término falla como *cualquier cambio no planeado en las variables de operación de un sistema de potencia*, también es llamado perturbación, y es causada por:

- *Falla en el sistema de potencia* (Cortocircuito)
- *Falla extraña al sistema de potencia* (En equipos de protección)
- *Falla en la red* (Sobrecarga, fluctuación de carga, rayos, contaminación, sabotajes, daños) [1].

Cuando se produce una falla las magnitudes asociadas al Sistema Eléctrico de Potencia alcanzan valores situados fuera de sus rangos normales de funcionamiento. Por lo que determinadas áreas del sistema pueden pasar a operar en condiciones desequilibrada, con el riesgo que ello conlleva para los diferentes elementos que lo integran [2]. La falla más rigurosa es el cortocircuito trifásico, siguiendo el bifásico a tierra, bifásico, monofásico y las fases abiertas. La ubicación más desfavorable depende del sistema, pero corresponde generalmente a puntos cercanos al generador [3].

Como se ha mencionado anteriormente entre las fallas más comunes están los cortocircuitos, este es un fenómeno eléctrico que ocurre cuando dos puntos entre los cuales existe una diferencia de potencial se ponen en contacto entre sí, caracterizándose por elevadas corrientes circulantes hasta el punto de falla [4]. Éstos se pueden clasificar según como se muestra la Figura 1.1 donde se exponen las clasificaciones de los cortocircuitos de acuerdo a la forma de contacto de los conductores y algunos ejemplos de cortocircuitos basados en el número de fases que intervienen.

Las fallas que existen por cortocircuitos son: trifásico simétrico, aislado o a tierra, bifásico aislado (cortocircuito entre dos líneas), bifásico a tierra (entre dos líneas y el conjunto a tierra) y monofásico

(una línea conectada a tierra). Los cortocircuitos asimétricos involucran una, dos o tres fases y llevan al sistema a un nuevo punto de operación con corrientes elevadas, pero con un sistema desbalanceado y los cortocircuitos trifásicos simétricos implican las tres fases y llevan al sistema a un nuevo punto de operación elevadas pero el sistema continua siendo simétrico balanceado [5]. En el Anexo I se muestra los diferentes tipos de cortocircuitos a través del conocimiento de variables [6].

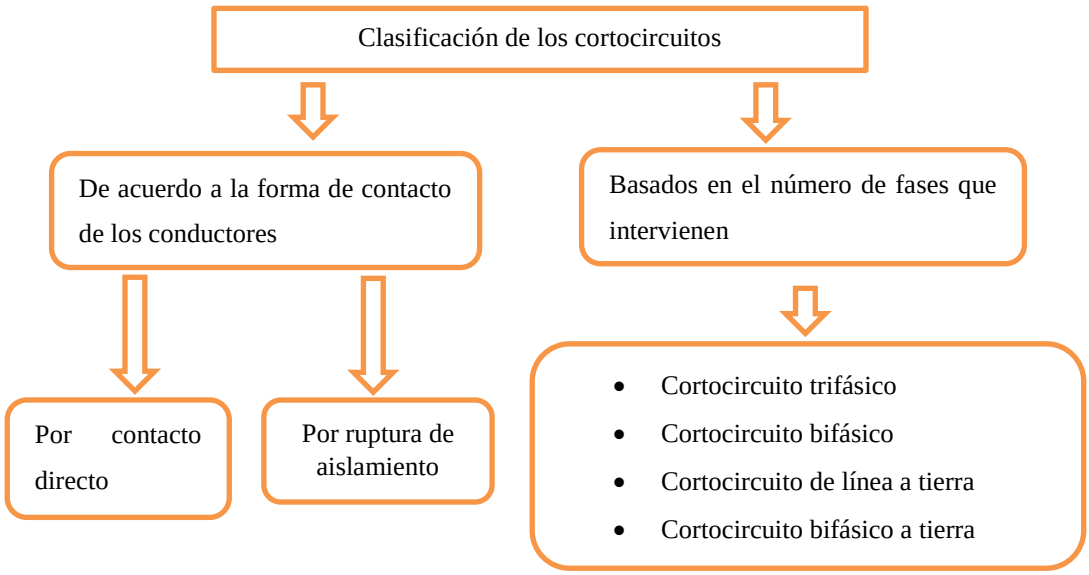


Figura 1.1 Clasificación de los cortocircuitos en Sistemas Eléctricos de Potencia.

Existen varios métodos para la correcta detección de fallas [7], [8], [9], y su localización [10], [11]. Éstos son muy precisos, pero carecen de una integración con las nuevas tendencias de las tecnologías. Son técnicas que requieren fundamentalmente, de la labor humana para aplicarse. El tiempo es un factor esencial a la hora de detectar, localizar y actuar sobre la falla. Este es inversamente proporcional a la gravedad de los daños provocados por la falla.

La ingeniería del conocimiento e Inteligencia Artificial ha logrado avances significativos en la solución de problemas relacionados con los Sistemas Eléctricos de Potencia. No responden a un algoritmo matemático definido, por lo que el desarrollo de sistemas de conocimientos o expertos, imitan el comportamiento de un experto humano en la solución de problemas complejos que requieren de un conocimiento amplio y mucha experiencia [12].

1.2 Tecnologías para el Manejo de Servicios de Telecomunicaciones. Internet de las cosas (IoT).

Desde hace unos años, ha cambiado la forma en que interactuamos en diferentes entornos de nuestra vida. Dicho cambio se indujo a partir de la evolución de las redes de comunicaciones tales como internet, permitiendo con el paso del tiempo, que objetos con cierta capacidad de procesamiento y almacenamiento se conectaran a internet, vislumbrando así el concepto del internet de las cosas (IoT) [13].

Según Luis García [14], el IoT es una revolución tecnológica que representa el futuro de la informática y las comunicaciones, su desarrollo depende de la dinámica de innovación técnica en varios campos importantes, desde comunicaciones inalámbricas, sensores hasta nanotecnología. Las redes de suministro de electricidad, por ejemplo, serán una parte del universo de información del Internet.

El IoT se ha convertido en una de las infraestructuras de red más importantes dentro de una ‘‘Smart City’’ (ciudad inteligente) [15]. Los sensores inalámbricos (Wireless Sensor Networks) se comportan como la puerta de enlace entre una Smart City e IoT [16]. Éstos están ubicados en diferentes puntos de la ciudad con el objetivo de captar datos importantes. El estándar IEEE 802.15.4 [17] recoge todo lo relacionado con este tema. Las aplicaciones que se le pueden dar a esta tecnología son inmensas [18], [19], [20].

La arquitectura para monitorizar la ciudad a través de los sensores es la que se muestra en la Figura 1.2. La transmisión de datos se realiza desde el sensor usando una puerta de enlace a Internet que presenta un protocolo de comunicación, los cuales varían según la finalidad que se le va a dar al servicio [21].

El sistema provee aplicación para los usuarios. Los datos de sensores diversos son coleccionados y registrados en bases de datos. Entonces, el último valor puede ser ostentado en consola. Los resultados de leer sensores también pueden ser exhibidos en la aplicación de la consola [22].

También se puede considerar el Internet de las Cosas como una aplicación especial de la Web Semántica que pretende realizar el procesamiento intelectual y compartir de forma más amplia la información del producto basado en la plataforma de la Web Semántica. La importancia de esta clasificación es que los resultados de investigación y la tecnología avanzada en la Web Semántica se pueden utilizar como referencia con el fin de acelerar la realización de Internet de las Cosas [23].

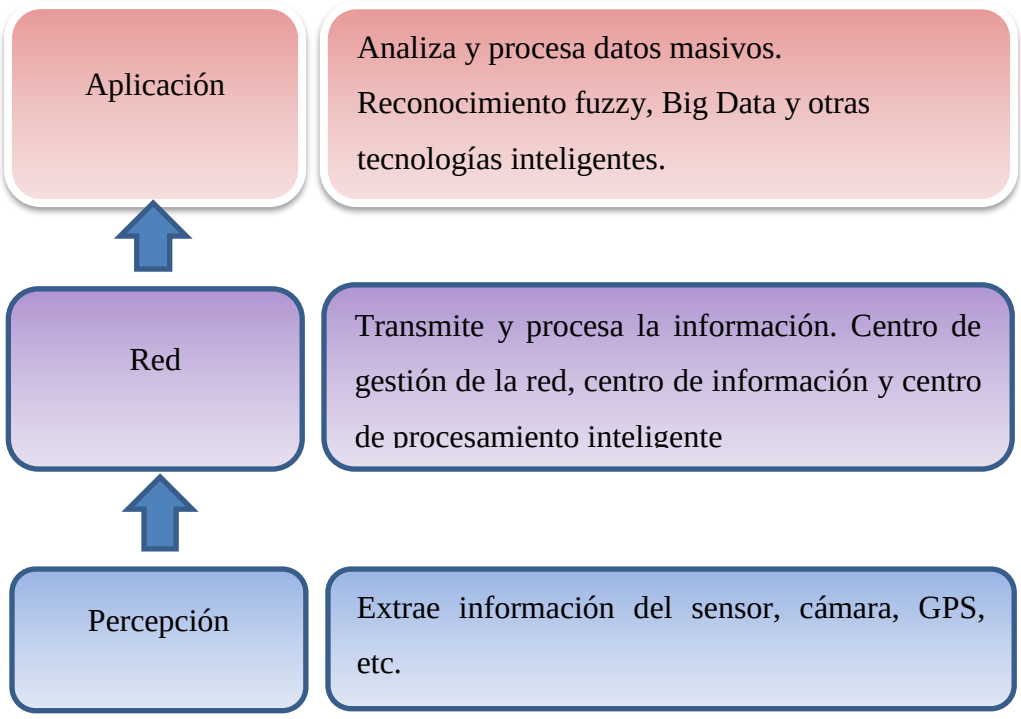


Figura 1.2 Arquitectura de una Smart City [22].

1.3 Web Semántica

Según Tim Berners-Lee (considerado el padre de la Web y miembro del World Wide Web Consortium (W3C)) [24], la Web Semántica es una extensión de la web actual. Permite en trabajo cooperativo entre personas y computadoras otorgando un significado bien definido a la información. De tal manera que diferentes aplicaciones razonen, ubiquen y reutilicen el contenido de la Web a través de datos bien definidos.

La web 3.0 (como también es llamada la Web Semántica) es un movimiento que busca darle a la Web una estructura bien definida y cambiar el manejo actual de la información [25].Los principales componentes de la Web Semántica son los metalenguajes y los estándares de representación XML, XML Schema, RDF, RDF Schema y OWL, así como el lenguaje SPARQL para la consulta de datos RDF [26].

Para comprender la dimensión que puede tener en la actualidad, el trabajo público de estas comunidades ha conducido a avances en la Web de hoy [27] :

- Linked Data (datos vinculados), en la forma estructurado, mecanografiado y derreferenciado. Organizaciones como BBC y New York Times, desarrollan su contenido como Linked Data.
- Google, Yahoo!, Microsoft, Facebook, y muchas otras compañías grandes de la Web, así como también numerosos proyectos de investigación revelan gráficos de conocimiento, que definen, estructuran y vinculan cientos de millones de entidades, mejoran la búsqueda, proporcionan una mejor correspondencia publicitaria, mejoran las respuestas de sus asistentes personales artificiales, etc.
- Los sistemas comerciales de administración de bases de datos (por ejemplo, Oracle) brindan soporte nativo para lenguajes de Web Semántica.
- Mas de 2.5 billones de páginas web tienen un marcado que se ajusta al formato schema.org, que les permite describir con precisión el contenido estructurado en sus sitios utilizando un vocabulario compartido.
- La Organización Mundial de la Salud se está desarrollando para que las enfermedades sean utilizadas por todos los países miembros de las Naciones Unidas como una ontología disponible en la Web.

Tanto los datos del dominio como la información de trasfondo necesaria para poder realizar el razonamiento se modelan en colecciones de términos, relaciones, clases y propiedades, en forma de vocabularios controlados, taxonomías u ontologías. Utilizando lenguajes de marcado como, RDF, RDF(S) u OWL. Toda esta información se almacena en triples o tripletas, compuestos de un sujeto y un objeto unidos por una relación. Las clases se interrelacionan con otras por estas relaciones o roles. Los individuos pertenecientes a estas clases cuentan con diferentes propiedades o datos. La información modelada de esta manera también se puede abstraer como grafos con nodos que representan las clases, interconectados por las relaciones, representadas como flechas [28]. La Figura 1.3 proporciona información acerca de la relación entre tecnologías de la Web Semántica.

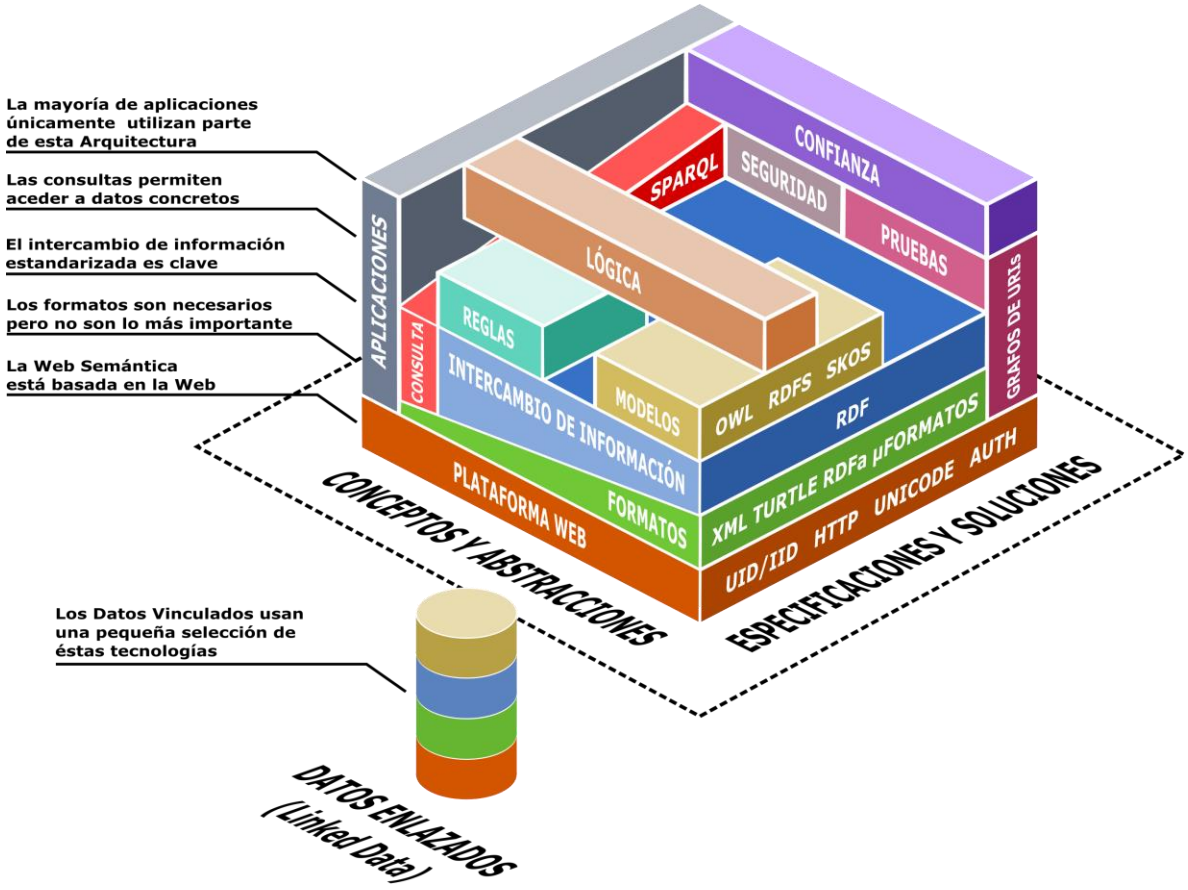


Figura 1.3 Relación entre tecnologías de la Web Semántica, disponible en: https://www.researchgate.net/figure/Semantic-Web-technologies-and-Linked-Data-231-The-Linked-open-data-approach_fig5_242878178

1.3.1 Ontología

Las ontologías son el mecanismo para el almacenamiento y el razonamiento sobre la información semántica de los objetos. Es definida por Lourdes Morán [29] como una taxonomía de conceptos con atributos y relaciones, que proporciona un vocabulario consensuado para definir redes semánticas de unidades de información interrelacionadas. Según María Keet [30] una ontología es un artefacto de ingeniería que debe tener un formato procesable por máquina que se adhiera fielmente a la lógica.

Los componentes más comunes de una ontología son [31]:

- ***Individuos:*** instancias u objetos (lo básico u objetos de "bajo nivel")
- ***Clases:*** conjuntos, colecciones, conceptos, clases en programación, tipos de objetos, o tipos de cosas.

- ***Atributos:*** aspectos, propiedades, rasgos, características, o parámetros que objetos (y clases) pueden tener.
- ***Relaciones:*** formas en la cual clases y los individuos se pueden relacionar unos con otros.
- ***Funciones:*** complejas estructuras formadas de cierta relación que pueden ser usada en lugar de un término individual en una declaración.
- ***Restricciones:*** establecen descripciones formales de lo que debe ser verdad con el objetivo de que alguna aserción pueda ser aceptada como entrada.
- ***Reglas:*** declaraciones con forma de oraciones que describen inferencias lógicas que puede ser derivables de una aserción en una forma particular.
- ***Axiomas:*** aserciones (incluyendo reglas) en una forma lógica que juntos incluyen toda la teoría que la ontología describe en su dominio de aplicación.
- ***Eventos:*** los cambios de los atributos o relaciones.

1.3.1.1 RDF

El marco de descripción de recursos (RDF) es un modelo de datos para los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.

El modelo de datos RDF es similar a los enfoques de modelado conceptual clásicos como entidad-relación o diagramas de clases, ya que se basa en la idea de hacer declaraciones sobre los recursos (en particular, recursos web) en forma de expresiones sujeto-predicado-objeto. Estas expresiones son conocidos como triples en terminología RDF. El sujeto indica el recurso y el predicado denota rasgos o aspectos del recurso y expresa una relación entre el sujeto y el objeto [32].

Informalmente, el modelo de datos RDF pueden definirse en base a cuatro conceptos [33]:

- **Recursos**: Un recurso puede ser definido como un objeto o cosa que se desea describir. Los recursos pueden ser personas, libros, páginas web, o cualquier otra cosa, real o abstracta. Cada recurso en RDF es identificado de manera única a través de una IRI (Internationalized Resource Identifier).
- **Propiedades**: Una propiedad se refiere a un atributo o una relación de un recurso. Un atributo consiste en una característica propia de un recurso, la cual tiene un recurso específico. En RDF,

las propiedades son consideradas tipos especiales de recursos, por lo tanto, también se identifican usando IRIs.

- **Declaración**: Una declaración establece una afirmación precisa sobre alguna propiedad de un recurso. Una declaración se representa usando una estructura especial denominada triple RDF.
- **Descripción**: El conjunto de declaraciones (o triples RDF) asociadas a un recurso conforman la descripción del mismo. La descripción de un dominio de aplicación está conformada por la unión de las descripciones de todos los recursos del dominio.

La combinación de RDF con otras herramientas como RDF Schema y OWL permite añadir significado a las páginas, y es una de las tecnologías esenciales de la Web Semántica.

1.3.1.2 OWL

Como su nombre lo indica OWL (Ontology Web Language), es un lenguaje ontológico de la web. Se basa en el esquema RDF y RDFs. Permite realizar un razonamiento de la descripción lógica de la Web Semántica. OWL es una extensión de RDF por lo que posee todas sus características y otras adicionales como son la aplicación de operaciones lógicas, además atribuye a las relaciones ciertas propiedades como la cardinalidad, simetría, transitividad o relaciones inversas [34].

Una de las características más importantes de esta especificación es la definición de tres sublenguajes dentro de la misma: OWL Lite, OWL DL y OWL Full, aquí listados en orden creciente de complejidad, que permiten al usuario dotarse de un subconjunto de reglas sintácticas adaptado a la medida de sus necesidades [35]:

- **OWL Lite:** Proporciona un lenguaje sencillo para quienes solamente necesitan categorizar objetos y definir relaciones entre ellos.
- **OWL DL:** Extiende OWL Lite para permitir mayor expresividad, garantizando a la vez que ciertas propiedades lógicas se mantengan dentro de la ontología definida. Entre sus características está la lógica de primer orden.
- **OWL Full:** Las características más llamativas son alta expresividad, su mayor importancia es garantizar la integridad del lenguaje. En este modelo no es posible realizar un razonamiento automatizado.

Con el fin de mejorar la especificación original del lenguaje OWL, un grupo de investigadores, principalmente de la Universidad de Manchester, propuso el 19 de diciembre de 2006 una extensión del mismo llamada OWL 1.1. Esta extensión tenía como finalidad la creación de un nuevo grupo de trabajo para adicionar nuevas características como mayor expresividad sintáctica, extensiones sobre los tipos de datos y capacidad de anotaciones, metamodelado simple, entre otras. La conformación de este grupo de trabajo produjo la publicación y aceptación de la especificación OWL 2 como nueva Recomendación W3C para lenguajes de descripción del conocimiento [36].

La versión OWL 2 incorpora nuevas funcionalidades que mejoran la expresividad del lenguaje. Entre estás característica destacan la posibilidad de definir las claves en las clases, cadenas de propiedades, tipos de datos y rangos de datos más complejos, restricciones de cardinalidad cualificadas, así como la definición de propiedades asimétricas, reflexivas y disjuntas. Además incorpora tres perfiles para OWL DL [37]:

- **OWL 2 EL**. Este perfil es cercano a la lógica descriptiva EL++, la cual permite realizar tareas básicas de razonamiento en tiempos polinómicos. Es apropiado para aplicaciones que requieren ontologías muy largas y donde la prioridad sea el rendimiento.
- **OWL 2 QL**. Este perfil está orientado a aplicaciones que utilizan ontologías relativamente ligeras, que requieran grandes volúmenes de instancias de datos y donde la consulta de información a través de consultas relacionales (SQL) es de gran utilidad, ya que permite responder a consultas de un modo formal y completo en un tiempo razonable, computacionalmente hablando.
- **OWL 2 RL**. Este perfil permite implementar algoritmos con tiempo de razonamiento polinomial a través de tecnologías de bases de datos basadas en reglas que operan directamente en tripletas RDF. OWL 2 RL está orientado a aplicaciones que requieran razonamiento escalable sin sacrificar demasiado el poder de expresividad. Este perfil es adecuado para aplicaciones que hagan uso de ontologías relativamente ligeras, que organicen un gran número de individuos y donde es de gran importancia manejar directamente datos en forma de tripletas RDF.

Las ontologías se suelen codificar usando lenguajes ontológicos.

1.3.2 Vocabularios ontológicos para IoT

Las ontologías son vocabularios comunes que, junto con otras tecnologías que proveen de herramientas y lenguajes para generar marcado y procesamiento semántico, harán posible la Web Semántica.

1.3.2.1 GEO

Vocabulario especificado para la construcción de ontologías geográficas y datos geoespaciales. Su implementación es tan amplia como para ser usado por dos campos tan distantes como son la detección de desastres meteorológicos [38] y el turismo [39].

Esta ontología está definida como parte de la ontología que analiza las particiones geográficas para [40]:

- Establece cuales y que tipos de entidades geográficas que existen, y cómo pueden estar definidos y clasificados en un sistema ontológico que los junta.
- Desarrolle una teoría de representación espacial.
- Discute de qué manera y cómo emergen las descripciones geográficas de realidad desde el sentido común pueden estar combinados con descripciones de diferentes disciplinas científicas.

Una geo-ontología describe entidades, relaciones semánticas y relaciones espaciales [41]:

1. Las entidades pueden asignarse a ubicaciones en la superficie de la Tierra.
2. Las relaciones semánticas entre estas entidades incluyen, por ejemplo, hipernymy (relación de clase a subclase), hiponimia (relación de subclase a clase), mereonomía (parte de un todo y sinonimia).
3. Relaciones espaciales entre entidades (por ejemplo, adyacencia, contención espacial, proximidad y conectividad).

Sirve para establecer la localización de los cortocircuitos dentro de una línea o en un poste en específico cuando se genera el stream data, las propiedades *geo:in*, *geo:from* y *geo:until* son esenciales para conocer la localización de los mismos.

1.3.2.2 SOSA

La principal innovación de esta generación de SSN (Semantic Sensor Network) ha sido la introducción de la ontología de Sensor, Observación, Muestra y Actuador (SOSA), que proporciona un núcleo ligero

para SSN. SOSA tiene como objetivo ampliar el público objetivo y las áreas de aplicación que pueden hacer uso de las ontologías de la Web Semántica [42].

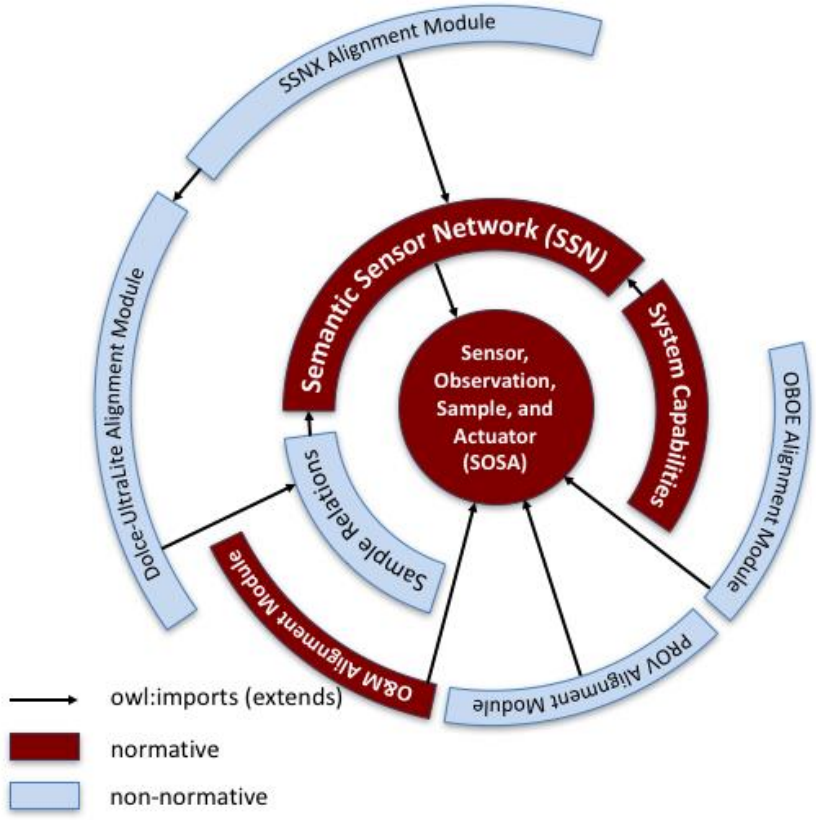


Figura 1.4 Las ontologías SOSA y SSN y sus módulos verticales y horizontales [49].

SOSA no solo reemplaza el antiguo patrón de diseño de ontología de Sensor de Estímulo-Observación (SSO), sino que proporciona un marco flexible pero coherente para representar las entidades, relaciones y actividades involucradas en la detección, el muestreo y la actuación. Está destinado a ser utilizado como un vocabulario liviano, fácil de usar y altamente extensible que atrae a una amplia audiencia más allá de la comunidad de la Web Semántica, pero se puede combinar con otras ontologías, como SSN para proporcionar una axiomatización más rigurosa donde sea necesario. Al mismo tiempo, SOSA actúa como un nivel mínimo de interoperabilidad, es decir, define aquellas clases y propiedades comunes para las cuales los datos se pueden intercambiar de manera segura a través de todos los usos de SSN, sus módulos y SOSA [43].

SSN sigue una arquitectura de modularización horizontal y vertical (Figura 1.4) al incluir una ontología central simple, pero con alto nivel de autonomía llamada SOSA (Sensor, Observation, Sample, y Actuator) por sus clases y propiedades elementales.

1.3.2.3 IotStream

IotStream es un modelo semántico ligero para la anotación de data stream que extiende la ontología SOSA (y por extensión SSN). Este se compone únicamente de cuatro conceptos principales, como se puede ver en la Figura 1.5. Estos conceptos son: IotStream, StreamObservation, Analytics y Event.

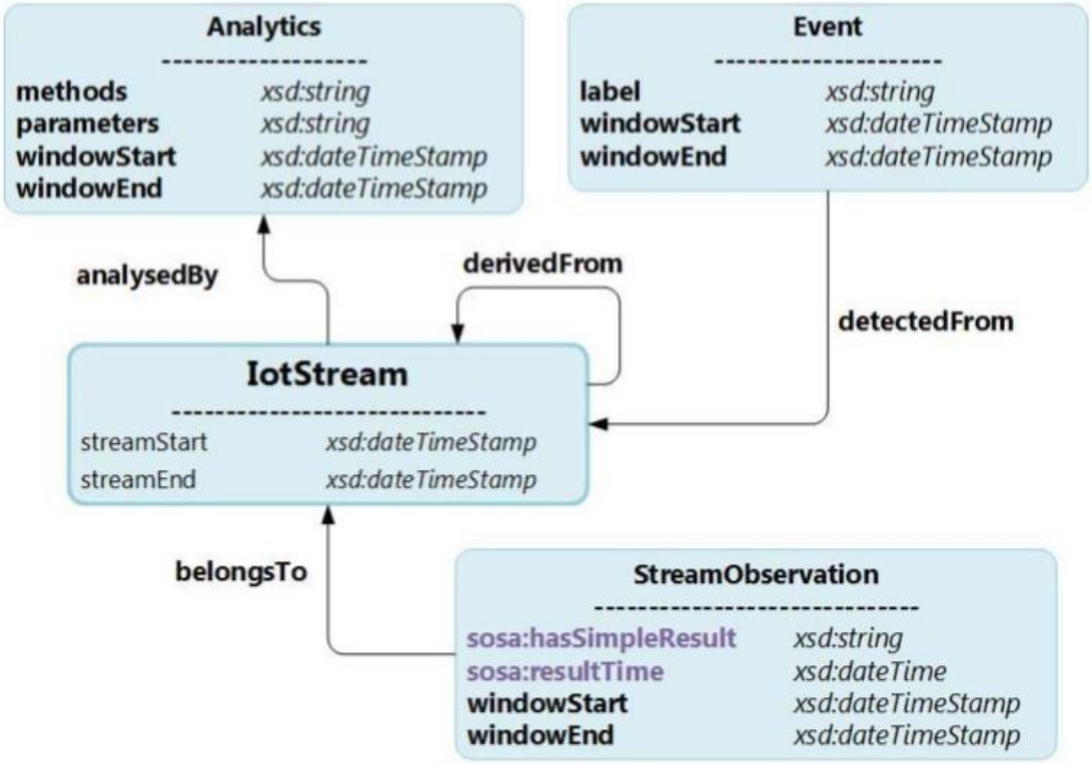


Figura 1.5 Clases y propiedades de IotStream [49].

El modelo de información de IotStream adopta varios conceptos que se consideran atributos centrales para proporcionar un contexto del mundo real. El primero se relaciona con los atributos espaciales de IotStream. La ontología geográfica del W3C proporciona un conjunto de conceptos básicos que representan la ubicación de una entidad. La noción principal de interés es el *geo:Point* que contiene propiedades geoespaciales (latitud, longitud y altitud). La ontología IoT-lite amplía las propiedades para incluir la ubicación relativa y la altitud relativa. Para mantener el contexto histórico de

StreamObservations, especialmente en el caso de la movilidad, a geo:Point se vincula a cada StreamObservation. IotStream también se asocia con un área de cobertura definida donde también es relevante. El concepto iot-lite:Coverage se usa para definiciones de cobertura simples, y GeoSPARQL, que es una ontología bien establecida para atributos espaciales.

Aunque es el dispositivo sensor el que genera el IoT stream, a través de Internet, los datos suelen ser proporcionados por un servicio de capa de aplicación TCP/IP. IoT-Lite proporciona iot-lite:Service, una clase que contiene campos relacionados con la dirección del punto final del servicio, el tipo de interfaz y el vínculo a la descripción de la interfaz, que proporciona detalles sobre cómo interactuar con el servicio.

Finalmente, a lo largo de la vida útil de un IotStream, la calidad de las observaciones del stream puede cambiar con el tiempo. Para el análisis de datos, el conocimiento de la calidad es muy importante para que se puedan aplicar medidas de adaptación cuando sea necesario. La ontología de Calidad de la Información (QoI) proporciona el concepto qoi:Quality que tiene subclases que se enfocan en un aspecto particular de la calidad, como qoi:Timeliness y qoi:Completeness of observations. La Figura 1.6 ilustra cómo IoT-Stream está vinculado a estos conceptos externos [44].

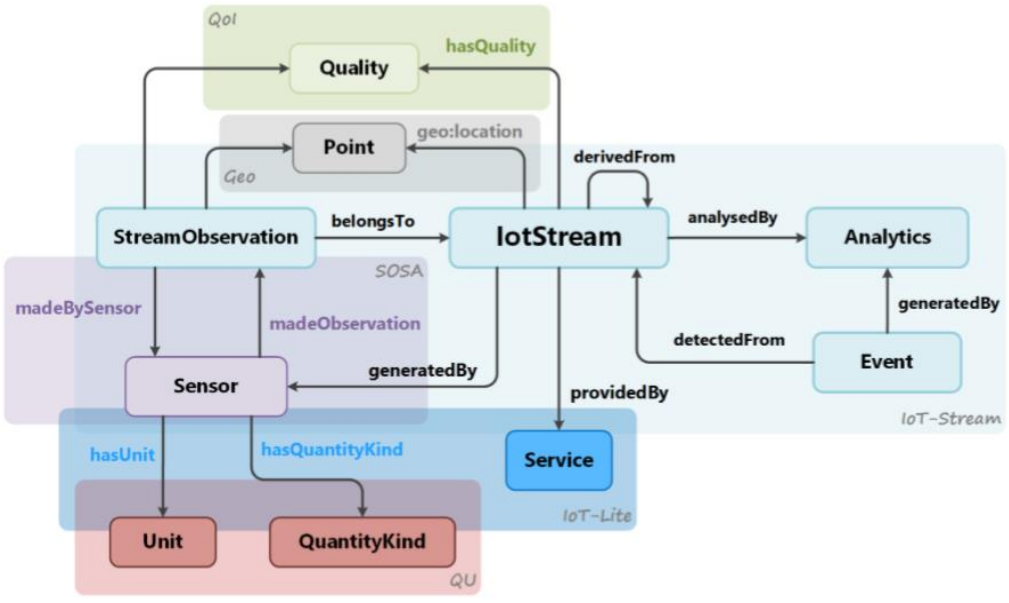


Figura 1.6 IotStream vinculada con las ontologías principales.

1.3.3 Streams

Una Ontology Stream O_m^n desde el punto de tiempo "m" hasta el punto de tiempo "n" es una secuencia de ontologías [$(O_m^n(m), O_m^n(m+1), \ldots, O_m^n(n)$] donde m, n ∈ N y m < n.
O_m^n(i) es una instantánea de una ontología de stream O_m^n en el punto de tiempo "i", refiriéndose a un conjunto de axiomas en un DL L. Una transición de O_m^n(i) a O_m^n(i+1) es una actualización [45].
En general se utiliza el término ”stream” para describir los flujos de datos donde una vez que se envía un dato éste no se va a volver a enviar nunca. El caso más común de “streams” son las señales de vídeo y audio en directo, micrófonos y sensores. Ellos no pueden volver a enviar los valores que están recogiendo y enviando a otros dispositivos ya que no disponen de un sitio para guardarlos. Los streams se usan en situaciones donde es más importante que haya un flujo continuo de datos. Que estén correctos al 100%, ya que son tolerables las incorrecciones, su volumen es tan grande que solo cabe una cantidad limitada de ellos en el dispositivo y se descarta enseguida los más antiguos y porque simplemente no interesa más que el valor que tiene ahora mismo.
Un Data Stream Management System es un sistema que está diseñado para ejecutar consultas continuas sobre un flujo continuo de datos o data stream. Existen varios tipos de arquitecturas que trabajan con stream data, en el Anexo III se muestran varias de estas. Estos datos, que llegan en cada momento, permanecen sólo por un corto periodo de tiempo en memoria, pero se obtiene constantemente nuevos datos devueltos por las consultas continuas.

Los datos de streaming son datos que se generan a partir de muchas fuentes, que normalmente envían los registros de datos simultáneamente en conjuntos de tamaño pequeño (varios kilobytes). Son heterogéneos.

Estos datos deben procesarse de forma secuencial y gradual, registro por registro o en ventanas de tiempo graduales, y se utilizan para una amplia variedad de tipos de análisis, como correlaciones, agregaciones, filtrado y muestreo. La información derivada del análisis aporta a las empresas visibilidad de numerosos aspectos del negocio y de las actividades de los clientes, como el uso del servicio, la actividad del servidor, los clics en un sitio web y la ubicación geográfica de dispositivos, personas y mercancías, y les permite responder con rapidez ante cualquier situación que surja [46]. Esto sería magnífico a la hora de detectar y corregir con tiempo algunas fallas que puedan tener los Sistemas Eléctricos de Potencias.

Las aplicaciones de ciudades inteligentes tienen la capacidad de procesar transmisiones de eventos en tiempo real, extraer información relevante e identificar valores que no siguen las tendencias generales. Más allá de la identificación de eventos relevantes, la extracción de conocimiento de alto nivel de data streams heterogéneos y multimodales es un componente importante de IoT. Las técnicas de razonamiento de stream existentes utilizan conocimientos previos y consultas de flujo para razonar sobre data stream. Las técnicas actuales no satisfacen las necesidades de IoT debido a la falta de un tratamiento adecuado de la incertidumbre (por ejemplo, posibles razones de atasco de tráfico frente a la razón más probable de atasco de tráfico) en el entorno de IoT [47].

Por otra parte, los términos "ABox" y "TBox" se utilizan para describir dos tipos diferentes de declaraciones en una base de conocimiento. Las declaraciones de TBox describen una conceptualización de un dominio de interés mediante la definición de diferentes conjuntos de individuos descritos en términos de sus características (propiedades). ABox son declaraciones que cumplen con TBox sobre individuos que pertenecen a estos conjuntos. Por ejemplo, un estudiante específico es un individuo en el conjunto llamado "Estudiante". Este conjunto se puede definir como un subconjunto de todas las personas que asisten a alguna institución educativa, lo que permite establecer la institución educativa específica a la que asiste cada individuo. Juntas, las declaraciones ABox y TBox forman una base de conocimiento o un gráfico de conocimiento [48].

La composición de la base de datos está dada por un grupo de reglas que se encargan del control de los datos garantizando el flujo coherente de información.

El lenguaje de reglas de la Web Semántica (SWRL)[1] es un lenguaje propuesto para la web 3.0 que se puede usar para expresar reglas y lógica, combinando OWL DL o OWL Lite con un subconjunto del lenguaje de marcado de reglas. Estas tienen la forma de una implicación entre un antecedente (cuerpo) y el consecuente (cabeza). El significado deseado puede leerse como: siempre que se cumplan las condiciones especificadas en el antecedente, también deben cumplirse las condiciones especificadas en el consecuente. En el Anexo II se muestra una tabla resumen acerca de las diferentes extensiones de SWRL con sus respectivas características diferenciales [49]. La arquitectura conceptual de nuestro sistema es esbozada en la Figura 1.7. Consta de tres campos: Los datos, la ontología y las reglas.

[1] https://www.w3.org/Submission/SWRL/

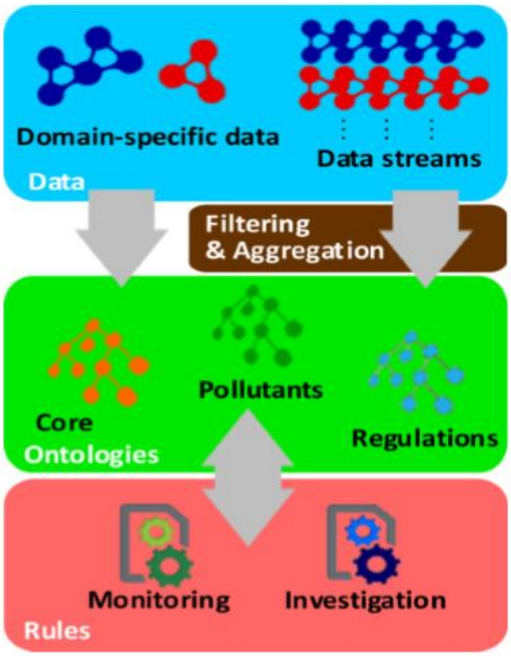


Figura 1.7 Arquitectura conceptual de C-SWRL [55].

1.3.4 SPARQL

SPARQL es un lenguaje estandarizado para la consulta de grafos RDF [50]. Para cumplir con los requisitos específicos de los datos de IoT se han desarrollado herramientas para reforzar las consultas de los datos y para extender las funcionalidades de los estándares SPARQL:

- stSPARQL y stRDF extienden el lenguaje de consultas SPARQL y las representaciones RDF con dimensiones espaciales y temporales para facilitar las consultas a los datos de los sensores las cuales dependen fundamentalmente del tiempo y lugar [51].
- C-SPARQL y SPARQL streaming son otras extensiones para incluir consultas de datos continuos [52, 53].

1.3.4.1 Consultas SparqlStreams

SPARQL streams es una extensión de SPARQL que incluye la búsqueda de datos en streams. Está basado en SPARQL 1.1. Su rasgo esencial es procesamiento de datos en streams. Además provee operadores que posibilitan la generación de nuevos streams [54].

C-SPARQL se define en términos de varias extensiones ortogonales de SPARQL, de tal manera que una consulta SPARQL normal también es una consulta C-SPARQL [55].

- ✓ Similar a los grafos RDF, cada secuencia RDF se identifica mediante el uso de un IRI, pero en lugar de ser una colección estática de triples RDF, una secuencia es una secuencia temporizada de tripletas RDF, producida continuamente por una fuente de datos. Cada triple RDF en una secuencia se anota con una marca de tiempo.
- ✓ La característica distintiva de C-SPARQL es el soporte para consultas continuas, es decir, consultas que se registran en secuencias RDF y luego se ejecutan continuamente. Las consultas C-SPARQL producen como salida los mismos tipos de salida que SPARQL: respuestas booleanas, selecciones de enlaces variables, construcciones de nuevos triples RDF o descripciones RDF de los recursos involucrados. Estas salidas se renuevan continuamente con cada ejecución de consulta. Además, las consultas C-SPARQL pueden designarse para producir nuevas secuencias RDF.
- ✓ Ampliar SPARQL a C-SPARQL requiere varias adiciones para definir streams, sus ventanas, sus marcas de tiempo y construcciones de lenguaje para agregar información de flujo. En particular, el registro de consultas y la designación de la estructura de la ventana son el aspecto clave de C-SPARQL.

EP-SPARQL (Event Processing SPARQL) es una extensión a SPARQL que posibilita el procesamiento de eventos complejos y el razonamiento continuo [56].

1.3.5 Técnicas para la transmisión de Data Stream sobre ontologías

Al existir varias fuentes de datos, estos generan una gran cantidad información que hay que procesar, darle un orden y almacenarlos. Esto no es posible sin la aplicación de una técnica que se encargue de darle el correcto procesamiento de los datos. Para solucionar este problema existen varios sistemas encargados de organizar el flujo de información recibida, guardarlo y poder realizar búsquedas en tiempo real.

1.3.5.1 OBDI y OBDA

Un sistema de integración de datos basado en ontología (OBDI) es un sistema de gestión de información que consta de tres componentes: una ontología, un conjunto de fuentes de datos y el

mapeo entre los dos. La ontología es una descripción formal y conceptual del dominio de interés para una organización dada (o una comunidad de usuarios), expresada en términos de conceptos relevantes, atributos de conceptos, relaciones entre conceptos y afirmaciones lógicas que caracterizan el conocimiento del dominio. Las fuentes de datos son los repositorios accesibles por la organización donde se almacenan los datos relacionados con el dominio. En el caso general, dichos repositorios son numerosos, heterogéneos, cada uno gestionado y mantenido independientemente de los demás. El mapeo es una especificación precisa de la correspondencia entre los datos contenidos en las fuentes de datos y los elementos de la ontología. El objetivo principal de un sistema OBDI es permitir que los consumidores de información consulten los datos utilizando los elementos de la ontología como predicados. En el caso especial en el que la organización gestiona una única fuente de datos, se utiliza el término sistema de acceso a datos basado en ontologías (ODBA) [57].

OBDA (Ontology Based Data Access) es un acercamiento para acceder la información guardada en las fuentes de datos múltiples por medio de una capa de abstracción que media entre las fuentes de datos y consumidores de datos. Muchos sistemas están implementando esta tecnología lo que los hace tener una mayor eficiencia y rapidez a sus aplicaciones. Algunos de ellos son D2RQ[2], Mastro[3], morph-RDB[4], Ontop[5], y otros. La variedad de sistemas demuestran la seguridad y eficiencia que cuenta [58]. Una línea de tiempo que representa las diferentes técnicas de integración de datos es mostrada en la Figura 1.8, la cual muestra las nuevas técnicas y lenguajes que incorpora OBDA en la actualidad.

En el paradigma de OBDA, una ontología define un esquema global de alto nivel de fuentes de datos (existentes) y brinda un vocabulario para consultas de usuario. Un sistema de OBDA reescribe tales consultas y ontologías en el vocabulario de las fuentes de datos y luego delega la evaluación real de la consulta a un sistema adecuado como por ejemplo una base de datos relacional [59].

La lógica de descripción DL-Lite, que apuntala OWL 2 QL es la que representa la ontología en OBDA. Ellos pueden reformular como un juego de preguntas correlativas encima de las fuentes [60].

---

[2] http://d2rq.org

[3] http://obdasystems.com/mastro

[4] https://github.com/oeg-upm/morph-rdb

[5] https://ontop-vkg.org/

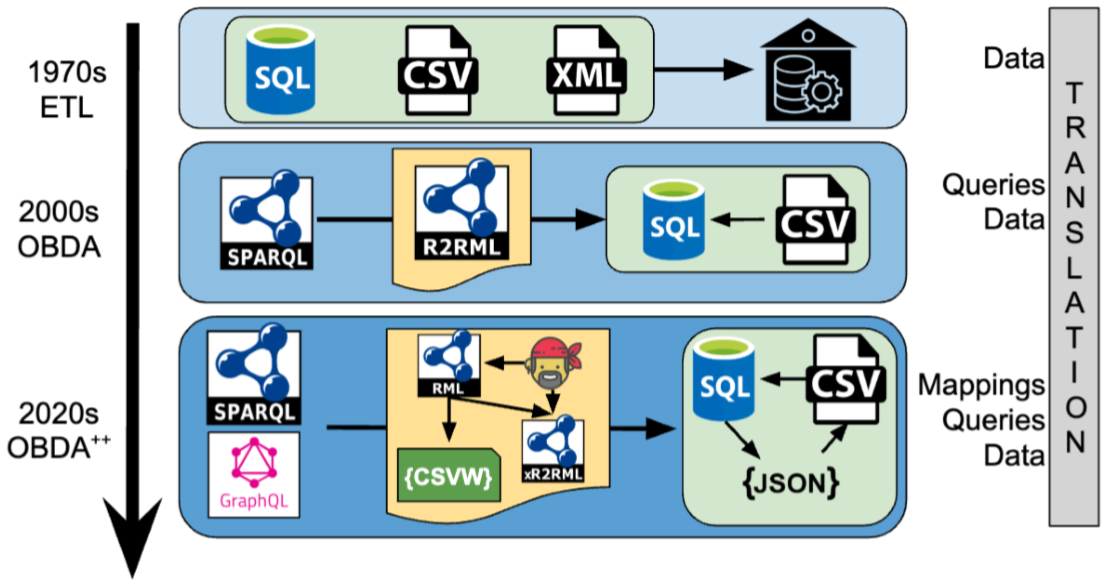


Figura 1.8 Evolución de las técnicas de integración de datos [66].

1.3.5.2 Real-Time Stream Annotation

El marco de anotación en tiempo real [61] tiene como objetivo la anotación semántica de data stream de IoT teniendo en cuenta la reducción de dimensionalidad y la confiabilidad. El marco (Figura 1.9) consta de cuatro unidades principales.

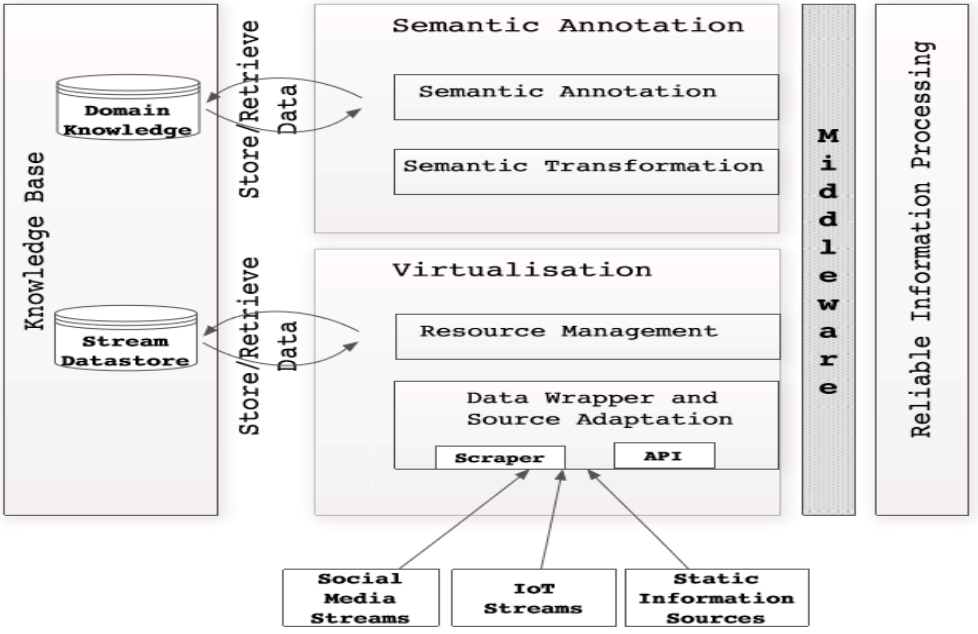


Figura 1.9 Marco de anotación en tiempo real en una aplicación.

- Virtualización

El componente de virtualización facilita el acceso a la fuente de datos heterogéneas e infraestructura que ocultan las facetas técnicas de los datos como la localización, estructura de almacenamiento, el formato de acceso y la tecnología de streaming.

En el contexto de ciudades inteligentes y procesamiento de información en tiempo real, los recursos de IoT representan los sensores y actuadores, así como almacenes de datos que coleccionan la información pertinente al funcionamiento de la ciudad. El recurso de virtualización en IoT permite el modelado de los sensores, actuadores, repositorios de datos, ciudadanos, de una manera que habilita un dispositivo, como pude ser una aplicación de falla en sistema eléctrico, un aparcamiento de autos permitiendo el acceso a los recursos sistemáticamente. Los dispositivos de comunicación (por ejemplo, los teléfonos inteligentes) también puede usarse como los sensores virtuales con la experiencia de la ciudad para observar fenómenos.

- Middleware

Existen muchas soluciones que ofrecen comunicación en sistemas distribuidos. Las deficiencias de las alternativas son combinaciones de acoplamiento de espacio (es decir, el emisor y el receptor necesitan tener referencias entre sí), el tiempo (es decir, los componentes deben interactuar al mismo tiempo) y la sincronización (es decir, los componentes individuales bloquean su actividad mientras esperan para que otros procesos terminen). Para resolver estos problemas, se utiliza un mecanismo de publicación / suscripción que desacopla el tiempo, el espacio y la sincronización. Además, la lógica de entrega de mensajes se maneja a través de un intermediario de mensajes, que lo desajusta de la capa de aplicación. En particular, se utiliza el protocolo de cola de mensajes avanzado (AMQP) que se ha introducido como un estándar abierto para middleware orientado a mensajes. El protocolo divide la tarea de intermediación de mensajes en intercambios y colas de mensajes, por lo que el intercambio decide qué mensajes se enviarán a qué cola. Esto conduce a una mayor flexibilidad para los desarrolladores y evita la necesidad de implementaciones estáticas.

- Procesamiento de Información fiable

Lo dinámico y heterogéneo que caracterizan a los ambientes IoT involucran cambios los cuales llevan a ocasionar errores en los datos. La procedencia también juega un papel importante en las aplicaciones de

las ciudades inteligentes. Las aplicaciones adquieren los datos de las fuentes heterogéneas, algunos de ellos más fiables (por ejemplo, los datos gubernamentales), y otros menos fiables. Basado en usuario o preferencias de la aplicación, el proveedor de la aplicación podría escoger usar los datos menos fiables en los casos que tiene la información más actualizada. Los datos que procesa el módulo se realiza el análisis de la procedencia para afirmar la fiabilidad de los mismos.

- Modelación de datos y anotación semántica

Las aplicaciones de ciudad inteligente utilizan datos de diferentes fuentes de transmisión. Por lo tanto, la cantidad de tráfico generado por estas aplicaciones puede ser voluminosa, particularmente para aplicaciones en tiempo real en entornos con dispositivos que tengan limitaciones de recursos, por ejemplo, sensores con ancho de banda, memoria o energía limitada. Por un lado, el modelo de datos propuesto debe ser ligero para reducir el tráfico y el tiempo de procesamiento. Por otro lado, debe representar explícitamente el significado y las relaciones de los términos en los vocabularios.

1.4 Inteligencia Artificial

El término Inteligencia Artificial (IA) se aplica cuando una máquina imita las funciones «cognitivas» que los humanos asocian con otras mentes humanas, como, por ejemplo: "aprender" y "resolver problemas".

La IA no es solo Deep Learning ni Deep Neutral Networks, sino que incluye varios subcampos como son:

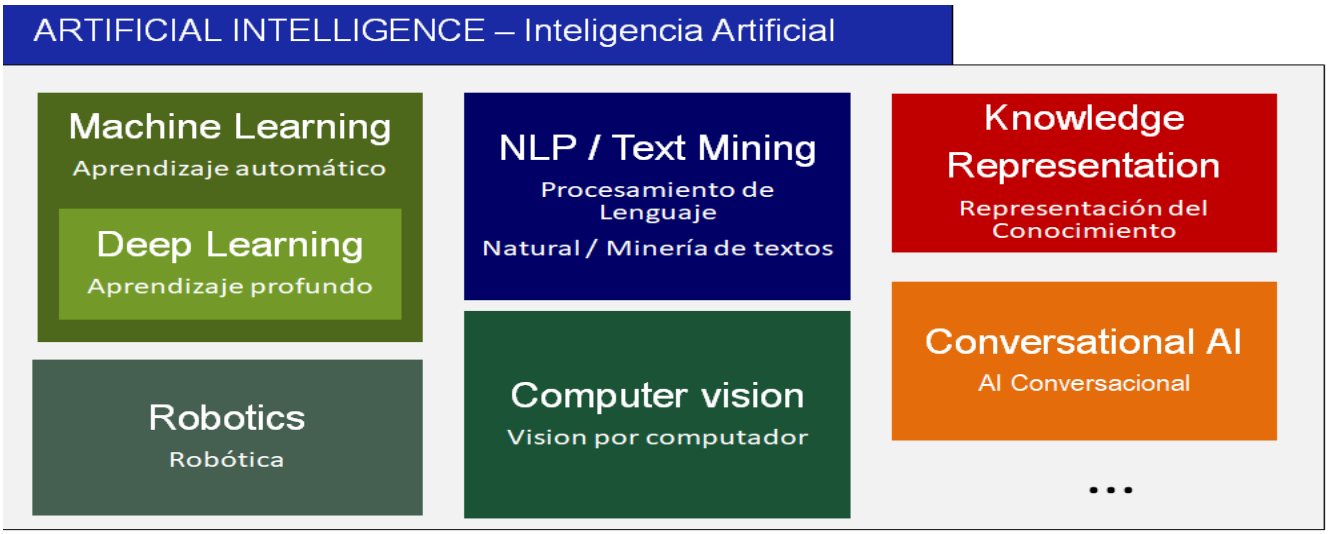


Figura 1.10 Campos de la IA. Disponible en: https://www.linkedin.com/pulse/beyond-artificial-intelligence-buzzword-boris-villazon-terrazas/

- La IA conversacional es una forma de Inteligencia Artificial que permite a las personas comunicarse con aplicaciones, sitios web y dispositivos en un lenguaje natural similar al humano a través de la entrada de voz, texto, tacto o gestos.
- La Visión por Computador es un campo de la Inteligencia Artificial y las Ciencias de la Computación que tiene como objetivo brindar a las computadoras una comprensión visual del mundo.
- La robótica es una rama de la IA, que se compone de ingeniería eléctrica, ingeniería mecánica e informática para el diseño, construcción y aplicación de robots.
- La Representación del Conocimiento es un subcampo de la Inteligencia Artificial cuyo objetivo principal es estudiar cómo se puede representar el conocimiento sobre el mundo y qué tipo de razonamiento se puede hacer con ese conocimiento.
- El aprendizaje automático o Machine Learning es un paradigma que forma parte de la Inteligencia Artificial. Un campo de investigación que formalmente enfoca en la teoría, actuación, y propiedades de aprender sistemas y algoritmos. Actualmente se encarga de reconocer patrones en conjuntos de datos y extraer información o decidir acciones que realizar en base a dichos patrones. Según el tipo de tarea y la cantidad de datos de los que precisamos se pueden utilizar diferentes tipos de algoritmos como pueden ser los árboles de decisión, las mixturas de gaussianas o las redes neuronales entre muchos otros.

Se han desarrollado frameworks para admitir versiones de transmisión masiva de algoritmos típicos de aprendizaje automático. Por ejemplo, han desarrollado un marco de este tipo [82] para la minería de data stream utilizando inducción de árbol de decisión, aprendizaje de redes bayesianas, agrupamiento de k-means y se ha desarrollado el algoritmo EM para mezclas de gaussianos.

Más recientemente, han aparecido paquetes de software similares para modelos de pronóstico como Massive Online Analysis (MOA) [83] un marco de código abierto para análisis de stream en tiempo real, Rapid-Miner [84] un sistema de minería de datos con complemento para procesamiento de transmisión, MEKA [85], una extensión de etiquetas múltiples de la popular biblioteca WEKA para aprendizaje automático, Scikit-learn [86] se ha vuelto una de las más populares librerías de código abierto Machine Learning para Python, etc.

Los algoritmos de aprendizaje automático se organizan en taxonomía, según el resultado deseado del algoritmo. Los tipos de algoritmos comunes incluyen [87]:

- Aprendizaje supervisado --- donde el algoritmo genera una función que asigna las entradas a las salidas deseadas. Una formulación estándar de la tarea de aprendizaje supervisado es el problema de clasificación: se requiere que el alumno aprenda (para aproximar el comportamiento de) una función que mapea un vector en una de varias clases al observar varios ejemplos de entrada-salida de la función [88].
- Aprendizaje no supervisado --- que modela un conjunto de entradas: los ejemplos etiquetados no están disponibles.
- Aprendizaje semi-supervisado --- que combina ejemplos etiquetados y no etiquetados para generar una función o clasificador apropiado.
- Aprendizaje por refuerzo --- donde el algoritmo aprende una política de cómo actuar dada una observación del mundo. Cada acción tiene algún impacto en el medio ambiente, y el entorno proporciona comentarios que guían el algoritmo de aprendizaje.

El rendimiento y el análisis computacional de los algoritmos de aprendizaje automático es una rama de la estadística conocida como teoría del aprendizaje computacional. El aprendizaje automático consiste en diseñar algoritmos que permitan que una computadora aprenda. El aprendizaje no implica necesariamente la conciencia, pero el aprendizaje es una cuestión de encontrar regularidades estadísticas u otros patrones en los datos. Por lo tanto, muchos algoritmos de aprendizaje automático apenas se parecerán a cómo los humanos podrían abordar una tarea de aprendizaje. Sin embargo, los algoritmos de aprendizaje pueden dar una idea de la dificultad relativa de aprender en diferentes entornos.

El campo del Machine Learning es muy extenso, aunque puede ser separarlo en cuatro clases basadas en el objetivo de su aplicación. La división se basa en: clasificación, clustering, regresión y detección de anomalías. Una aplicación real puede situarse en una categoría anteriores o en varias uniendo varios modelos.

a) **Clasificación**

La clasificación es la aplicación del Machine Learning a un conjunto de datos para asignar objetos según sus características a una serie de clases ya definidas. El algoritmo modela los patrones que presentan muestras previamente clasificadas y aprende de dichas muestras a clasificar los nuevos objetos que se le proponen.

Las aplicaciones de la clasificación mediante Machine Learning pueden ser muy diversas, pueden ser detectar si un objeto en una imagen es válido o no para su venta (control de calidad en una fábrica), determinar a quién pertenece una huella dactilar en un sistema de identificación o decirnos si un comentario en una red social está a favor o en contra de nuestra marca junto a muchas otras.

b) **Clustering**

El clustering también conocido como segmentación, se basa en agrupar objetos que contienen características comunes o similares para formar grupos de objetos parecidos.

Una aplicación puede ser la recomendación de venta de productos en tiendas online, en este caso el modelo puede encontrar los productos que son similares a los que se buscan y recomendarlos. También puede ser aplicable a otros modelos de negocio como YouTube o Netflix.

c) **Regresión**

La regresión tiene como objetivo modelar conjuntos de datos para predecir valores continuos, es en cierta manera similar a la clasificación, pero en vez de asignar una clase al objeto propone un número (o varios). Al igual que la clasificación parte de un conjunto de datos previamente numerados para aprender los patrones que sigue este etiquetado.

Algunos de los ejemplos que se encuentran en el mundo empresarial de la regresión pueden ser la predicción de demanda de un producto en determinada hora del día o la cotización de una determinada empresa en bolsa.

d) **Detección de anomalías**

La detección de anomalías realiza un análisis de patrones en los datos para detectar datos extraños o que se salen de un determinado conjunto de valores. Este conjunto de modelos puede ser utilizado en aplicaciones que necesiten de un seguimiento continuo de determinados valores clave que puedan afectar a los procesos de la empresa.

Se puede aplicar este tipo de modelos a encontrar posibles fraudes en movimientos bancarios, detección prematura en fallos de maquinarias o detección de intrusiones en servicios web.

Las técnicas tradicionales de aprendizaje automático y los algoritmos de ingeniería de funciones tienen una capacidad limitada para procesar datos naturales en su forma original. Por el contrario, Deep Learning es más poderoso para resolver problemas de aprendizaje y análisis de datos que se encuentran en grandes conjuntos de datos. De hecho, ayuda a extraer automáticamente representaciones de datos complejos de grandes volúmenes de datos en bruto no supervisados y sin clasificar. Las herramientas que trabajan con Deep Learning están comparadas en [89].

El aprendizaje profundo o Deep Learning es un subconjunto del aprendizaje automático donde los algoritmos se crean y funcionan de manera similar a los del Machine Learning, pero existen numerosas capas de estos algoritmos, cada una de las cuales proporciona una interpretación diferente a los datos de los que se alimenta. Esta red de algoritmos se denomina redes neuronales artificiales, un intento de imitar la función de las redes neuronales humanas presentes en el cerebro.

Además, debido a que el Deep Learning se basa en el aprendizaje jerárquico y la extracción de diferentes niveles de abstracciones de datos complejos, es adecuado para simplificar el análisis de grandes volúmenes de datos, la indexación semántica, el etiquetado de datos, la recuperación de información y tareas discriminatorias como la clasificación y predicción en una representación interna adecuada o vector de características desde el cual el subsistema de aprendizaje, a menudo un clasificador, podría detectar o clasificar patrones en la entrada [65].

1.4.1 Deep learning en Data Streams

Una instancia de datos (x) proviene de algún espacio R^d y puede representarse mediante un vector de longitud d; es decir, $x = [x_1, ..., x_d]$. Supongamos un flujo de datos de instancias $x_t \mid t = 1, ...$ (de longitud desconocida). Tenga en cuenta que se usa el subíndice de un vector para denotar el paso en el tiempo y el subíndice de un valor para indicar un valor particular de un vector (por ejemplo, $x_k \mid 1 < k < d$).

La tarea del aprendizaje del flujo de datos es aprender algún clasificador h para proporcionar una clasificación y_{t+1} para cada x_{t+1}:

$y_{t+1} = h(x_{t+1})$

Esta clasificación es a menudo una sola clase $y \in [1, ..., C]$, pero generalmente al caso de múltiples tareas / multidimensionales de múltiples variables objetivo $y \in N_+^k$, cada una representada como vector $y = [y_1, ..., y_k]$.

Posteriormente a la clasificación (paso de tiempo t), se puede proporcionar la verdadera clasificación de x_t: y_t. Tenga en cuenta que se denota el paso de tiempo t + 1 para el tiempo de prueba y el paso de tiempo t para el tiempo de actualización. En el paso de tiempo t, un alumno de flujo de datos usa cada par (x_t, y_t) como ejemplo de entrenamiento para actualizar su modelo. En el caso totalmente supervisado, la clasificación verdadera y siempre está disponible con posterioridad a la clasificación. En el caso semi-supervisado / parcialmente etiquetado, solo un subconjunto de t = 1, ... tiene etiquetas

verdaderas; por tanto, para algunos x_t no existe una clasificación de verdad fundamental disponible [90].

1.5 Técnicas de Inteligencia Artificial asociadas a StreamData

El procesamiento y análisis de data stream es una característica que se encuentra presente en muchas aplicaciones y sistemas de software de la actualidad. Debido al crecimiento de internet, la automatización de sistemas, el aumento de la conectividad social y el avance en la tecnología, las aplicaciones generan flujos de datos (Data streams) potencialmente infinitos, volátiles y continuos, lo que requiere un procesamiento en tiempo real, simple y rápido.

1.5.1 Técnicas de procesamiento basadas en Minería de Datos

La Minería de Datos permite analizar factores de influencia en determinados procesos, predecir o estimar variables o comportamientos futuros, segmentar o agrupar ítems similares, además de obtener secuencias de eventos que provocan comportamientos específicos, siendo su principal ventaja inferir en comportamientos, modelos, relaciones y estimaciones de los datos, para poder desarrollar predicciones sobre los mismos, sin la necesidad de contar con patrones o reglas preestablecidas, permitiendo tomar decisiones proactivas y basadas en un conocimiento acabado de la información [62].
También se pueden categorizar según el modo de funcionamiento a lote y streaming. La Figura 1.10 muestra estas dos clasificaciones de las técnicas de minería de datos adoptadas en IoT [63].

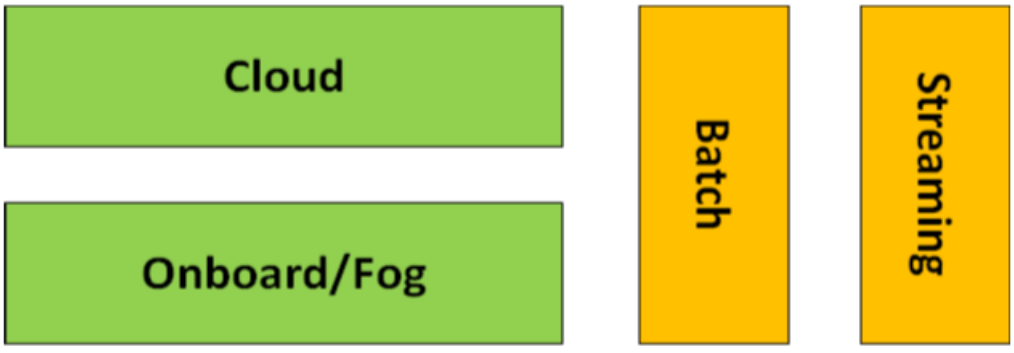


Figura 1.11 Categorización de las técnicas de Minería de Datos para IoT.

- Los métodos de minería de datos integrados tienen la capacidad de ejecutarse en entornos con recursos limitados. Esta permite lo que se ha denominado objetos inteligentes para su realización. Los objetos inteligentes son cosas que son no solo pueden sentir el entorno en el que operan, sino también interpretar eventos y reaccionar a ellos.
- Los métodos de minería de datos basados en la nube están diseñados para la escalabilidad a través de la paralelización y distribución de procesos y conjuntos de datos para grandes volúmenes de datos. Estos métodos son la mejor opción para aplicaciones de IoT que se ejecutan a nivel nacional, o más generalmente en una gran área geográfica. Cuando los datos se recopilan y posiblemente se agregan a partir de varias "cosas", se pueden utilizar para tareas de minería de datos a más largo plazo. Las dos categorías mencionadas anteriormente organizan los métodos de minería de datos aplicados a IoT aplicaciones según el lugar de ejecución. Sin embargo, independientemente de dónde se encuentren los datos se realiza el proceso de minería, hay dos modos de operar estos métodos, a saber, por lotes y streaming.
- Los métodos de minería de datos por lotes operan en datos almacenados, ya que los métodos son inherentemente iterativos. Estos métodos se adaptan a las aplicaciones de IoT que operan con datos históricos en diferentes niveles de granularidad. Por tanto, estos métodos se basan naturalmente en la nube. Sin embargo, en aplicaciones con suficiente capacidad de almacenamiento, los métodos por lotes siguen siendo soluciones válidas. Una gran cantidad de métodos se encuentran bajo esta categoría, incluidos los métodos de clasificación y regresión para la minería de datos predictiva, y métodos de análisis de agrupamiento y enlace/afinidad para la minería de datos descriptivos.
- Los métodos de minería de datos de streaming se aplican a datos en vivo y se adaptan mejor a la aplicación de IoT cuando la velocidad de los datos es alta, y hay una necesidad en tiempo real para actuar proceso de modelado. Los métodos de transmisión se pueden utilizar en la nube, o más típicamente orientados.

1.6 Big Data

El Big Data se refiere, básicamente, a cantidades masivas de datos digitales cuyo procesamiento resulta imposible para los sistemas informáticos convencionales. Este incluye formatos heterogéneos de datos: estructurados, semiestructurados y desestructurados. Los principales algoritmos para el filtrado, clasificación y agrupamiento de datos son mostrados en el Anexo V [64]. Esta complejidad se debe a

las tres principales características de este tipo de datos, las cuales se conocen como “las 3 V del big data” [65]:

- **Volumen**: la cantidad de datos es enorme, una empresa puede llegar a recolectar de petabytes (1 millón de Gb) o exabytes (1 mil millones de Gb) de información.
- **Velocidad**: los datos se producen de manera extremadamente rápida, a cada segundo se están generando grandes cantidades de estos.
- **Variedad**: los datos son complejamente heterogéneos. Tienen distintos formatos, tamaños, tipologías, estructuras y provienen de múltiples fuentes.

Para el procesamiento de datos distribuidos, escalables y tolerantes a fallos se aplica la arquitectura Lambda (λ) mostrada en el Anexo IV junto a Kappa. Ambas se construyen, generalmente, con componentes distribuidos desarrollados sobre la JVM (Java Virtual Machine), por tanto, son soft-real-time systems (orden de segundos). Algunos de las características que la distingue una de otra son:

Lambda:

- Orientada a la analítica de datos tradicional.
- Periódicamente se recogen y procesan grandes volúmenes de datos estáticos (batch) al tiempo que se procesan “on the fly” datos dinámicos (streaming), combinando así volumen y velocidad.

Kappa:

- Orientada a la analítica en tiempo real (soft strict).
- Se evita almacenar los datos y se procesan en cuanto se reciben minimizando el tiempo que el dato está en el pipeline.
- La idea aquí es no recomputar todos los datos en la capa batch, sino hacerlo en la capa streaming y únicamente recomputar si se produce un cambio en la lógica de negocio.

Siguiendo la taxonomía de los algoritmos de clustering convencionales, los algoritmos de agrupamiento de stream data se pueden clasificar en: métodos de partición, métodos basados en densidad, basados en cuadrículas y métodos basados en modelos [66].

• Los algoritmos de particionamiento dividen los datos en k grupos, donde k es especificado por el usuario. La partición se basa en alguna optimización de criterio como Suma de errores cuadrados. En general, estos algoritmos producen agrupaciones esféricas y no manejan valores atípicos. Los

algoritmos de agrupación de data streams relacionados son los algoritmos Stream Framework [67], Clustream [68], SWClustering [69] y StreamKM ++ [70].

• Los algoritmos basados en densidad consideran los clústeres como regiones de alta densidad que están bien separadas por regiones de baja densidad. Pueden descubrir grupos de formas arbitrarias e identificar valores atípicos. El número de conglomerados no se requiere como entrada, los parámetros de entrada se refieren a la definición de la vecindad del objeto y su densidad. Los algoritmos de agrupación de flujos de datos relacionados son DenStream [71], rDenStream [72], C-DenStream [73], SDStream [74], HDDStream [75] y HDenStream [76].

• Los algoritmos basados en cuadrículas son una categoría especial de algoritmos basados en densidad, donde las regiones consisten en celdas de cuadrícula. En particular, el espacio de datos se divide en un número finito de celdas que forman una estructura de cuadrícula en la que se realiza la agrupación. Los algoritmos de agrupación de data streams relacionados son D-Stream [77], DDStream [78], MR-Stream [79] y PKS-Stream [80].

• Los algoritmos basados en modelos intentan ajustar un modelo a los datos, asumiendo que los datos se generan a partir de k distribuciones de probabilidad (típicamente gaussianas). En esta categoría pertenece SWEM [81].

1.7 Conclusiones del capítulo

Existen muchos modelos tecnológicos que facilitan el desarrollo de sistemas que utilizan streams. Los más importantes son los que provienen de las técnicas de Big Data gestados para manejar grandes volúmenes de datos en determinadas condiciones. En el caso de los cortocircuitos estas tecnologías influyen notoriamente en su detección y son en la actualidad los modelos tecnológicos que más utilizan las empresas eléctricas para localizar fallas.

El uso de ontologías y vocabularios semánticos garantiza que las máquinas entiendan el entorno de la red eléctrica sin la necesidad de que un ser humano este pendiente del Sistema. Las ontologías manejan el razonamiento basado en casos que es capaz de asegurar la interpretación y la detección de los cortocircuitos o las averías. A nuestro juico los elementos menos desarrollados son los sistemas de consulta que sin bien están desarrollados a nivel de artículo científico no constituyen herramientas para el desarrollo de sistemas reales orientados al uso de usuarios reales.

CAPÍTULO 2. DISEÑO DE UN SISTEMA PARA LA DETECCIÓN DE CORTOCIRCUITOS MEDIANTE TÉCNICAS DE DATA STREAM

Para representar IoT data streams, se necesitan conceptos que figuren dispositivos, ubicación, tiempo, unidades de cantidad, valores y streams. En la descripción de dispositivos, hay algunos modelos para representar sensores y sus observaciones. El modelo más representativo es la ontología SSN que describe los sensores con sus propiedades, sistemas, despliegues, estímulos y observaciones [91]. La ontología SOSA es un núcleo ligero para SSN que proporciona conceptos para sensores, valores de observación y características de interés [43]. IoT-Lite es otro modelo ligero para conceptos de IoT con el objetivo de una anotación, un procesamiento y un tiempo de consulta semánticos rápidos. IoT-Lite se inspiró en el modelo de referencia de IoT-A [92]. El enfoque de IoT-Lite, SOSA y SSN está más en los dispositivos de detección y es apropiado para el descubrimiento de sensores, ya que carecen de conceptos específicos para la anotación y agregación de streams.

Hay algunos modelos de ubicación, como Geo[6] que ayudan a buscar dispositivos IoT. Geo es un modelo popular que representa datos de ubicación en RDF. La ontología ofrece solo algunos términos básicos simples que se pueden usar en RDF cuando existe la necesidad de describir latitudes, longitudes y altitudes. El uso de RDF como portador de latitud, longitud y altitud simplifica la capacidad de mezcla de datos entre dominios, así como la descripción de entidades que están ubicadas en el mapa (por ejemplo, realizar consultas geoespaciales para sensores, implementaciones, plataformas o sistemas). GeoSPARQL es un estándar para la representación y consulta de datos geoespaciales enlazados para la Web Semántica del Consorcio Geoespacial Abierto (OGC) [93].

La Ontología de Tiempo[7] es un modelo semántico bien conocido y ampliamente utilizado para representar el tiempo. Tiene un vocabulario para representar información sobre relaciones topológicas (ordenamiento), duración y posición temporal (es decir, información de fecha y hora). El tiempo se puede expresar usando un reloj convencional, tiempo Unix, tiempo geológico y otros sistemas de referencia. Las ontologías de tiempo se han utilizado para anotar transmisiones y han inspirado la consulta de datos de transmisión. También hay algunas ontologías para proporcionar cantidades,

[6] https://www.w3.org/2003/01/geo/

[7] https://www.w3.org/TR/2017/REC-owl-time-20171019/

unidades, dimensiones y valores. La ontología QU[8] es una de las ontologías más conocidas en este campo. La ontología QU ha sido desarrollada para soportar diferentes usuarios del Lenguaje de Modelado de Sistemas (SysML).

Otro aspecto importante de la anotación de data stream es la calidad de la información (QoI) porque los datos defectuosos pueden tener consecuencias costosas [94]. Cuando se habla de calidad de la información, las categorías o métricas son importantes para describir los detalles. Hay cinco métricas comunes: integridad, corrección, concordancia, vigencia, plausibilidad [95] y seguridad [96].

Las ontologías mencionadas anteriormente podrían ayudar en la anotación de data streams, pero no abordan todos los conceptos necesarios para este tipo de datos y se pierden conceptos esenciales, como la agregación de streams que podrían aprovechar el tiempo de procesamiento al consultar data streams. Hay pocas ontologías que representan los datos en stream. Un representante es Stream Annotation Ontology (SAO). SAO se ha construido en la parte superior de algunas ontologías bien conocidas para presentar IoT data streams: TimeLine [97], PROV-O [98], SSN [91] y Event Ontology [97]. Los conceptos StreamData, StreamEvent, StreamAnalysis, Observación, Sensor y Segmento permiten que esta ontología describa conceptos temporales con precisión. Con la clase StreamData, SAO puede proporcionar un flujo de datos como un punto o segmento temporal y describe la salida de la observación como un evento con la clase StreamEvent [61].

2.1 Ontología IoT-Stream

El diseño de IoT-Stream se basa en un conjunto de principios, siempre teniendo en cuenta que nuestro objetivo es una ontología ligera que amplía SOSA para proporcionar conceptos de anotación de stream.

2.1.1 Visión y Diseño

Para reflejar esto en el diseño de la ontología, es necesario definir los conceptos de análisis y derivación de stream. Ya que se espera que los data streams de IoT produzcan observaciones en la escala de Big Data. Es necesario mantener un enfoque ligero para definir los conceptos de observaciones del stream y separarlos de las descripciones relacionadas con IoT stream en su conjunto. Para que una ontología sea eficaz para su adopción, su desarrollo debe tener una base con las mejores

[8] https://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu-rec20.html

prácticas bien establecidas, como las definidas en la guía de creación de ontologías [99]. La primera consideración es definir el dominio y alcance de la ontología. En este caso gira en torno al concepto de data streams producidos por fuentes de IoT, con un enfoque en conceptos que apoyan el análisis de datos, la detección de eventos y la procedencia. La siguiente consideración es adoptar los conceptos existentes de otras ontologías que pueden enriquecer el modelo con metadatos útiles para describir el concepto principal; en este caso, IoT Stream. Aquí, los conceptos relacionados con el espacio, el tiempo, el tema, la asociación de dispositivos, la exposición del servicio y la calidad de los datos son muy relevantes, los cuales están disponibles y bien establecidos en la comunidad de IoT. La tercera consideración es qué términos deben capturarse en la ontología. Con respecto a los streams y stream data, el instante o intervalo de tiempo en el que se realizó la observación es importante. En cuanto al valor, debe ser simple, pero flexible, ya que los formatos de observación pueden variar entre sistemas. Para los términos relacionados con el análisis de datos aplicados a los streams, se necesitan términos como los métodos y parámetros utilizados en una técnica en particular. Un resultado importante del análisis son los eventos y alertas que se detectan en los streams y, por lo tanto, se deben capturar las etiquetas y los aspectos temporales. La cuarta consideración es adoptar una jerarquía para las clases definidas. La ontología SSN proporciona un concepto para las observaciones y cumple los requisitos básicos definidos anteriormente, por lo que se ha adoptado y ampliado para adaptarse a la naturaleza de las observaciones de streams. Otro aspecto a tener en cuenta es que la ontología debe reflejar principios adoptados en ontologías que han sido destacados por organismos de estandarización para la aplicación de buenas prácticas, como las Good Ontologies del W3C[9]. Estos principios se relacionan con la calidad de la documentación asociada de la ontología, y el hecho de que sea un identificador de recursos internacionalizado (IRI) está desreferenciado. También debe demostrar la adopción por parte de los productores de datos para la anotación, y debe estar respaldada por las herramientas existentes. Una vez que se aplican todos estos aspectos, se puede formular un modelo de información para IoT data streams.

2.1.2 Modelo de información

El modelo de información principal se enfoca en modelar las observaciones de los streams, su análisis y los eventos que se detectan a partir de ella, que se capturan en cuatro clases. Estas clases reflejan los

[9] http://www.w3.org/wiki/Good_Ontologies

conceptos de un *IotStream*, *StreamObservation*, *Analytics* y *Event*. Como se muestra en la Figura 2.1, la clase central a la que las otras clases se vinculan directamente es *IotStream*. Esta abstracción representa unos streams que se originan en una fuente de datos de IoT. Tiene propiedades de anotación capturan la vida útil del streams que se utilizaría principalmente como referencia en lugar del consumo real de una aplicación. Las propiedades de esta anotación son *streamStart* y *streamEnd*. Como se mencionó anteriormente, la visión de un *IotStream* es que, al igual que las vías fluviales, pueden ramificarse de otras corrientes y, por lo tanto, derivarse de otras *IotStream*. Esto puede ser el resultado de alguna forma de procesamiento del *IoTStream* que se deriva de, como el filtrado, el remuestreo o agregación.

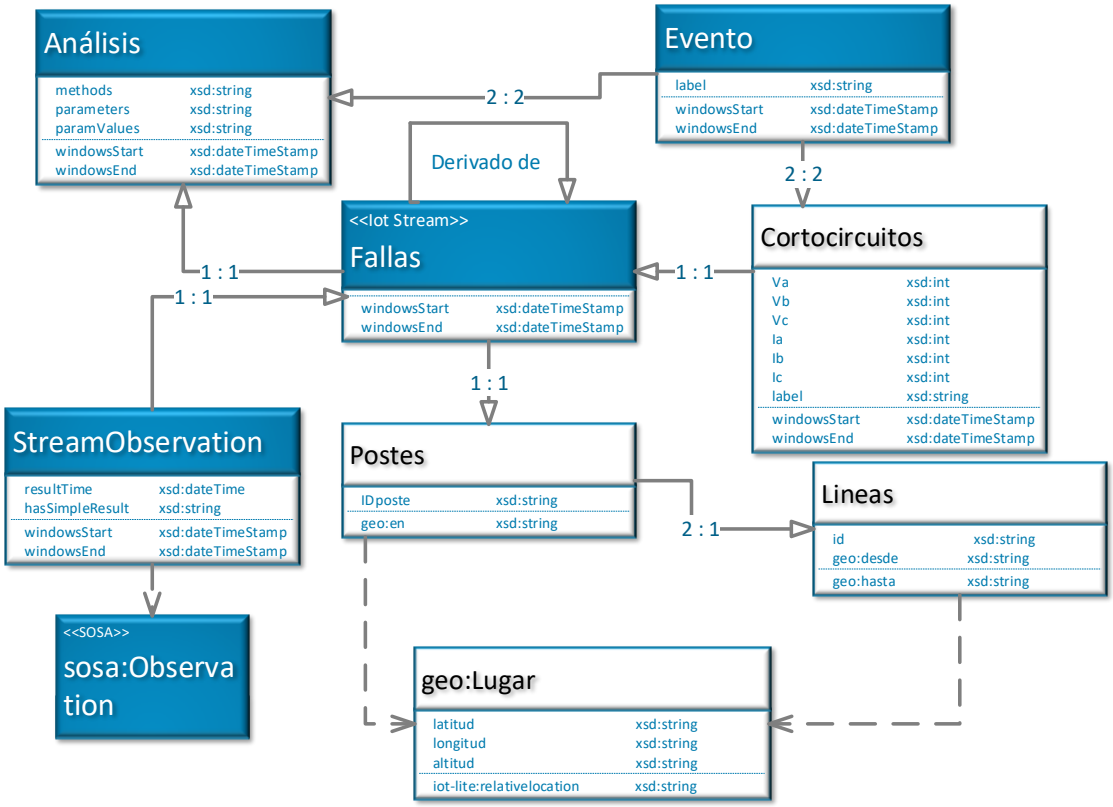


Figura 2.1 IoTStream clases y propiedades.

El valor de la observación de un streams puede tomar varias formas. El primero es atómico, como punto de datos. En segundo lugar, podría tener una representación masiva como un vector de puntos de datos. StreamObservation también contiene una marca de tiempo instantánea que mantiene un registro de cuándo se capturó la observación. Al considerar la reutilización de otras ontologías populares, la

ontología SOSA proporciona una clase que cumple con los requisitos para capturar una observación de sensor, que es la clase *sosa:Observation* y sus propiedades de tipo de datos, *sosa:hasSimpleResult* y *sosa:resultTime*. Otro requisito es poder capturar el aspecto temporal de una observación como intervalo o ventana, *sosa:Observation* proporciona la propiedad de tipo de datos *sosa:fenomenTime* para este propósito, y se vincula a la clase *time:TemporalEntity* de la popular Time Ontology. La clase *sosa:Observation* se ha ampliado con una subclase, StreamObservation, para incluir propiedades de tipo de datos directas la cual representan ventanas temporales. Estos se capturan en las propiedades de datos *windowStart* y *windowEnd*, que representan el inicio y el final de la ventana, respectivamente. Otra consideración importante para *StreamObservations* es segregarlo del resto de los metadatos, ya que el número de instancias creadas sería significativamente mayor en proporción con respecto al número de IotStreams y, por esta razón, la extensión de la instanciación de *sosa:Observation* se mantiene intencionalmente al mínimo.

StreamObservation que pertenecen a IotStreams puede ser el resultado de las lecturas del sensor o el resultado de un proceso de análisis. En el caso de que IotStream se analice mediante un proceso de análisis de datos, la clase *Analytics* captura los métodos de las técnicas de análisis de datos aplicadas en IotStream. Puede ser un solo proceso o una cascada de procesos y, por lo tanto, se representa como una cadena de vectores con las propiedades de datos *paramValues*, *methods* y *parameters*. La propiedad de datos *methods* capturan los diferentes métodos o algoritmos con los que se ha analizado el stream. La propiedad de datos *parameters* establecidos para estos métodos también se capturan como una cadena de vectores, por lo que el primer elemento en el vector de métodos se corresponde con el primer elemento en el vector de parámetros. Vale la pena señalar que los métodos pueden establecer múltiples parámetros, por lo que el elemento correspondiente en la propiedad de los datos de los parámetros puede ser una matriz de parámetros en sí mismo también. Para cada parámetro, los valores que se establecen se capturan en la propiedad de datos *paramValues*. El proceso *Analytics* que se aplica a un IotStream puede posiblemente estar activo durante una ventana temporal con la vida útil de un IotStream. Por lo tanto, las propiedades de datos *windowStart* y *windowEnd* se utilizan para este caso. La clase *Analytics* también se puede utilizar exclusivamente para definir el proceso de análisis de datos que se utiliza para generar *Events* que a través de *detectedFrom* manifiestan un IotStream, que sería aplicable en casos como la clasificación o el agrupamiento. *Event* contiene propiedades que capturan la etiqueta de propiedad de datos que se utiliza para describir el *Event*, y el intervalo temporal del evento

también es relevante. Esta puede ser información útil para que los científicos de datos comprendan cómo se generó el evento.

2.1.3 Modelos vinculados

Como se mencionó anteriormente con respecto a la reutilización de ontologías, el modelo de información adopta varios conceptos que se consideran atributos centrales para proporcionar un contexto del mundo real al IoT stream.

El primero se relaciona con los atributos espaciales de IotStream. La ontología geográfica del W3C proporciona un conjunto de conceptos básicos que representan la ubicación de una entidad. El concepto principal de interés es el *geo:Point* que contiene propiedades geoespaciales (latitud, longitud y altitud). La ontología IoT-lite [100] amplía las propiedades para incluir la ubicación relativa y la altitud relativa. Para mantener el contexto histórico de StreamObservations, especialmente en el caso de la movilidad, un *geo:Point* se puede vincular a cada StreamObservation. IoTStream también se puede asociar con un área de cobertura definida donde también es relevante. El concepto *iot-lite:Coverage* puede usarse para definiciones de cobertura simples, y GeoSPARQL [93], que es una ontología bien establecida para atributos espaciales, puede usarse para definiciones de áreas más complejas.

El siguiente subconjunto de conceptos adoptados se relaciona con la fuente generadora de IotStream, los fenómenos y la medición de sus observaciones. Como los stream en el mundo real son generados por sensores, el concepto de sensor SOSA [43] está vinculado. A través de las propiedades de objeto definidas por IoT-Lite, los conceptos *qu:QuantityKind* y *qu:Unit* de la ontología QU también están vinculados.

Aunque es el sensor el que genera el IoT stream, a través de Internet, los datos stream suelen ser proporcionados por un servicio de capa de aplicación TCP/IP. IoT-Lite proporciona *iot-lite:Service class* que contiene campos relacionados con la dirección del punto final del servicio, el tipo de interfaz y el enlace a la descripción de la interfaz, que proporciona detalles sobre cómo interactuar con el servicio.

Finalmente, a lo largo de la vida útil de un IotStream, la calidad de las observaciones del stream puede cambiar con el tiempo. Para el análisis de datos, el conocimiento de la calidad es muy importante para que las medidas de adaptación se puedan aplicar cuando sea necesario. La ontología Calidad de la

Información (QoI) proporciona el concepto *qoi:Quality* que tiene subclases que se enfocan en un aspecto particular de calidad, como qoi:Timeliness y qoi:Completeness de las observaciones. La Figura 2.2 ilustra cómo IoT-Stream está vinculado a estos conceptos externos, y la Tabla 2.1 enumera los espacios de nombres de las ontologías vinculadas y sus prefijos preferidos.

Prefijo	Namespace
iot-lite	http://purl.oclc.org/NET/UNIS/fiware/iot-lite#
iot-stream	http://purl.org/iot/ontology/iot-stream#
owl	http://www.w3.org/2002/07/owl#
qoi	https://w3id.org/iot/qoi#
qu	http://purl.oclc.org/NET/ssnx/qu/qu#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
Rdfs	http://www.w3.org/2000/01/rdf-schema#
Sosa	http://www.w3.org/ns/sosa/
wgs84_pos	http://www.w3.org/2003/01/geo/wgs84_pos#
Xml	http://www.w3.org/XML/1998/namespace
Xsd	http://www.w3.org/2001/XMLSchema#

Tabla 2.1 Prefijos y namespaces de las ontologías vinculadas.

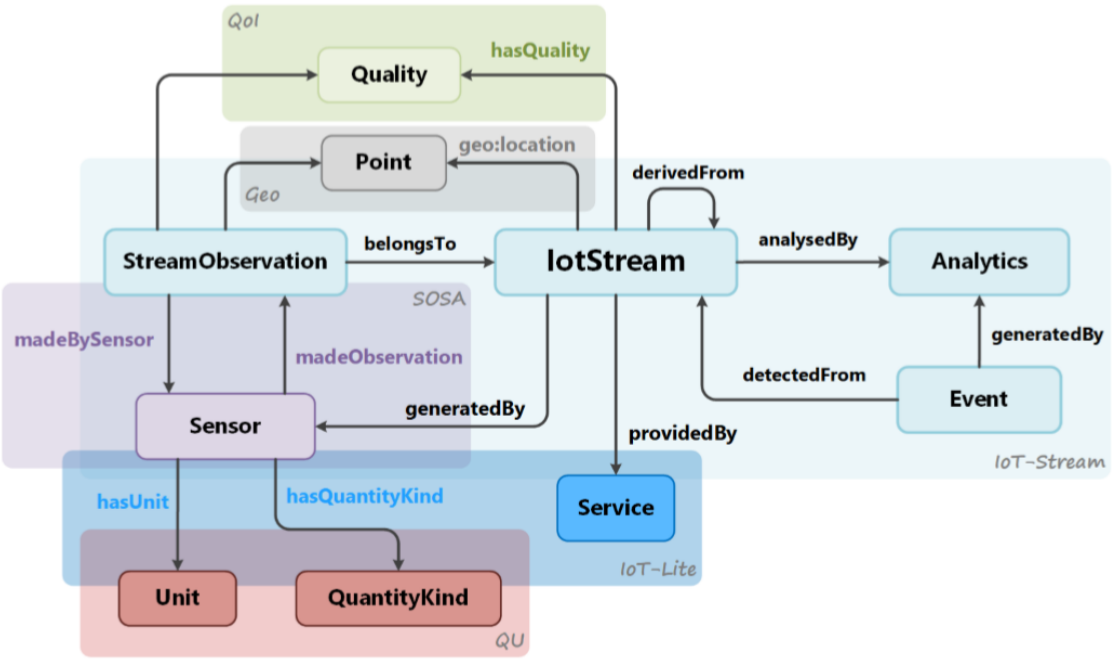


Figura 2.2 IoTStream enlazado con las ontologías principales.

2.1.4 Navegación y consulta de modelos

Es importante considerar cómo se debe navegar o consultar un grafo basado en IoT-Stream. El concepto de IoTStream sirve como un nodo raíz en un grafo que enlaza directamente con los metadatos más importantes en el contexto de las consultas relacionadas con los datos de IoT. La Figura 2.3 ilustra una instancia de IotStream. Aquí, el IoTStream se genera mediante *sosa:Sensor* que *sosa:madeObservations* mide el *qu:QuantityKind* de corriente con un *qu:hasUnit v*. *StreamObservations* que pertenece a un IotStream se anotan atómicamente. El IoTStream *geo:location* en un *geo:Point* da la ubicación del stream donde se evidencia en *iot-lite:relativeLocations*. Gracias a la ubicación del stream se puede determinar en qué poste y línea ocurre el cortocircuito. IotStream se proporciona mediante un *iot-lite:Service* con un *iot-lite:endpoint* con un RESTful *iot-lite:interfaceType*. Información acerca de la *qoi:Frequency* de IotStream está vinculada a través de un proceso de monitoreo. Un segundo IotStream se deriva de él a través de un proceso de Analytics que implica filtrado y agregación. Se detecta un Event que indica "Cortocircuito Trifásico" desde el segundo IotStream.

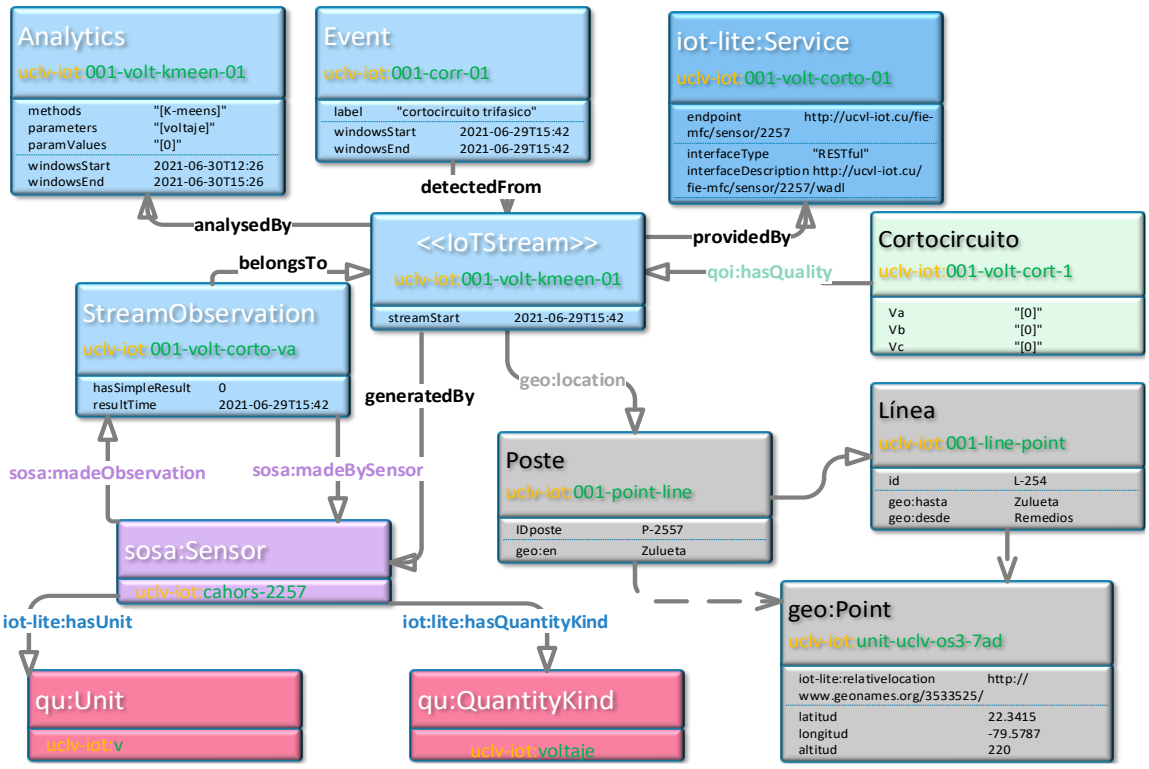


Figura 2.3 Instanciación de IoTStream.

2.1.5 Métricas y documentación de ontología

La ontología está basada en los vocabularios descritos en la Tabla 2.2. Las clases principales contienen la red Eléctrica, las Fallas, los tipos de línea, además también se presentan las clases Sensor, ObservableProperty con las variables de energía y los elementos que permiten la detección de las coordenadas geográficas. Las propiedades del objeto en la ontología se unen de la siguiente forma: class/individuals. Ellos son los elementos más importantes en la ontología y actúan para gestionar los cortocircuitos en la red. Se listan las propiedades del objeto principales en la ontología propuesta en tabla.

Object Property	Domain	Range	Description
sosa:madeObservation	sosa:Actuation red:Operaciones	sosa:Sensor	Permite la relación entre las operaciones de control de cortocircuitos y el Sensor.
sosa:resultTime	sosa:Actuation sosa:Observation	time:Time	Relaciona las operaciones y la observación de los cortocircuitos durante un tiempo determinada a través de las variables observadas.
sosa:observes	sosa:ObservableProperty sosa:Actuation red:Lineas red:Datos	sosa:Sensor Sosa:Actuation	El sensor observa las propiedades observables (variables) en las líneas. Estas variables son datos observados en tiempo real o mediante datos históricos. De estas variables depende la actuación o la toma de decisiones.
sosa:observedProperty	sosa:Observation	sosa:ObservableProperty	Relaciona las observaciones con las propiedades observables (variables).
Sosa:madeObservation	Sensor	StreamObservation	Enlaza los sensores con las observaciones de los streams.
sosa:hasFeatureOfInterest	sosa:Observation sosa:Actuation	sosa: FeatureOfInterest	La observación y la actuaction tienen objetos específicos en un sistema eléctrico, estas son las líneas, los postes, los transformadores, los generadores, las cargas, los trasformadores, etc.
sosa:hosts	sosa:Platform	sosa:Sensor	Todos los sensores están hosteados en una plataforma de software libre llamada Sofía
sosa:madeByActuator	sosa:Actuation red:Datos red:Operaciones	sosa:Actuator red:Posiciones red:Protecciones	La Actuation, los Datos, las Operaciones, la Red eléctrica y las fallas están relacionadas con el

	red:RedElectrica red:Fallas	red:Operaciones sosa:Actuation	actuador el que se relaciona con las posiciones, las protecciones, las operaciones y la actuación.
sosa:madeBySensor	sosa:Observation	sosa:ObservableProperty red:RedElectrica	Relación entre las observaciones, las propiedades observables y toda la red eléctrica.
sosa:wasOriginatedBy	ssn:Stimulus	sosa:Observation	Todo cortocircuito está relacionado con una observación.
Iot-stream:derivedFrom	IotStream	IotStream	Relaciones entre dos streams donde uno es generado por otro.
Iot-stream:detectedFrom	Event	IotStream	Los eventos que afectan la red eléctrica son detectados stream.
Iot-stream:analizedBy	IotStream	Analitycs	Cada stream tiene un servicio de análisis y un metido de Inteligencia Artificial que lo maneja.
Iot-stream:providedBy	IotStream	Iot-lite:Service	Cada stream está incluido en un servicio que posee su interface y su SPARQL endpoint.
qoi:hasQuaility	IotStream	qoi:frequency	Determina la frecuencia con que se actualiza el sistema.
geo:en	IotStream	Lugar y Falla	Enlaza el lugar y la falla.

Tabla 2.2 Propiedades del objeto, dominio, rango, y la descripción de propiedades del objeto principal.

Reglas y condiciones:

- Postes(?p) ^ core:Va(?p, 0) ^ Vb(?p, 0) ^ Vc(?p, 0) ^ core:Ia(?p, 0) ^ Ic(?p, ?ic2) ^ Ib(?p, ?ib) ^ swrlb:lessThanOrEqual(?ic2, 0) ^ swrlb:greaterThanOrEqual(?ib, 0) -> BiFaseFailure(?p)
- (Ib some xsd:int[> "1"^^xsd:int]) and (Ic some xsd:int[> "1"^^xsd:int]) and (Ia some xsd:int[> "1"^^xsd:int]) and (Vb value 0) and (Vc value 0) and (Va value 0)

La primera regla es para cuando se cumpla que un cortocircuito es del tipo bifase, este reune ciertas condiciones y de cumplirse todas, estamos en presencia de un cortocircuito bifásico. El próximo punto es una condición el cual pertenece un cortocircuito trifásico ya que dice que si todos los voltajes están en 0 y las Corrientes no son 0.

El modelo de global para IoTStream tiene 25 clases, 18 object properties, 19 data properties, 316 axiomas, 228 axiomas lógicos y 88 axiomas declarados, 13 subclases, 26 individual, 4 clases equivalentes, un inverseObjectProperties, 22 ObjectPropertyDomain, 36 DataPropertyDomain, 57

ClassAssertion, 27 ObjectPropertyAssertion y 31 DataPrepertyAssertion. Las clases, data property y object property son mostrados en la Figura 2.4.

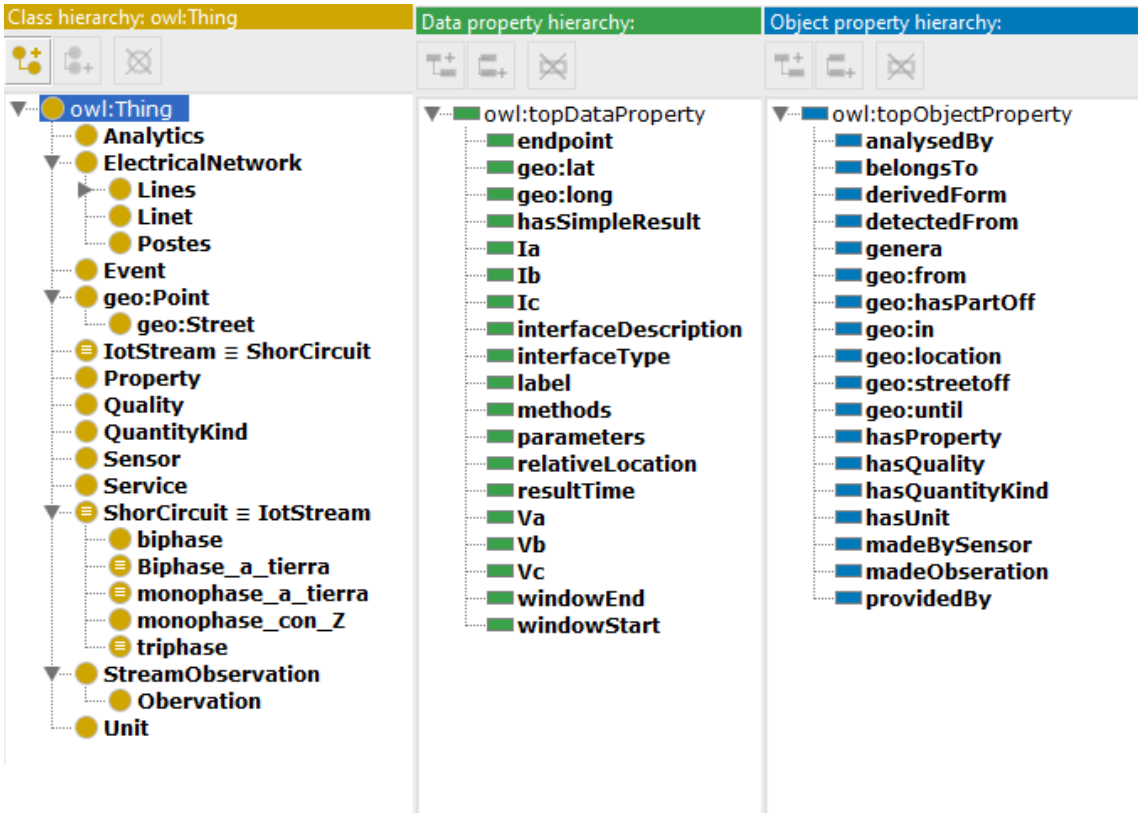


Figura 2.4 Clases, data property y object property de la ontología.

2.2 Arquitectura del Sistema

Al definir una nueva ontología para IoT, es fundamental demostrar cómo se puede utilizar para sistemas orientados a IoT. Esta sección define qué entidades del sistema se necesitan y se pueden adoptar enfoques para explotar IoT-Stream con respecto a los requisitos relacionados con la anotación, la publicación, la persistencia, la consulta y la suscripción a IoTStreams.

2.2.1 Entidades del Sistema

Para que un sistema admita la adopción de IoT-Stream, las entidades esenciales del sistema necesarias, como se muestra en la Figura 2.5, serían:

• **Registro**: principalmente responsable de almacenar información sobre un IotStream en un triple store y exponer un SPARQL endpoint para manejar consultas. También podría ser empleado por un productor de IotStream para almacenar *StreamObservations*.

• **Productor**: responsable de registrar *IotStreams* y publicar las *StreamObservations* generadas a partir de sus sensores. Si es capaz de almacenar y exponer *StreamObservations*, puede actuar como el servicio de IoT.

• **Consumidor**: una aplicación o servicio que descubre IoTStreams a través del registro y consume *StreamObservations* utilizando el servicio de IoT designado. En el contexto del análisis de datos, el consumidor puede absorber datos analizados para inteligencia empresarial o preprocesamiento de *StreamObservations* (como agregación o filtrado).

• **Broker**: una alternativa a las *StreamObservations* persistentes, en la que los Consumidores y los Servicios pueden suscribirse a *StreamObservations* en tiempo real publicadas por un Productor para el corredor de datos de stream. En este caso, la persistencia depende del consumidor.

• **Servicio de análisis**: empleado por un consumidor o productor para absorber o generar IoTStreams analizados, mediante la aplicación de técnicas específicas de análisis de datos con un método específico o un conjunto de métodos. Esto podría ser parte del sistema interno del consumidor o un microservicio externo que se enfoca en un tipo particular de análisis.

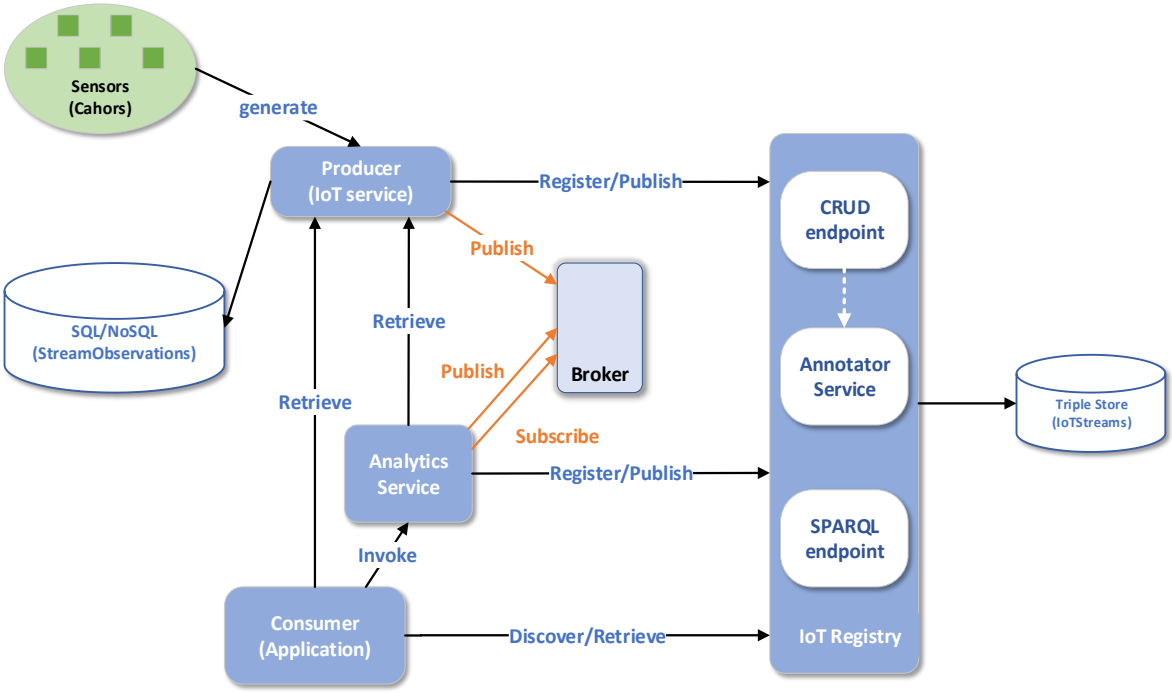


Figura 2.5 Interacciones y entidades del sistema para IoT Stream.

2.2.2 Almacenamiento y búsqueda.

Los almacenes triple store son adecuados para almacenar información finita sobre las novedades, pero no los datos de series temporales que estén asociados con ellos. En el caso de los datos de la serie temporal, a medida que se acumulan más datos para adaptar un conjunto, el triple store comienza a luchar para responder a las consultas en períodos de tiempo no razonables. Si se requiere una anotación atómica, entonces un conjunto de datos *StreamObservation* debe separarse del resto de los metadatos, excepto su enlace al IoTStream al que pertenece. En este caso, se puede utilizar una consulta SPARQL para descubrir IoTStreams y, a su vez, recuperar observaciones de otro conjunto de datos utilizando *iot-lite:endpoint*. Si la recuperación de *StreamObservation* se realiza sin un SPARQL endpoint, también se puede recuperar *iot-lite:interfaceDescription* para saber qué parámetros pasar para recuperar una observación instantánea o un conjunto de observaciones dentro de una ventana definida.

Otro aspecto a considerar es la variabilidad de los metadatos asociados con un IotStream, como la ubicación y la calidad. En el caso de la ubicación, si el sensor que generador IotStream está conectado a una entidad móvil, entonces la información de ubicación debería ser capturada y vinculada a cada *StreamObservation*. Una consulta en este caso debería incluir una verificación del estado de movilidad

del sensor usando *iot-lite:isMobile*. El propio IotStream solo tendría la información de ubicación actual vinculada a él. Para la calidad de la información, métricas como *qoi:Timeliness* puede cambiar durante la vida útil de un IotStream, por lo que experimenta problemas de computación o conectividad, ya sea internamente, es decir, en el dispositivo o externamente causados por un nodo intermediario como una puerta de enlace. En estos casos, sería necesario almacenar nuevas instancias de QoI con cada *StreamObservation* o vincularlas a cada una.

2.3 Diseño del modelo de comunicación

MQTT son las siglas MQ Telemetry Transport, aunque en primer lugar fue conocido como Message Queuing Telemetry Transport. Es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue**.** Está basado en TCP/IP como base para la comunicación.

Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos (CPU, RAM, …). La arquitectura de MQTT sigue una **topología de estrella**, con un nodo central que hace de servidor o "broker" con una capacidad de hasta 10000 clientes. El broker es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del broker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones. Es un servicio de mensajería push con el modelo de publicador/suscriptor (pub-sub). En este tipo de arquitecturas los clientes (publicadores o suscriptores) intercambia datos con un servidor centralizado o broker. Para organizar las colas de envío y recepción de datos estas se organizan por asuntos o topics. Cada cliente puede publicar un mensaje en un determinado topic o recibir información de los topics a los que se encuentra suscripto, por lo que la comunicación puede ser *uno a uno*, o de *uno a muchos*.

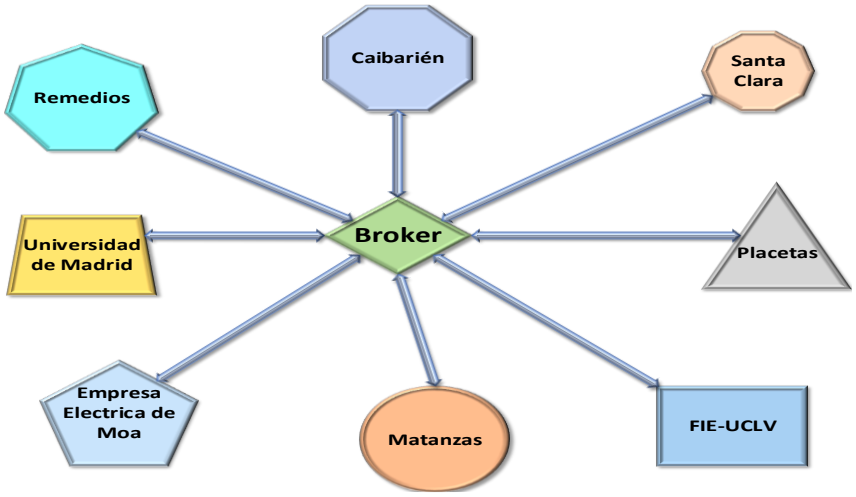


Figura 2.6 Ejemplo de comunicación a través del protocolo MQTT.

La figura anterior (Figura 2.6) muestra muy simplificado cómo funciona el protocolo MQTT el cual utiliza un bróker que es el encargado de enviarte la información correspondiente según uno se halla subscrito a sus servicios. Estos topics están estructurados jerárquicamente lo que permite, en nuestro caso que si la FIE-UCLV quiere recibir los datos de todos los sensores que se encuentran en Zulueta, no tiene que suscribirse a cada sensor, sino que solo se subscribe en Zulueta.

2.4 Herramientas de análisis de datos

Un sistema que consume IoT data streams necesita emplear algún tipo de análisis de datos para manejar el grado de volumen, velocidad, intermitencia, irregularidad y dimensionalidad que los acompañan. Las bibliotecas y los frameworks para lenguajes de programación populares han permitido la creación de herramientas que manejan datos según su naturaleza y los conocimientos esperados que se obtengan. Dependiendo de la aplicación, las herramientas involucrarían alguna forma de preprocesamiento, aprendizaje automático o correlación. El resultado de tales técnicas se puede alimentar para enriquecer un gráfico de conocimiento semántico. El servicio web Recepción de Conocimiento (RC) permite al consumidor experimentar por primera vez, las fuentes de datos de IoT remotas con diferentes cascadas de métodos para estudiar cuál funciona mejor para ellos. Al exponer una interfaz RESTful, RC puede consultar flujos de datos desde un almacén data stream de IoT al aceptar una consulta SPARQL con un formato predefinido para las variables de salida. A su vez, el servicio generará un nuevo data stream basado en los métodos seleccionados y sus parámetros correspondientes. Las nuevas

StreamObservations luego se anotan y se vinculan a un nuevo *IotStream*, con los detalles de *Analytics* empleados, y luego se envían de vuelta al *Consumidor*. La Figura 2.7 ilustra el proceso.

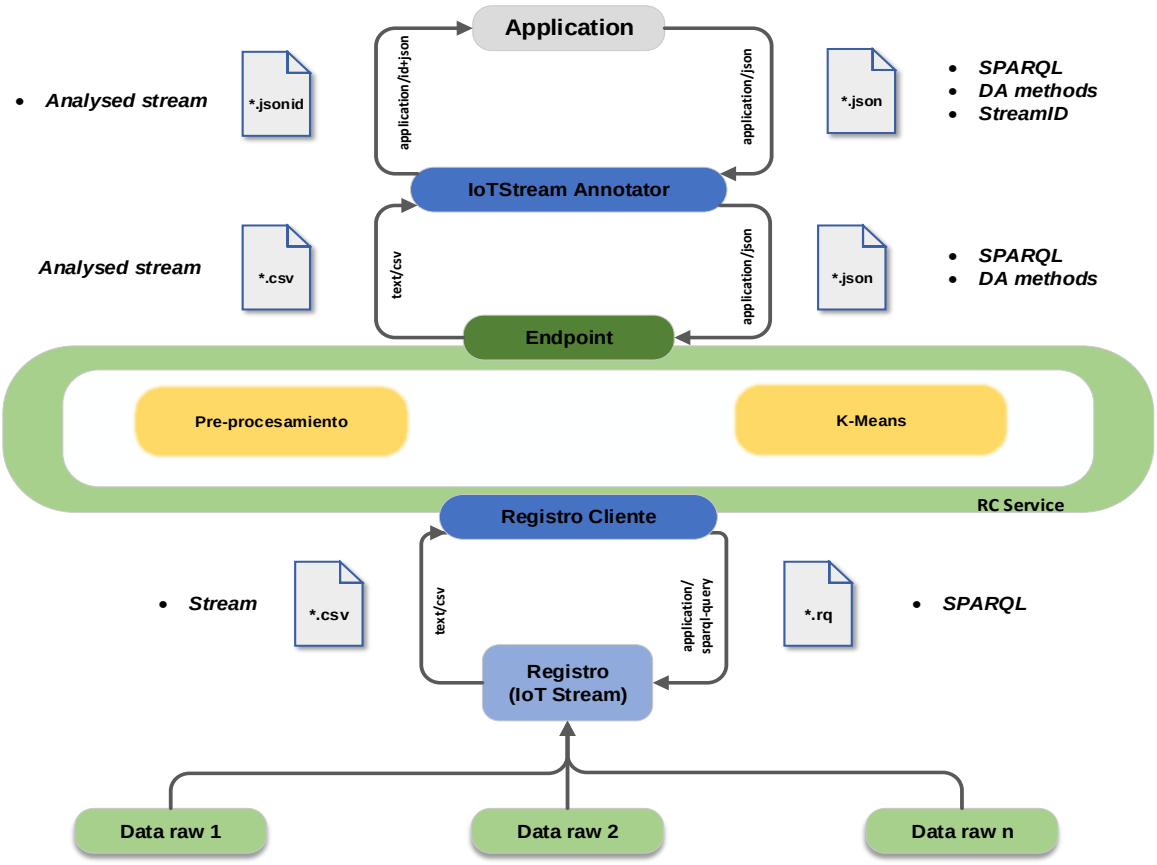


Figura 2.7 Análisis del IoTStream usando el servicio RC.

2.5 Crawling y motores de búsqueda para IoT Data Stream.

El framework IoTCrawler proporciona un rastreador y un motor de búsqueda para descubrir fuentes multidominio de IoT data stream. El rastreador o crawler extrae metadatos de las fuentes de datos y los empuja a través de una Capa de Adaptación que anota semánticamente los metadatos como instancias de IotStream y se almacena en un repositorio de metadatos RDF. Los metadatos capturados aquí incluyen la identificación del IotStream descubierto y el *sosa:Sensor* que lo genera, y el *qu:QuantityKind* que se mide y el *qu:Unit* que se usa para medir. La información de geolocalización, si se proporciona, tendrá *geo:Point* que contiene información absoluta y/o *iot-lite:relativeLocation*. El objetivo del motor de búsqueda es proporcionar a los consumidores resultados que incluyan

información sobre cómo llegar e interactuar con IoT data stream *iot-lite:Service*. El sistema adopta un intermediario de datos desde FIWARE[10], que emplea la interfaz programada de aplicación NGSI-LD [101]. Esta interfaz utiliza un mismo modelo para encapsular modelos de datos específicos para una plataforma particular, en este caso, IoT-Stream. Los metadatos recopilados se distribuyen entre varios brokers, conocido como Repositorio de Metadatos Distribuidos (MDR). El framework emplea componentes de procesamiento que descubren y analizan los data streams del MDR distribuido. Uno de los componentes, el módulo de Enriquecimiento Semántico, aplica un análisis de datos para evaluar el *qoi:Quality*, y se devuelve al MDR, y se adjunta al IotStream correspondiente. El módulo también aloja extractores de patrones que buscan patrones en los stream data para un dominio específico. Ciertos patrones que se detectan en IotStream y se traducirán en *Event*, que luego se enviarán al MDR. Son estos *Events* los que servirán como palabras clave para que los consumidores las utilicen para buscar el data stream de interés. Para habilitar esto, los componentes relacionados con el motor de búsqueda indexarán y clasificarán IoTStreams, *sosa:Sensors*, *qu:QuantityKind* y *Events* según la ubicación y las métricas de QoI. Esto permitirá a los consumidores realizar búsquedas instantáneas o suscribirse a actualizaciones de servicios de data stream según sus preferencias. La Figura 2.8 muestra los componentes arquitectónicos en el framework IoTCrawler y la anotación de transmisiones utilizando IoT-Stream.

[10] https://www.fiware.org/

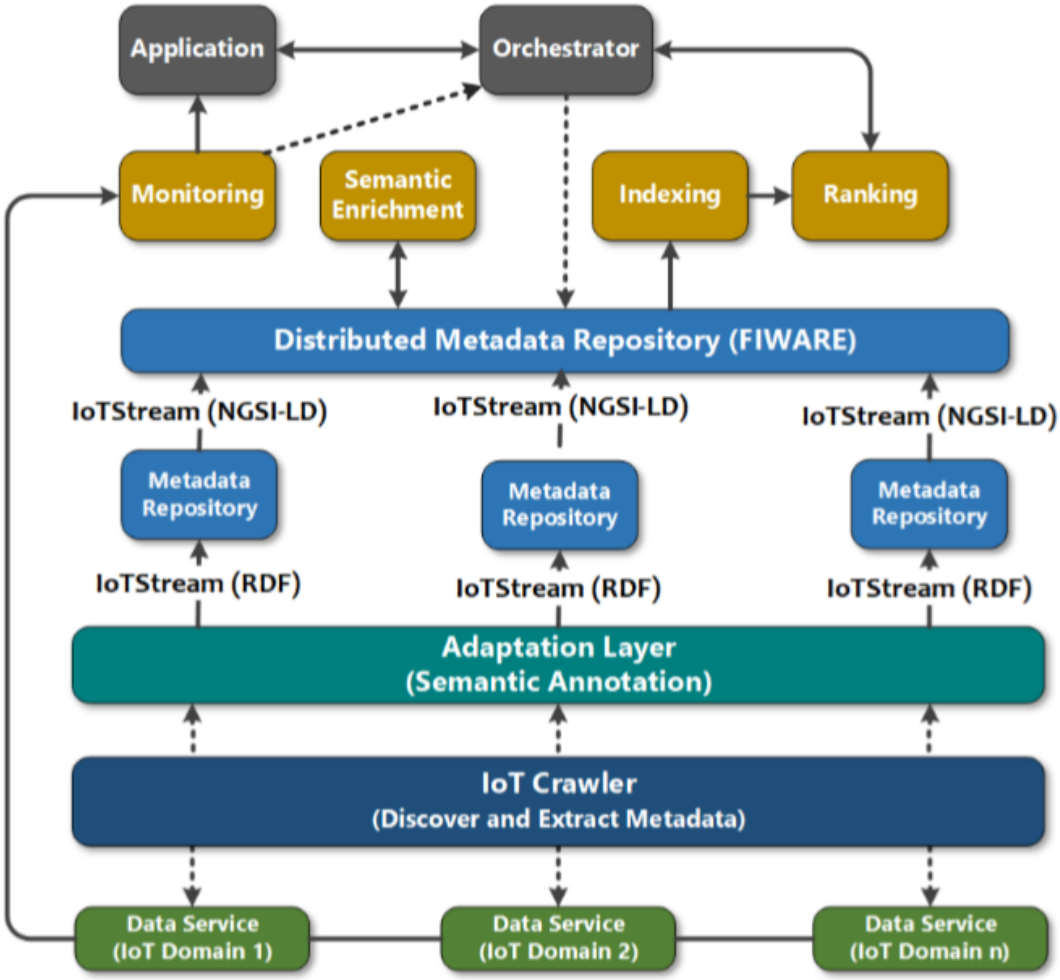


Figura 2.8 Arquitectura IoT Crawler con el uso de IoT-Stream. Tomado de [44].

2.6 Sensor

Los dispositivos LoRa de Semtech son una solución de largo alcance y bajo consumo de energía ampliamente adoptada para IoT. Brinda a las empresas de telecomunicaciones, los creadores de aplicaciones de IoT y los integradores de sistemas el conjunto de funciones necesarias para implementar redes de IoT interoperables y de bajo costo, puertas de enlace, sensores y Servicios de IoT en todo el mundo. Las redes de IoT basadas en la especificación LoRaWAN se han implementado en 100 países y Semtech es miembro fundador de LoRa Alliance, la alianza de IoT de más rápido crecimiento para aplicaciones de redes de área amplia de baja potencia.

Desde hace más de 100 años, en todo el mundo, CAHORS ayuda a transportar energía y facilita el acceso a la información. CAHORS ofrece soluciones y equipos globales adaptados a las especificidades de redes de distribución, fluidos y redes de comunicación de media y baja tensión. CAHORS implementa su considerable experiencia en todo el mundo para encontrar la solución adecuada y óptima para los clientes.

Los dispositivos LoRa de Semtech crean redes más inteligentes con detección precisa de fallas en la línea, mejoran el costo operativo y previenen fallas para una mayor eficiencia de la red [102].

"CAHORS identificó los dispositivos LoRa de Semtech y el protocolo LoRaWAN como la plataforma de Internet de las cosas (IoT) que ofrece las capacidades probadas para digitalizar la red de redes", dijo Christophe Aubigny, director de estrategia de producto de CAHORS. "Nuestras nuevas soluciones Sentinel monitorean continuamente la red de mediano voltaje y aprovecha las capacidades de bajo alcance y baja potencia de los dispositivos LoRa para transmitir datos de fallas en tiempo real. Los datos precisos y actualizados sobre la funcionalidad de la red permiten a los clientes detectar, localizar y abordar las fallas de la red de manera más eficiente y evitar costosas fallas del sistema ".

Instalados en postes de soporte de líneas eléctricas como se puede ver en la Figura 2.9, los productos Sentinel basados en LoRa de CAHORS monitorean la transferencia de campo de voltaje para localizar y predecir fallas en líneas eléctricas, incluidas fallas monofásicas conectadas a tierra y fallas continuas y fugitivas de múltiples fases. Los sensores identifican fallas en tiempo real y transmiten data streams de energía (dirección de fallas) a través de redes LoRaWAN a los administradores de la red de energía. Permite una respuesta rápida a las fallas en curso, aumenta la eficiencia general de la red y previene cortes de energía.

“Con la mayor digitalización de las redes inteligentes, las empresas eléctricas buscan aplicaciones más inteligentes para simplificar la integración y la implementación”, dijo Rémi Demerlé, director de marketing vertical para servicios públicos en el grupo de productos inalámbricos y de detección de Semtech. “LoRa de Semtech representa una tecnología líder para el mercado vertical de servicios públicos inteligentes, creando soluciones de medición que son flexibles, escalables, fáciles de usar e implementar. Los dispositivos LoRa aceleran la comercialización de soluciones de IoT que ofrecen a los clientes la capacidad de reducir los gastos operativos y crear nuevas eficiencias " [103].

Figura 2.9 Dispositivo Cahors instalado en un poste eléctrico.

2.7 Conclusiones del capítulo

El diseño del Sistema para la detección de fallas se sustenta de varios modelos. Un modelo apoyado en una ontología de dominio que facilita el razonamiento de los datos, un modelo de anotación que especifica el tipo de falla eléctrica, su localización geográfica y un algoritmo de agrupamiento que facilita el reconocimiento de las posibles fallas a través de un patrón de comportamiento numérico.

CAPÍTULO 3. PRUEBA CONCEPTUAL DEL DISEÑO DE UN SISTEMA PARA LA DETECCIÓN DE CORTOCIRCUITOS MEDIANTE TÉCNICAS DE DATA STREAM

Las aportaciones del presente trabajo radican en el empleo de técnicas avanzadas para viabilizar procedimientos comunes vinculados al sistema electro-energético. Para ello se diseñaron e implementaron utilizando diferentes herramientas de software, es una prueba conceptual que demuestra como el concepto que se maneja en el Capítulo 2 de la tesis se corresponde con una posible ejecución real.

En este Capítulo se presentan las pruebas y resultados de la propuesta. Su objetivo es mostrar mediante una prueba de concepto que el diseño del software funciona con las herramientas al uso para estos fines. Se muestra el proceso desde que se generan los datos hasta que se realiza una búsqueda y se mapean los resultados coherentemente.

3.1 Herramientas utilizadas

Como se mencionó anteriormente para alcanzar los objetivos es necesario el trabajo con varios softwares. Gracias a estos es que se puede crear una ontología, aplicarle un algoritmo de Machine Learning a los datos, almacenarlos en una base de datos y llevarlos a un triple store.

3.1.1 Creación de la ontología. Protégé

Protégé[11] es un editor de código abierto creado por la Universidad de Stanford, usado para construir ontologías y un marco general para representar el conocimiento. Está escrito en Java que es un lenguaje de programación orientado a objetos. Se usa para construir aplicaciones para la Web Semántica, para hacer una descripción semántica de la información.

Permite crear fácilmente clases y jerarquías, declarar propiedades para las clases, crear instancias e introducir valores, en un entorno de menús, botones y representaciones gráficas fáciles de usar. Las aplicaciones desarrolladas con Protégé se usan para la resolución de problemas y la toma de decisiones en un dominio especifico. El usuario puede realizar las siguientes operaciones: creación de una ontología, inserción de datos y optimización de datos de entrada.

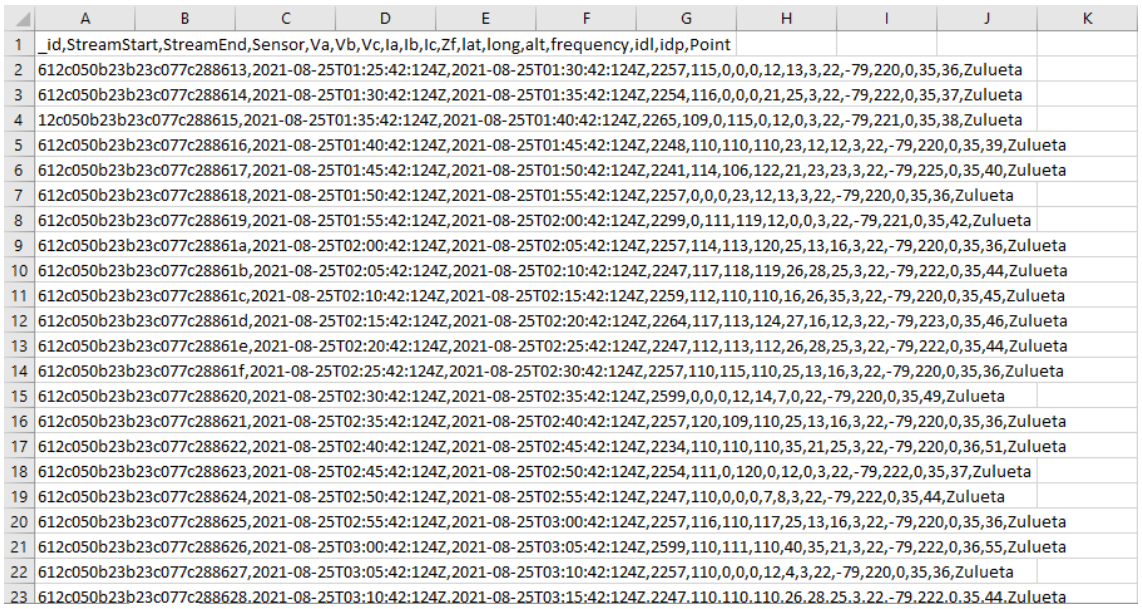
[11] https://protege.stanford.edu/

Esta poderosa herramienta en su versión 5.2 permitió hacer la ontología, crear sus clases, axiomas y todas las métricas que esta conlleva. Ahí se especificaron las reglas las cuales son esenciales a la hora del razonamiento incremental.

Protégé genera un archivo *.owl, que contiene la ontología que está lista para ser importada en otra herramienta compatible con datos expresados en grafos, en este caso Open Link Virtuoso.

3.1.2 Máquina generadora de números aleatorios

Para dar cumplimiento a los objetivos del presente trabajo, se necesitan datos, muchos datos. En Cuba no hay sensores que los generen, por lo que hay que simular las observaciones de los sensores a través de un código de programación. Crea números aleatorios en el rango que se necesitan y con las variaciones pertinentes para que se generen varios tipos de fallas. También en este código se especifican la frecuencia de obtención los valores, que en este caso es de cinco minutos. Así se simulan los datos provenientes de sensores, los cuales se guardan en un archivo *.csv y se muestran a continuación (Figura 3.1).



	A	B	C	D	E	F	G	H	I	J	K
1	_id,StreamStart,StreamEnd,Sensor,Va,Vb,Vc,Ia,Ib,Ic,Zf,lat,long,alt,frequency,idl,idp,Point										
2	612c050b23b23c077c288613,2021-08-25T01:25:42:124Z,2021-08-25T01:30:42:124Z,2257,115,0,0,0,12,13,3,22,-79,220,0,35,36,Zulueta										
3	612c050b23b23c077c288614,2021-08-25T01:30:42:124Z,2021-08-25T01:35:42:124Z,2254,116,0,0,0,21,25,3,22,-79,222,0,35,37,Zulueta										
4	12c050b23b23c077c288615,2021-08-25T01:35:42:124Z,2021-08-25T01:40:42:124Z,2265,109,0,115,0,12,0,3,22,-79,221,0,35,38,Zulueta										
5	612c050b23b23c077c288616,2021-08-25T01:40:42:124Z,2021-08-25T01:45:42:124Z,2248,110,110,110,23,12,12,3,22,-79,220,0,35,39,Zulueta										
6	612c050b23b23c077c288617,2021-08-25T01:45:42:124Z,2021-08-25T01:50:42:124Z,2241,114,106,122,21,23,23,3,22,-79,225,0,35,40,Zulueta										
7	612c050b23b23c077c288618,2021-08-25T01:50:42:124Z,2021-08-25T01:55:42:124Z,2257,0,0,0,23,12,13,3,22,-79,220,0,35,36,Zulueta										
8	612c050b23b23c077c288619,2021-08-25T01:55:42:124Z,2021-08-25T02:00:42:124Z,2299,0,111,119,12,0,0,3,22,-79,221,0,35,42,Zulueta										
9	612c050b23b23c077c28861a,2021-08-25T02:00:42:124Z,2021-08-25T02:05:42:124Z,2257,114,113,120,25,13,16,3,22,-79,220,0,35,36,Zulueta										
10	612c050b23b23c077c28861b,2021-08-25T02:05:42:124Z,2021-08-25T02:10:42:124Z,2247,117,118,119,26,28,25,3,22,-79,222,0,35,44,Zulueta										
11	612c050b23b23c077c28861c,2021-08-25T02:10:42:124Z,2021-08-25T02:15:42:124Z,2259,112,110,110,16,26,35,3,22,-79,220,0,35,45,Zulueta										
12	612c050b23b23c077c28861d,2021-08-25T02:15:42:124Z,2021-08-25T02:20:42:124Z,2264,117,113,124,27,16,12,3,22,-79,223,0,35,46,Zulueta										
13	612c050b23b23c077c28861e,2021-08-25T02:20:42:124Z,2021-08-25T02:25:42:124Z,2247,112,113,112,26,28,25,3,22,-79,222,0,35,44,Zulueta										
14	612c050b23b23c077c28861f,2021-08-25T02:25:42:124Z,2021-08-25T02:30:42:124Z,2257,110,115,110,25,13,16,3,22,-79,220,0,35,36,Zulueta										
15	612c050b23b23c077c288620,2021-08-25T02:30:42:124Z,2021-08-25T02:35:42:124Z,2599,0,0,0,12,14,7,0,22,-79,220,0,35,49,Zulueta										
16	612c050b23b23c077c288621,2021-08-25T02:35:42:124Z,2021-08-25T02:40:42:124Z,2257,120,109,110,25,13,16,3,22,-79,220,0,35,36,Zulueta										
17	612c050b23b23c077c288622,2021-08-25T02:40:42:124Z,2021-08-25T02:45:42:124Z,2234,110,110,110,35,21,25,3,22,-79,220,0,36,51,Zulueta										
18	612c050b23b23c077c288623,2021-08-25T02:45:42:124Z,2021-08-25T02:50:42:124Z,2254,111,0,120,0,12,0,3,22,-79,222,0,35,37,Zulueta										
19	612c050b23b23c077c288624,2021-08-25T02:50:42:124Z,2021-08-25T02:55:42:124Z,2247,110,0,0,0,7,8,3,22,-79,222,0,35,44,Zulueta										
20	612c050b23b23c077c288625,2021-08-25T02:55:42:124Z,2021-08-25T03:00:42:124Z,2257,116,110,117,25,13,16,3,22,-79,220,0,35,36,Zulueta										
21	612c050b23b23c077c288626,2021-08-25T03:00:42:124Z,2021-08-25T03:05:42:124Z,2599,110,111,110,40,35,21,3,22,-79,222,0,36,55,Zulueta										
22	612c050b23b23c077c288627,2021-08-25T03:05:42:124Z,2021-08-25T03:10:42:124Z,2257,110,0,0,0,12,4,3,22,-79,220,0,35,36,Zulueta										
23	612c050b23b23c077c288628,2021-08-25T03:10:42:124Z,2021-08-25T03:15:42:124Z,2247,110,110,110,26,28,25,3,22,-79,222,0,35,44,Zulueta										

Figura 3.1 Observaciones simuladas gracias a la máquina generadora de números aleatorios.

3.1.3 PyCharm

Python[12] es el lenguaje de programación más popular de los últimos años, se utiliza PyCharm[13], que es un IDE, es decir, no solo es un editor de código, sino que también tiene un depurador, un intérprete y otras herramientas que ayudarán a crear y exportar los programas que sean creados**.**

Los datos que en este caso preceden de una Máquina generadora de números homogéneos con tendencia a variaciones, simulando a los sensores, son procesados a través del algoritmo *K-Means.* Se utiliza un algoritmo de clasificación no supervisada ya que los datos no tienen etiquetas y se clasifican a partir de su estructura interna (propiedades, características).

K-Means es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en *k* grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Se suele usar la distancia cuadrática.

El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos, *k*, se establecen *k* centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.
2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.
3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

El algoritmo *K-Means* resuelve un problema de optimización, siendo la función a optimizar (minimizar) la suma de las distancias cuadráticas de cada objeto al centroide de su cluster.

---

[12] https://www.python.org/

[13] https://www.jetbrains.com/es-es/pycharm/

Los objetos se representan con vectores reales de d dimensiones $(\mathbf{x}1,\mathbf{x}2,\ldots,\mathbf{x}n)$ y el algoritmo *K-Means* construye k grupos donde se minimiza la suma de distancias de los objetos, dentro de cada grupo $\mathbf{S}=\{S1,S2,\ldots,Sk\}$, a su centroide. El problema se puede formular de la siguiente forma:

$$\underset{S}{min}\, \boldsymbol{E}(\boldsymbol{\mu_i}) = \underset{S}{min} \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \sum_{\boldsymbol{x_j} \in \mathbf{S_i}} \left\| \boldsymbol{x_j} - \boldsymbol{\mu_i} \right\|^2$$

donde $\mathbf{S}$ es el conjunto de datos cuyos elementos son los objetos $\mathbf{x}j$ representados por vectores, donde cada uno de sus elementos representa una característica o atributo. Se obtienen k grupos o clusters con su correspondiente centroide $\boldsymbol{\mu i}$.

En cada actualización de los centroides, desde el punto de vista matemático, se impone la condición necesaria de extremo a la función $E(\boldsymbol{\mu i})$ que, para la función cuadrática anterior es:

$$\frac{\partial E}{\partial \boldsymbol{\mu}_i} = 0 \implies \boldsymbol{\mu}^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}j \in S_i^{(t)}} \mathbf{x}j$$

y se toma el promedio de los elementos de cada grupo como nuevo centroide.

Las principales ventajas del método *K-Means* son que es un método sencillo y rápido. Pero es necesario decidir el valor de k y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.

Más adelante se describe paso a paso como el algoritmo transforma los datos en bruto y los clasifica. Este proceso es necesario a la hora de trabajar con Big Data. Cuando miles y miles de sensores ubicados en todo el país, envíen datos directamente a una base de datos, esta no va a aguantar la sobrecarga del procesamiento de los mismos sin antes haberlos clasificados. El método de agrupamiento KMeans tiene la capacidad de analizar bases de datos con más de 10 000 individuos.

En Python, existen herramientas que permiten a los desarrolladores generar mapas con una capa extra de representación y visualización. Uno de los recursos que permite esto es Folium[14], una librería que combina las capacidades de análisis de datos con Python y los mapas.

3.1.4 Base de datos NoSQL. Mongodb

Mongodb[15] es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++**,** que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.
MongoDB viene de serie con una consola desde la que se ejecutan los distintos comandos. Esta consola está construida sobre JavaScript, por lo que las consultas se realizan utilizando ese lenguaje. Además de las funciones de MongoDB, se utilizan muchas de las funciones propias de JavaScript. En la consola también se definen variables, funciones o utilizar bucles.
Si se quiere usar nuestro lenguaje de programación favorito, existen drivers para un gran número de ellos. Hay drivers oficiales para C#, Java, Node.js, PHP, Python, Ruby, C, C++, Perl o Scala.

Aunque se suele decir que las bases de datos NoSQL tienen un ámbito de aplicación reducido, MongoDB se puede utilizar en muchos de los proyectos que se desarrollan en la actualidad ya que tiene amplia referencia en los entornos de Big Data.

Cualquier aplicación que necesite almacenar datos semi estructurados puede usar MongoDB. Una de las ventajas más importantes de las bases de datos NoSQL es su escalabilidad horizontal e incluso tolerancia a fallos (dependiendo del tipo), por eso su interés para Big Data.
Tiene una enorme importancia ya que sirve además de almacenar datos como una fuente de datos si se quieren realizar búsquedas a datos históricos, o aplicar algún otro algoritmo de Machine Learning para predecir futuros eventos.
El archivo *.csv que se exporta después de aplicarle KMeans a los datos, es cargado en Mongodb.

[14] https://python-visualization.github.io/folium/

[15] https://www.mongodb.com/

3.1.5 Open Link Virtuoso

En esta prueba de concepto se utilizó Virtuoso Universal Server un híbrido del motor de base de datos. Un middleware que combina la funcionalidad de un sistema de administración de base de datos relacional tradicional (RDBMS), base de datos relacional de objetos (ORDBMS), base de datos virtual, RDF, XML, texto libre, servidor de aplicaciones web y servidor de archivos. En lugar de tener servidores dedicados para cada uno de los dominios de funcionalidad antes mencionados, Virtuoso es un "servidor universal"; habilita un único proceso de servidor multiproceso que implementa múltiples protocolos. La edición gratuita y de código abierto de Virtuoso Universal Server también se conoce como Open Link Virtuoso[16].

Virtuoso es un triple store el cual es adecuado para dar solución a nuestro trabajo. Los datos que se encuentran en MongoDB se cargan en Virtuoso. También la ontología diseñada en Protégé se importa en este software. A través de RDF Mapping Language (R2RML) se hacen coincidir los datos del csv provenientes de los sensores (ya aplicado KMeans y guardados en MongoDB), que son instancias, con sus respectivos campos en el grafo de la ontología. Esto permite que se haga un razonamiento y se puedan realizar búsquedas a los datos.

3.2 Algoritmo de clasificación no supervisada. K-Means

Los datos al ser generados se guardan en tablas, las cuales están almacenadas en archivos *.csv. Este se carga en PyCharm para analizar esos datos.

En el Anexo VI se encuentra el código de programación utilizado para aplicar el clustering el cual esta explicado a continuación:

- Las librerías de Python que se usaron para realizar este proceso de análisis, clasificación, agrupamiento y muestreo son numpy (Se utiliza para hacer cálculos científicos), pandas (Ayuda en el análisis de datos), matplotlib (Impresión de gráficos en alta calidad) y sklearn (Aquí se encuentra la función KMeans).

-Seguido se carga el archivo *.csv el cual tenga los datos a analizar. En este caso como es una prueba de concepto solo se utiliza una pequeña cantidad de datos. Al producirse un stream genera valores que

[16] https://virtuoso.openlinksw.com/

cambian como son los voltajes y las corrientes, pero también existen otras columnas en la tabla como son los id, los WindowsStart y WindowsEnd, la geolocalización, que son datos necesarios y muy importantes, pero en este momento solo se necesitan (para hacer el algoritmo) los valores de Va, Vb, Vc, Ia, Ib e Ic. Se creó una nueva variable la cual solo contenga estos valores y posteriormente se normalizaron los mismos.

-Con estos valores en la nueva variable ya normalizados se le aplica un método llamado "Codo de Jambú" el cual consiste en ir creando diferentes cantidades de clusters y calculando que tan similares son los individuos dentro de los mismos e ir plasmando esta información en una gráfica y así decidir cuantos clusters son necesarios para dar solución al problema. En la Figura 3.2 se presenta el resultado de aplicar este método a los datos seleccionados.

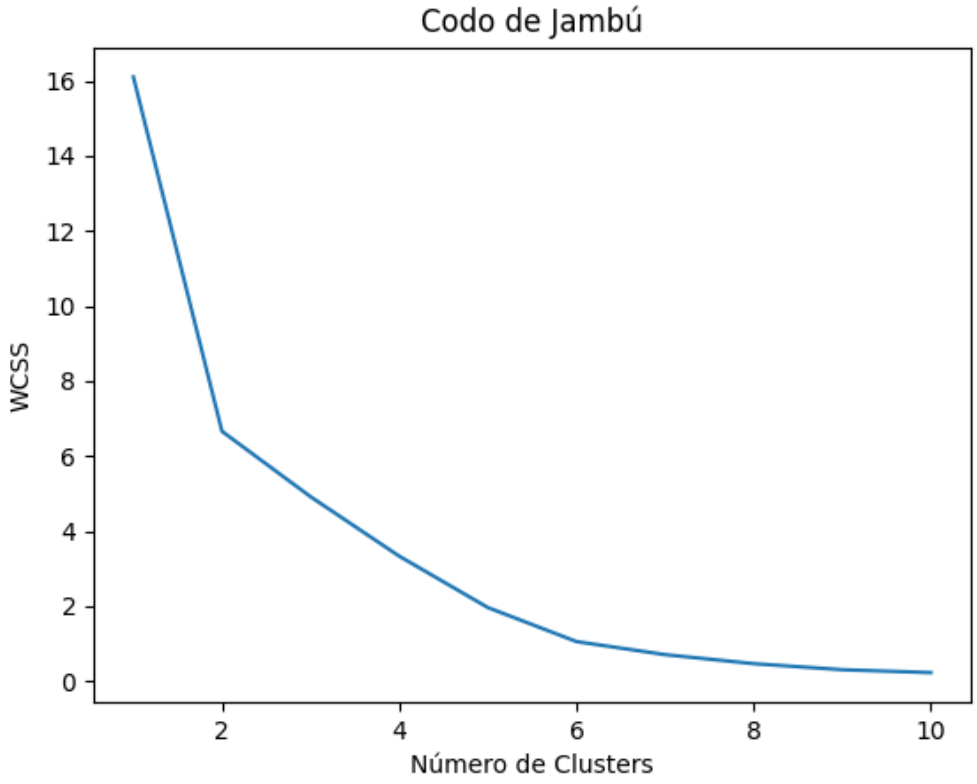


Figura 3.2 Método Codo de Jambú.

Mientras más similares sean los individuos, más distantes se encontrarán los clusters que se formen, que es lo que se busca en el clustering, la formación de grupos bien delimitados que contengan individuos cuya distancia entre ellos sea la menor posible. Para medir esta distancia se utiliza la suma

de los cuadrados dentro de cada grupo (WCSS). Como se muestra en la figura anterior el valor de WCSS va disminuyendo a medida que aumenta la cantidad de clusters, de manera que al llegar a los 10 clusters el valor de WCSS ya es muy pequeño en comparación a cuál se tenía con 1 clusters, lo cual es bueno, pero se estarían realizando demasiadas particiones, por lo que se busca un punto donde el valor de WCSS deje de disminuir de manera drástica. En este caso, ese punto es 2, por lo que este es la cantidad óptima de clusters a utilizar.

-Al conocer ya el número de clusters que se van a agrupar los datos, se aplica el método KMeans y se guardan en una nueva columna del csv, cada stream a que cluster pertenece. Seguido el análisis de componentes principales para dar una idea de cómo se formaron los clusters, se muestra en la Figura 3.3.

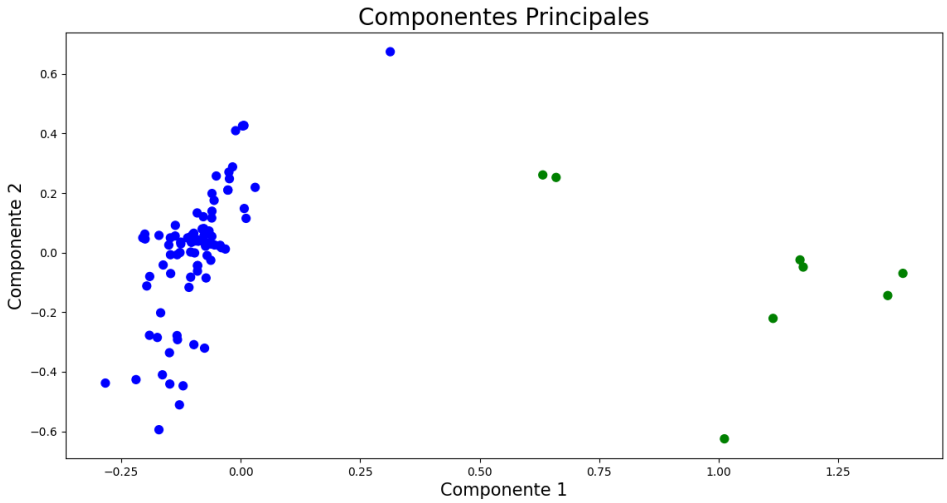


Figura 3.3 Datos ya agrupados.

Cada punto representa un streams, y los colores a cada clusters que pertenece. Los dos clusters están bien delimitados, ya que se están comparando valores de voltaje y corriente rondando los 110 volts y los 20 amperes respectivamente con respecto al otro cluster que son valores en 0. Como se puede apreciar, si se hicieron bien los cluster, no hay puntos azules y verdes mezclados, los azules representan los valores de la medición que se encuentran en los rangos de funcionamientos adecuados y los verdes los que poseen algún valor cero; ello indica que la probabilidad de ocurrencia de una falla en los clusters en verde es muy alta. Este algoritmo no es para decidir qué tipo de falla ocurrió ni para decir si

hay una falla, aunque si la detecta, es solo para agrupar los datos y después para ganar en precisión, el Razonamiento Incremental que utiliza la ontología y sus reglas deciden qué fue lo que ocurrió.

-Finalmente se guarda el archivo *.csv para ser cargados en la base de datos.

3.3 Base de datos y triple store

La base de datos NoSQL escogida es MongoDB ya que es muy fácil de usar y permite trabajar con Big Data. Todas las herramientas que son utilizadas a continuación, no guardan los datos históricos, por lo que si se quiere hacer un análisis o aplicar un algoritmo MongoDB salva los datos.

El archivo que contiene los datos se carga en el software Virtuoso (Figura 3.4), al igual que la ontología diseñada en Protégé (Figura 3.5). Es necesario relacionar estos datos con sus respectivos campos en el grafo importado. Para hacer posible esto es necesario utilizar R2RML Scripts. También hay que agregar manualmente los namespaces que fueron utilizados en la ontología y están mostrados en la Tabla 1.1 presente en el Capítulo 2.

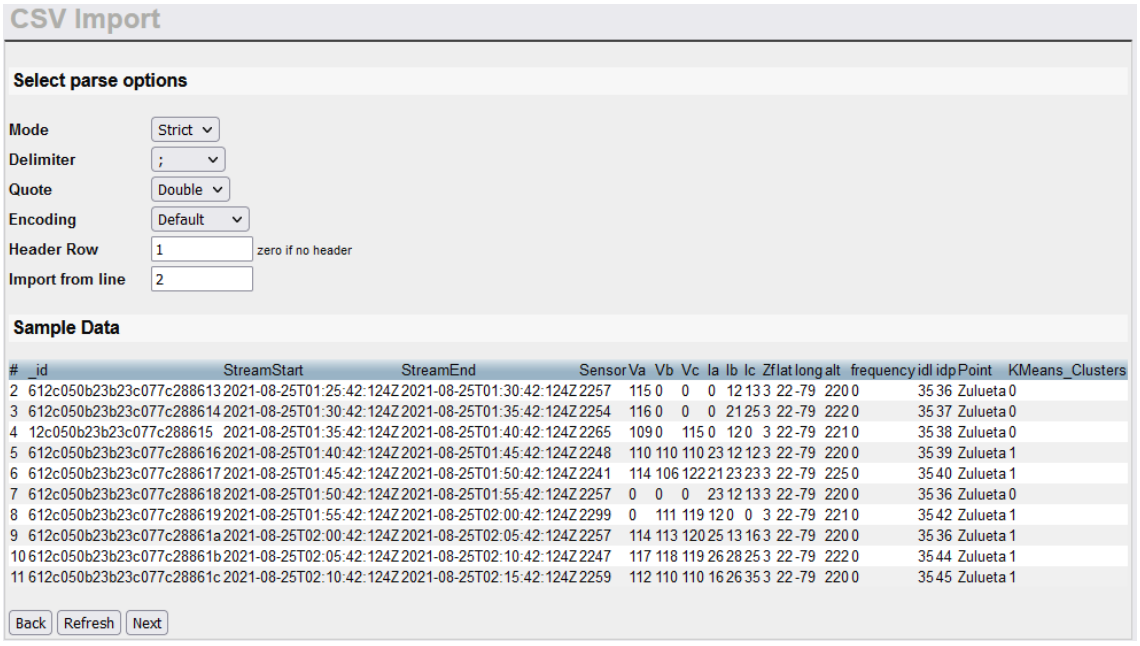


Figura 3.4 Cargando el CSV en Virtuoso.

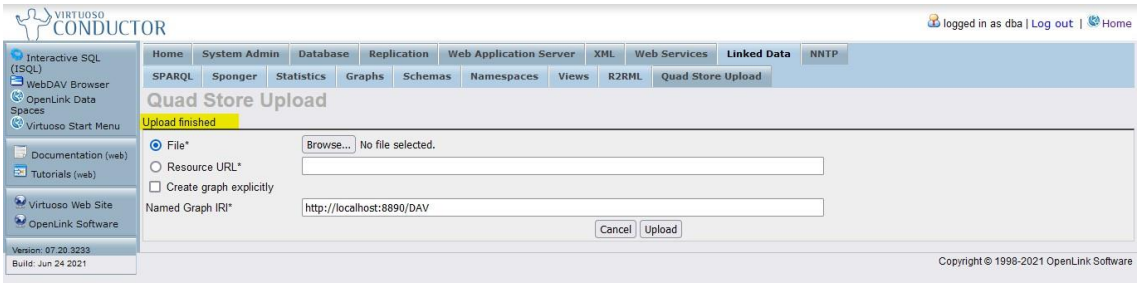


Figura 3.5 Importar el grafo en Virtuoso.

RDF Mapping Language (R2RML) es una recomendación del W3C que permite especificar reglas para transformar bases de datos relacionales a RDF. Estos datos en RDF se pueden materializar y almacenar en un sistema gestor de tripletas RDF (normalmente conocidos con el nombre triple store), en el cual se pueden evaluar consultas SPARQL. Sin embargo, hay casos en los cuales la materialización no es adecuada o posible, por ejemplo, cuando la base de datos se actualiza frecuentemente. En estos casos, lo mejor es considerar los datos en RDF como datos virtuales, de tal manera que las consultas SPARQL anteriormente mencionadas se traduzcan a consultas SQL que se pueden evaluar sobre los sistemas gestores de bases de datos relacionales (SGBD) originales.

El Conductor Virtuoso produce un archivo en el leguaje de mapeo R2RML[17] scripts. Estos scripts autogenerados crean una ontología genérica de las tablas relacionales SQL (Figura 3.6).

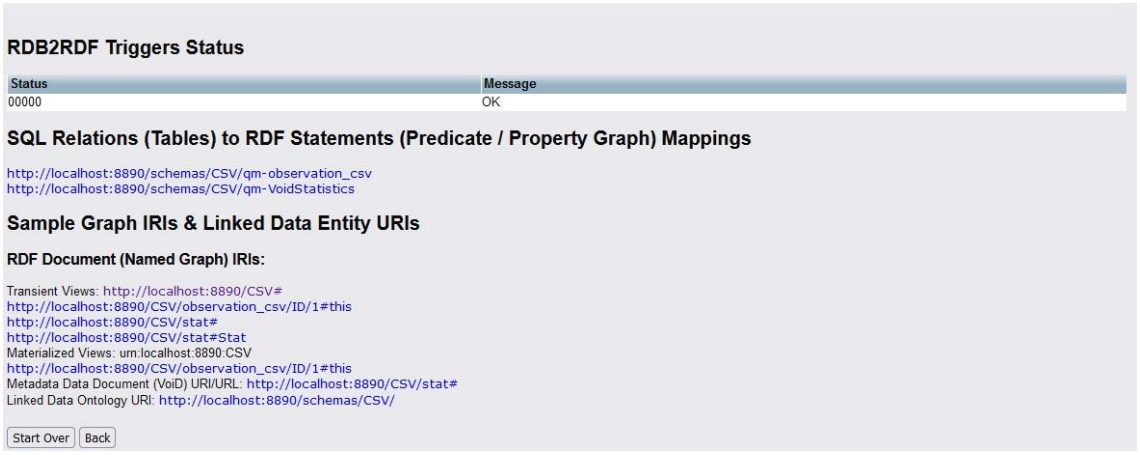


Figura 3.6 Resultado del mapeo de las tablas en RDF.

[17] https://www.w3.org/TR/r2rml/

Los usuarios de Virtuoso pueden personalizar estos scripts básicos para mapear tipos de columna generados automáticamente a propiedades de clase desde una ontología externa, o para mapear múltiples columnas de tabla a una sola relación RDF, entre otras cosas.

Se modifica el grafo R2RML generado por defecto (Anexo VII), con el cual se hace coincidir las columnas del csv donde están las instancias (datos) con su respectivo predicado dentro del grafo. En la Figura 3.7 se muestra una consulta en el software virtuoso donde se evidencia que el proceso anterior está correcto. Se puede percibir claramente que todos los valores de Ia, Ib e Ic precedentes del csv con los datos, se integraron correctamente al grafo como datos de Corriente, así mismo los valores de Va, Vb y Vc son Voltajes y Zf como Impedancia. Nótese que la consulta en SPARQL describe todas las observaciones que se están realizando sobre el csv.

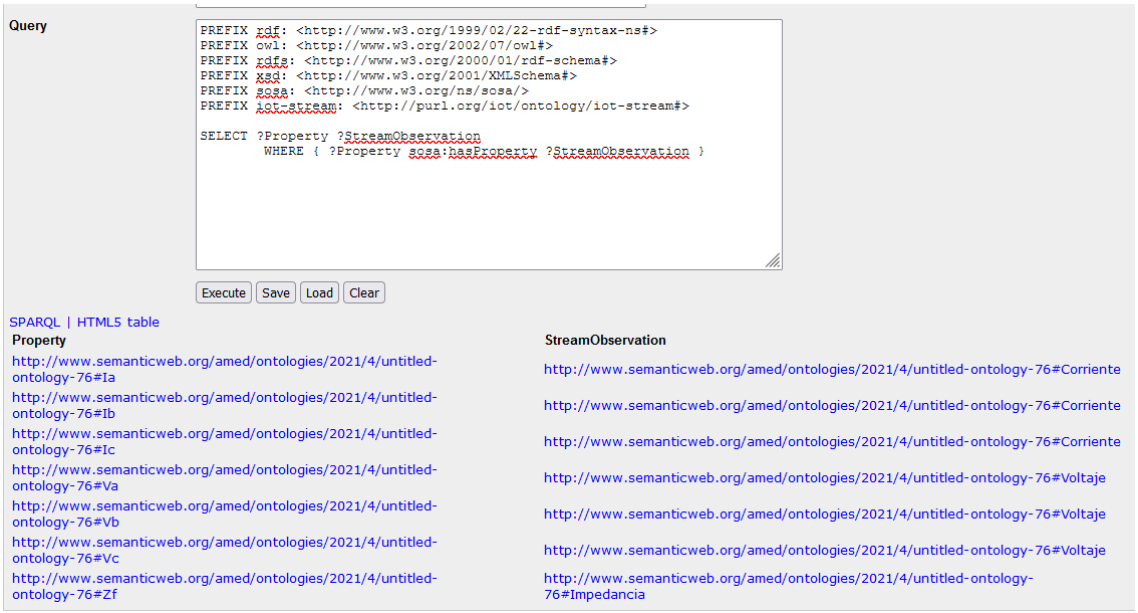


Figura 3.7 Consulta en SPARQL para demostrar la integración de los datos con la ontología en el triple store.

3.4 Consultas SPARQL

La herramienta Virtuoso tiene la capacidad de realizar consultas a los datos. Una vez que se ha realizado todo el proceso antes descrito, Virtuoso realiza un razonamiento, el cual esta descrito en la ontología a través de reglas que ya fueron declaradas en el capítulo de diseño. Estas reglas, a partir de condiciones escritas en el lenguaje SWRL, son capaces de analizar las condiciones que implican un

cortocircuito. El grafo capta las instancias que están en el csv mediante OBDA y anota los elemento asociados a la falla. En la Figura 3.8 se muestra un ejemplo de consulta la cual se le pide al sistema de análisis que devuelva todos los Cortocircuitos que están, indica que en Zulueta existe algún tipo de cortocircuito. En este caso se muestre el punto geográfico donde existe al menos un cortocircuito.

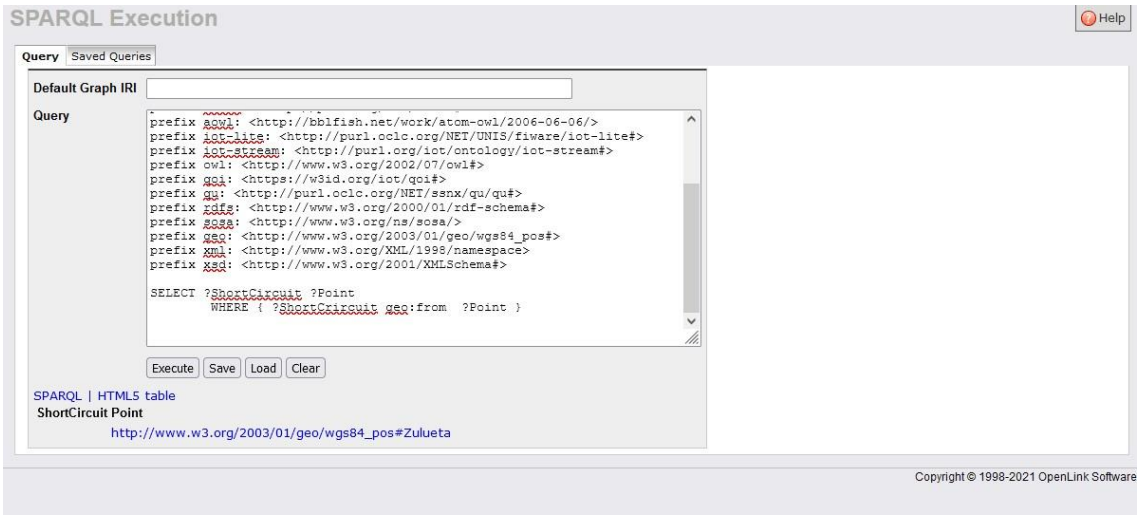


Figura 3.8 Ejemplo de consulta en SPARQL.

Con esto no basta, se necesita saber cuántos stream resultaron ser cortocircuitos y su ubicación geográfica. Para ello se realiza una nueva consulta modificando las clases que se quieren mostrar. En la Figura 3.9 se aprecia que como resultado de las inferencias existen solo tres streams que resultaron ser fallas.

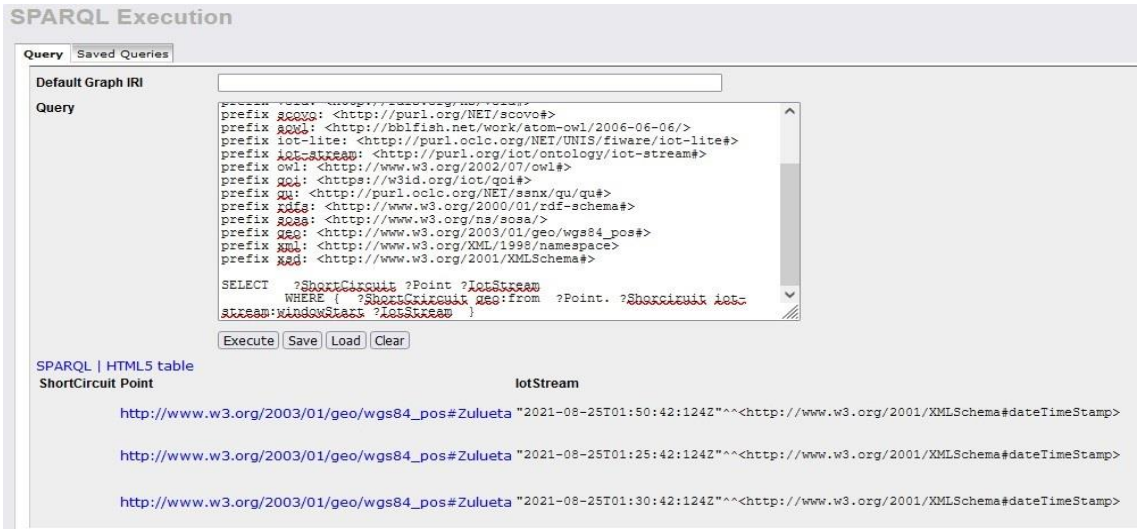


Figura 3.9 Consulta de las fallas y su ubicación geográfica.

Es importante conocer que sensor es el que está haciendo la lectura de los datos y sus propiedades. Para ello se realiza una nueva consulta (Figura 3.10) la cual tiene como objetivo que muestre la variable que está midiendo y el nombre del sensor que la mide.

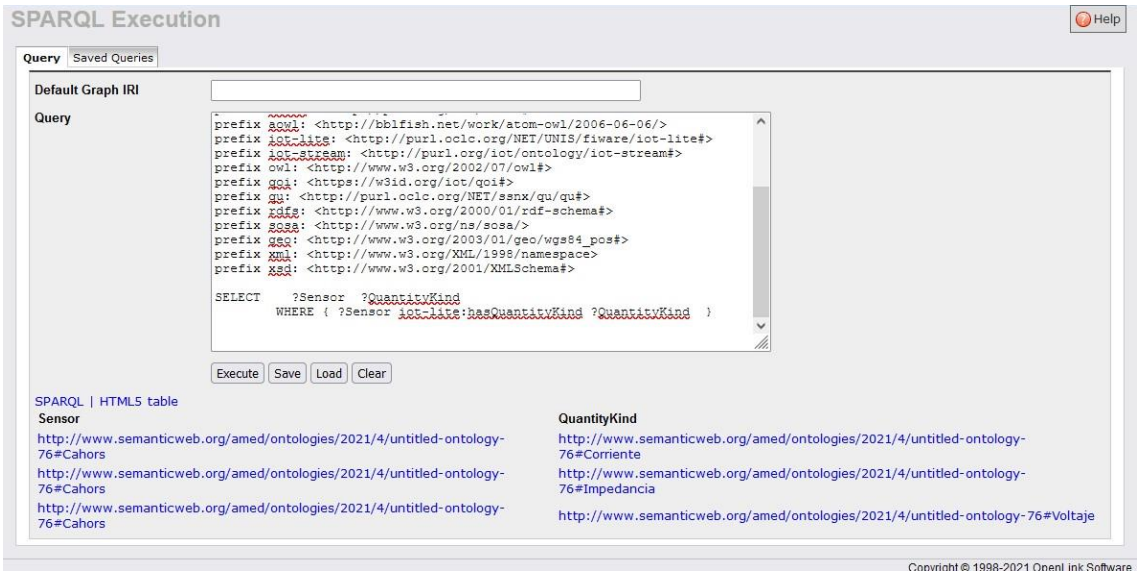


Figura 3.10 Consulta a las variables del sensor.

Si se quiere ver qué tipo de cortocircuito ocurrió se puede hacer una simple consulta, en la Figura 3.11, se pide que muestre los cortocircuitos que son bifásicos a tierra y su localización en el mapa.

Figura 3.11 Búsqueda de un tipo de cortocircuito en específico.

3.5 Representación geográfica. Folium

La representación de la consulta antes mostrada debe estar en un escenario amigable al usuario. Esto implica el desarrollo de un algoritmo final, el cual muestre estos datos en un mapa. Folium es una librería Python que permite crear mapas interactivos usando Leaflet.js[18]. Lo que hace, de forma elegante, es crear código java script que usa la maravillosa librería de mapas interactivos leaflet. En el Anexo IIX se encuentra el código de programación utilizado.

En cualquiera de las representaciones con Folium el primer paso es desarrollar el mapa sobre el que se va a mostrar el resto de la información**.** Para ello, hay que centrarlo en un punto con latitud y longitud. Folium proporciona la clase **folium.Map()** que toma el parámetro de ubicación en términos de latitud y longitud y genera un mapa a su alrededor.

Estos mapas son interactivos, puede acercar y alejar haciendo clic en los botones positivo y negativo en la esquina superior izquierda del mapa, también puede arrastrar el mapa y ver diferentes regiones.

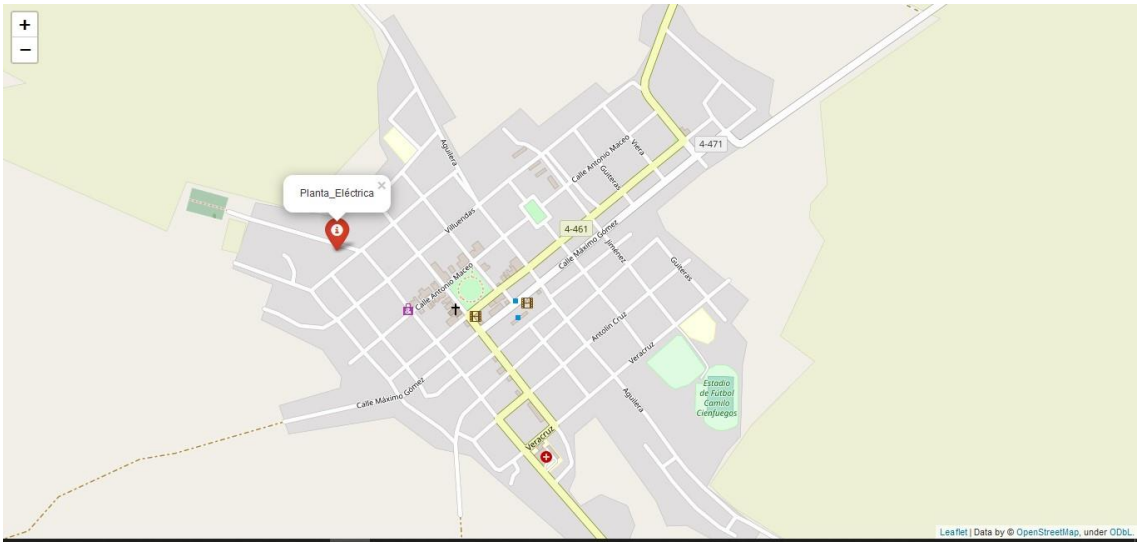


Figura 3.12 Mapa de Zulueta con el marcador de la Planta Eléctrica.

Los marcadores son los elementos que se utilizan para marcar una ubicación en un mapa. Por ejemplo, cuando utiliza Google Maps para la navegación, su ubicación se señala con un marcador y su destino con otro. Los marcadores se encuentran entre los elementos más importantes y útiles de un mapa.

[18] http://leafletjs.com/

Folium proporciona la clase **folium.Marker()** para trazar marcadores en un mapa. Simplemente se inserta las coordenadas de este marcador que en este caso es la Planta Eléctrica de Zulueta. Este es el punto de referencia (Figura 3.12) del poblado para los trabajadores de la Empresa Eléctrica.

Solo queda realizar un marcado en el mapa de todas los streams que resultaron ser cortocircuitos. Para ello se carga el archivo que contiene el resultado de la consulta en virtuoso. Aquí están los datos para darle al usuario ese entorno amigable tan necesario. En este caso solo ocurrieron tres cortocircuitos, algo que ocurre poco en un poblado tan pequeño, pero en ciudades más grandes es algo normal. Se crea un marcador por cortocircuito mostrando su posición en el mapa, el sensor que lo detectó, el tipo de cortocircuito, la altitud, la línea eléctrica que pertenece, el poste y el tiempo que se inició y terminó la observación (Figura 3.13).

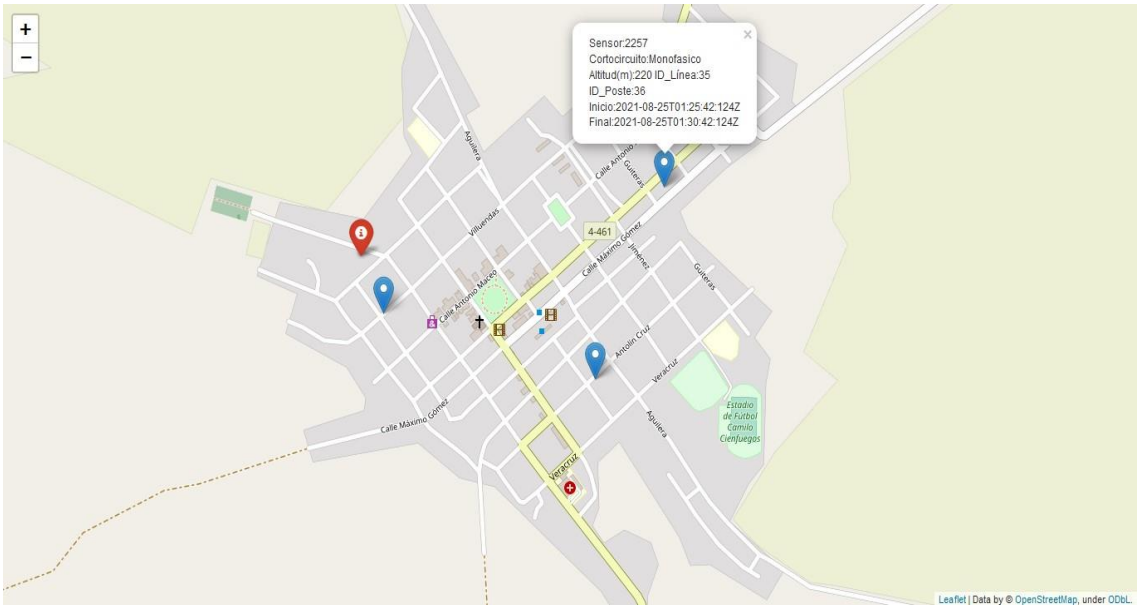


Figura 3.13 Resultado del mapeo de los cortocircuitos detectados.

Folium brinda plugins, los cuales tienen un gran valor según el uso que le sea dado. En este caso se utiliza **HeatMap()**, el cual es para representar los puntos geográficos en un mapa de calor. Así se puede identificar mejor las áreas abarcadas por la falla y si existen varios cortocircuitos cercanos facilita en su identificación (Figura 3.14). El código de programación utilizado esta en el Anexo IX.

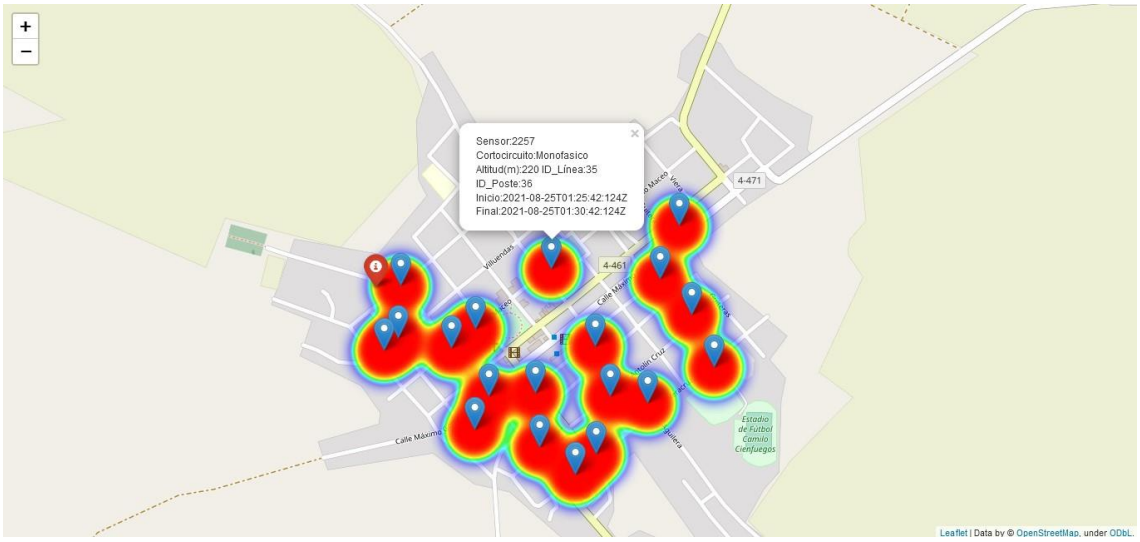


Figura 3.14 Mapa de calor.

En este caso al ser un pueblo relativamente pequeño ocurren pocas fallas comparadas con una gran ciudad. Desde el momento en que se genere un mapa de calor sobre una de estas inmensas ciudades se puede detectar que al ocurrir un cortocircuito estos derivan otros a su alrededor, y a través del mapa de calor se identifica de donde proviene la falla principal. Por esta razón se ha simulado la ocurrencia de varios cortocircuitos en Zulueta. Estos no son los cortocircuitos inferidos anteriormente por la ontología, ya que el número de muestras real es muy pequeño para visualizar la aplicación que tiene realizar un mapa de calor.

Otro plugins que brinda Folium es el de mapeo de densidad **MarkerCluster ()**, este literalmente lo que hace es unir los marcadores (cortocircuitos) cercanos en un solo marcador, con el objetivo de que si ocurren varias fallas cercanas no se llene en mapa de marcadores. Estos nuevos marcadores tienen la propiedad de informar el número de cortocircuitos que ellos están señalando y si se seleccionan muestran sobre el mapa el área donde se encuentran los cortocircuitos que le pertenecen. Para la correcta visualización de los marcadores se han simulado, al igual que para el mapa de calor, varios cortocircuitos (Figura 3.15). En el Anexo X se encuentra el código utilizado.

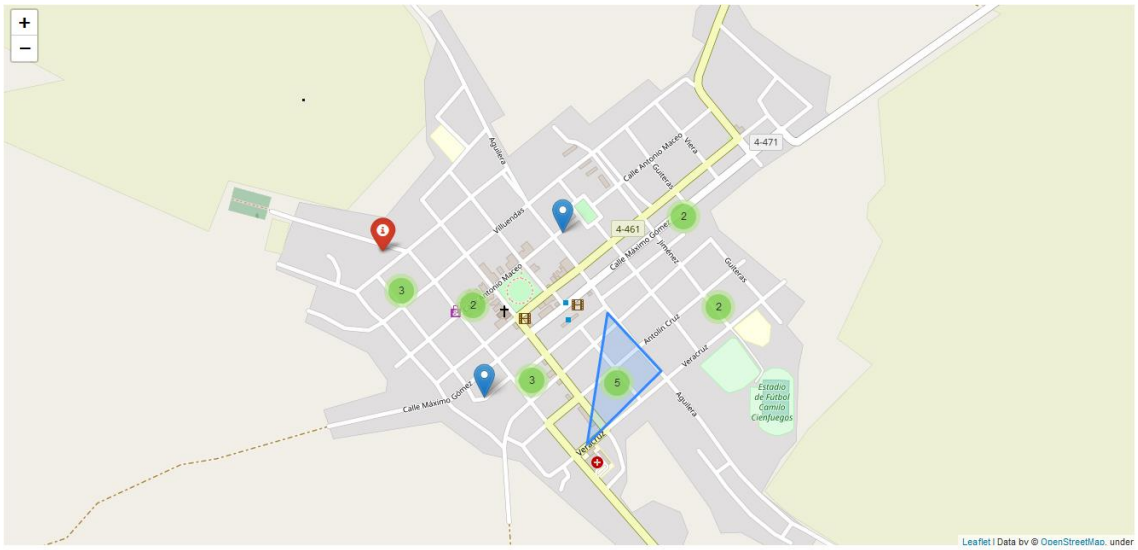


Figura 3.15 Varios marcadores.

3.6 Caso de uso

Un enfoque ilustrativo para demostrar cómo la ontología de IoT-Stream y las arquitecturas de sistemas poseen aplicabilidad en diversas esferas, es emplearlo en diferentes casos de uso. Aquí, las ciudades inteligentes y los dominios de vida inteligente se toman como ejemplo.

- Sistemas eléctricos en ciudades inteligentes

Para permitir que una ciudad se vuelva inteligente e impulse su economía digital a través de la investigación y la innovación industrial, la provisión de datos abiertos sobre diferentes aspectos de estas ciudades en crecimiento que tienen un impacto en el ciudadano es vital para lograr un desarrollo sostenible, un medio ambiente sano, seguro y próspero. Las ciudades de todo el mundo adoptan cada vez más esta visión. Algunos proporcionan datos históricos que son útiles para descubrir tendencias y predicciones futuras, y otros van un paso más allá para proporcionar datos sensoriales en tiempo real. Los datos se capturan en vivo y se publican en páginas web de dominio público donde cualquier usuario puede interactuar con ella. Los stream son tomados de los sensores que miden las variables que caracterizan los Sistemas Eléctricos. Para cada transformador, el sistema registra las variables de voltaje, corriente e impedancia en cada línea, por lo que si ocurre algún suceso inesperado (cortocircuito) se localiza casi automáticamente el lugar de la falla. También se proporcionan metadatos relacionados con la identificación, calibración, ubicación y lugar del sensor. Aquí, se puede

adoptar IoT-Stream para anotar este flujo de datos como datos vinculados, que luego se pueden usar o comparar con otros data stream relacionados con los Sistemas Eléctricos en otras provincias u otros postes. Además de anotar la instancia de la secuencia, sus observaciones, también puede anotar el proceso de análisis involucrado en el preprocesamiento de la secuencia de datos. Las ontologías vinculadas también se pueden utilizar para capturar información de ubicación importante para el análisis geoespacial utilizando *geo:Point*.

El proceso de análisis se puede realizar de la siguiente manera (Figura 3.16). En cada conjunto de datos, la frecuencia de los datos generados por los sensores es de cinco minutos y se establece un objetivo para representar patrones para cada hora. Con respecto a la representación del patrón horario, el tamaño del paso en este caso sería 12. Para el análisis, los datos se dividen en ventanas de 12 puntos de datos y se aplica K-Means en cada ventana. El resultado son patrones de datos por hora. Después aplicar los algoritmos de K-Means, el Razonamiento Incremental (RI) se usa para aplicar la agrupación en los patrones en tres grupos diferentes. Al observar los centros del cluster, se le da una etiqueta a cada grupo.

Las flechas azules indican el flujo de datos del stream data, mientras que las líneas grises ilustran el flujo eventual de los streams. En Analytics Services, "M" y "P" corresponden a los métodos y parámetros aplicados en los StreamObservations entrantes. Aquí, siete IotStream Producer publican StreamObservations al broker, y dos servicios de análisis están suscritos a notificaciones de StreamObservations por parte del broker. En el caso del Servicio de Análisis 1, el StreamObservations se recibe dentro de una hora y se procesan los datos usando K-Means, y la salida es anotada con una instancia de IotStream la cual fue *derivedFrom* y publicada en el broker. A continuación, se notifica al Servicio de Análisis 2 y se aplica RI en las observaciones de streams analizadas, y luego se agrupan en los grupos predefinidos. La salida de los clústeres se envía a un Generador de Eventos, que genera un Evento anotado con una etiqueta según el tipo de cortocircuito y la instancia de IotStream desde la que se detectó. Esta salida luego se publica en el broker para cualquier consumidor.

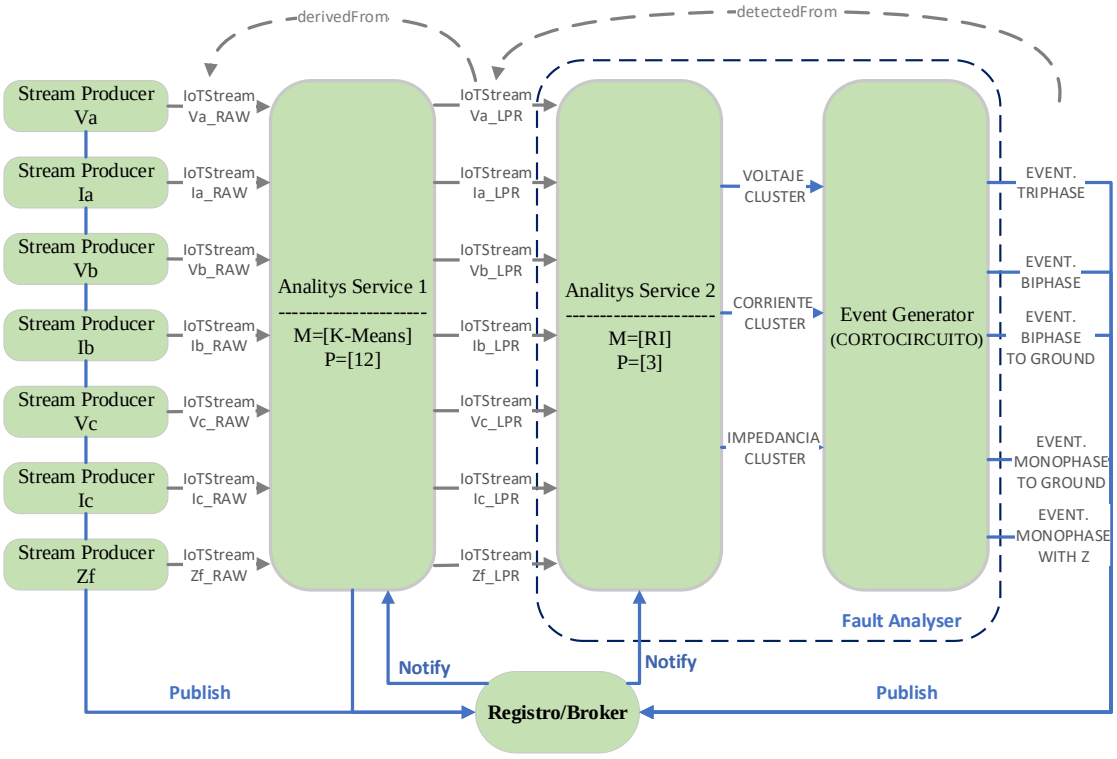


Figura 3.16 Proceso de anotación para la detección de cortocircuitos.

La puerta de enlace o Gateway sirve para comunicar los sensores con el router. A partir de ahí los stream pasan a la nube, que es la encargada de dirigir los datos a su destino. Primero los stream se almacenan en el triple store y semantic store, ya ahí entra en escena la ontología y todos los métodos de Big Data y Machine Learning, los cuales les dan atributos a esos datos aprobando su correcta utilización en tiempo real, como también su almacenaje para datos históricos. El sistema adopta un intermediario de datos desde FIWARE, que emplea la interfaz programada de aplicación NGSI-LD. Esto permite que a través de los protocolos de comunicación (TCP/IP y MQTT) y tecnología de Web Semántica se pueda intercambiar información en tiempo real desde un dispositivo conectado a internet como puede ser un teléfono móvil o una computadora, se muestra en la Figura 3.17. Los datos están publicados en la página web, la cual permite la interacción de cualquier usuario con ella, como, por ejemplo, ocurrió un apagón de electricidad en la zona Sur de Santa Clara, cualquier persona con conexión a internet puede entrar a esta página web y sabrá en tiempo real la magnitud, el tipo y la localización de la falla. Así mismo las empresas tienen acceso a estos servicios.

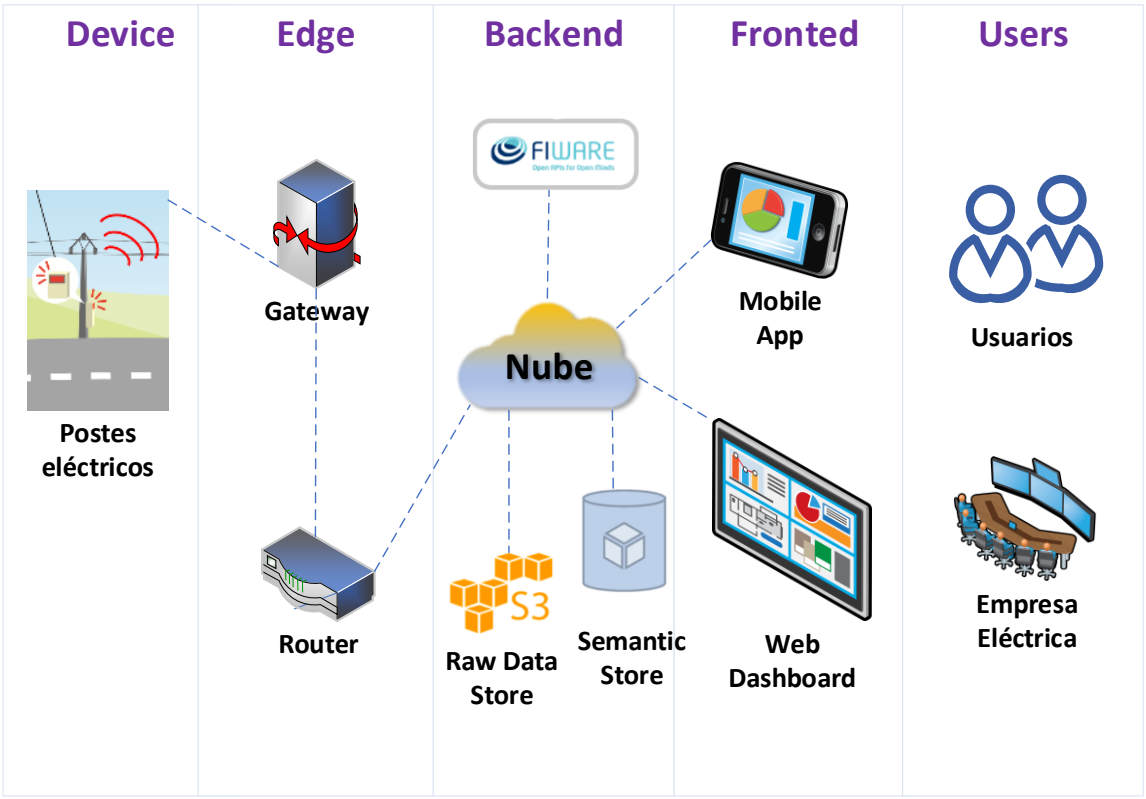


Figura 3.17 Captura de los procesos y rutas data stream de IoT para la detección de fallas.

3.7 Requisitos no funcionales

RnF_1. **Tipo de Aplicación:** El sistema debe estar concebido como una Aplicación Web que puede ser accedida por los clientes mediante una dirección URL.

RnF_2. **Finalidad:** El objetivo fundamental del sistema es poder informar a los usuarios de los despachos de carga y/o la Unión Eléctrica sobre las fallas mediante una interfaz en Línea. Para ello se requiere gestionar toda la información de las líneas según normas internacionales definidas por IEEE y otras definidas por el usuario. Además, se requiere controlar la adquisición de datos.

RnF_3. **Ambiente:** Para desplegar la aplicación se requiere de un servidor conectado a varias PCs (al menos 1 para que los usuarios consulten los stream en línea y otra para cada trabajador del despacho de cargo que los requiera según sus funciones).

RnF_4. **Utilizar un patrón de navegación:** Debe permitir el fácil acceso a las funcionalidades más utilizadas.

Confiabilidad

RnF_5. **Bloqueo de sesión automático:** Al transcurrir 10 minutos de inactividad en el sistema la sesión del usuario se bloquea automáticamente y solo puede ser reanudada si el usuario vuelve a autenticarse.

RnF_6. **Excepciones del sistema:** Ante un error imprevisto en el sistema lanza un mensaje de error con la causa que produjo dicho error.

Eficiencia

RnF_7. **Tiempo de respuesta próximo a los 5 segundos:** Ante una petición de búsqueda en sistema de mapas por parte del usuario el tiempo de respuesta no debe exceder los 5 segundos.

RnF_8. **Acceso de trabajadores:** El sistema debe soportar el acceso concurrente del 100% de los trabajadores de las entidades en un día de trabajo normal.

Soporte

RnF_9. **Se documentará el sistema:** Se documentará el sistema con un manual de usuario con el objetivo de explicar su uso y estará disponible como parte del sistema.

Restricciones de diseño

RnF_10. **Guardar en formato RDF:** El sistema debe estar diseñado para guardar los datos de las descripciones bibliográficas en formato RDF.

RnF_11. **Lenguaje Python:** Se utilizará para la construcción del sistema el lenguaje de programación Python y herramientas que se distribuyan bajo licencias libres.

RnF_12. **Mongo DB:** Se utilizará para la persistencia de los datos RDF.

RnF_13. **Desarrollo de nuevos módulos:** Los componentes del sistema deben desarrollarse siguiendo el principio de alta cohesión y bajo acoplamiento.

RnF_14. **Plataformas Windows y Gnu/Linux:** El sistema podrá ser accedido a través de un navegador web desde los sistemas operativos Windows y Gnu/Linux. Requisitos para la documentación de usuarios en línea y ayuda del sistema

RnF_15. **Manual de usuario:** El sistema incluirá un manual de usuario para aclarar dudas en cuanto al funcionamiento del mismo.

Componentes adquiridos

RnF_16. **Componentes open-source:** Todos los componentes del sistema deben ser open- source.

Interfaz Interfaces de usuario

RnF_17. **Navegadores:** Para acceder al Sistema debe usarse una versión del navegador Mozilla/Firefox igual o superior a la 1.0 o Internet Explorer igual o superior a la 5.0. No se garantiza la correcta visualización en otros navegadores.

RnF_18. **Acceso al ABCD:** Para acceder al Sistema el navegador debe tener habilitado el soporte para Cookies y Java Script.

Visualización:

RnF_19. **Interfaces Hardware:** La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024x768.

RnF_20. **Velocidad de transferencia servidor de Base de Conocimiento:** El servidor de ficheros, así como el servidor de aplicaciones, deberán contar como mínimo con: - RAM: 1GB. - Procesador: P IV 1.6 GHz. - HDD: 1 GB. - NIC: 1x GB.

RnF_21. **Características de las PC cliente:** Las condiciones mínimas de las PC clientes se mencionan a continuación: - Intel PIII 700 MHz. - RAM: 256 MB.

3.7 Análisis económico

Para evaluar el costo total del Sistema, primero hay que considerar que todas las herramientas informáticas utilizadas son gratis. Solo deja el gasto de los sensores y del servidor, el cual tendrá las bases de datos y el triple store.

En este caso se utiliza el sensor Cahors. El vendedor proporciona un sistema el cual tiene varios sensores que miden todas las variables que se necesitan y además temperatura, humedad, entre otras. El precio de cada unidad es de aproximadamente 183 euros.

3.8 Conclusiones del capítulo

La prueba conceptual desarrollada en esta investigación describe a cabalidad el Desarrollo de un Sistema de IoT no solo para la detección de fallas, sino que su aplicación puede implementarse en cualquier dominio de conocimientos que utilice sensores.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Los referentes teóricos de la detección de cortocircuitos en los Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica son varios. Las tecnologías de IoT y de la Web Semántica han evolucionado creando modelos de datos y sistemas que sustentan el desarrollo de aplicaciones de altísimo nivel en este terreno de la informática y las telecomunicaciones. Las aplicaciones fundamentales en este campo del conocimiento son los procesos e OBDA, las ontologías de dominio, las aplicaciones para datos enlazados y las técnicas para stream data que combinan técnicas de Inteligencia Artificial y Big Data
- El diseño del sistema para la detección de cortocircuitos en Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica posee todos los elementos utilizados en las tecnologías de IoT para sistemas de Web Semántica, pues combina modelos de agrupamiento y razonamiento incremental para facilitar la toma de decisiones.
- La prueba de conceptos la detección de cortocircuitos en Sistemas Eléctricos de Potencia en Cuba usando técnicas de Data Streams y Web Semántica demuestra la factibilidad del diseño al quedar confirmada el razonamiento y el agrupamiento de los cortocircuitos en determinada zona mediante diversas técnicas de visualización.

Recomendaciones

Toda vez que se reconocen las dificultades que acarrea el desarrollo de un sistema de este tipo en Cuba exponemos las siguientes recomendaciones.

- Notificar a la UNE la existencia de estas herramientas para el análisis de sus posibles implementaciones en los Sistemas Eléctricos de Potencia.
- Continuar desarrollando este sistema modelando otras fallas a fin de dar completitud a la base de análisis del Sistema.

REFERENCIAS BIBLIOGRÁFICAS

[1] S. Saha, M. Aldeen, C. P. J. I. J. o. E. P. Tan, and E. Systems, "Fault detection in transmission networks of power systems," vol. 33, no. 4, pp. 887-900, 2011.

[2] R. E. Brown, *Electric power distribution reliability*. CRC press, 2017.

[3] D. Hernández Morales, "Efecto de los cortocircuitos sobre el comportamiento estable de la generación sincrónica de redes débiles," Universidad Central" Marta Abreu" de Las Villas, Facultad de Ingeniería …, 2018.

[4] R. d. J. Aguas Ramos and A. N. Buelvas Berrocal, "Metodología para el Cálculo de las Corrientes de Cortocircuito," pp. 12-18, 2011.

[5] K. X. Cruz Rodríguez and C. M. Morales Cisneros, "Diseño y construcción de módulo didáctico de protecciones para sistemas eléctricos de potencia (SEP) aplicado a sistemas de transmisión," pp. 32-36, 2016.

[6] M. Macri, J. Suarez, and C. J. I. T. Dimenna, "Dinamica de un Transformador de Potencia Ante Cortocircuitos Externos," vol. 12, no. 2, pp. 161-168, 2001.

[7] M. Pignati, L. Zanni, P. Romano, and M. Paolone, "Method and system for fault detection and faulted line identification in power systems using synchrophasors-based real-time state estimation," ed: Google Patents, 2019.

[8] H. H. Alhelou, "Fault detection and isolation in power systems using unknown input observer," in *Advanced condition monitoring and fault diagnosis of electric machines*: IGI global, 2019, pp. 38-58.

[9] L. Zhao, P. Zhang, J. J. A. o. E. Huang, and E. Engineering, "Utilizing analytical hierarchy process and smart techniques in fault detection in electrical power networks," vol. 2, no. 5, pp. 19-25, 2019.

[10] M. Dashtdar, R. Dashti, and H. R. Shaker, "Distribution network fault section identification and fault location using artificial neural network," in *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, 2018, pp. 273-278: IEEE.

[11] Y. Q. Chen, O. Fink, and G. J. I. T. o. I. E. Sansavini, "Combined fault location and classification for power transmission lines fault diagnosis with integrated feature extraction," vol. 65, no. 1, pp. 561-569, 2017.

[12] M. Žarković and Z. J. E. P. S. R. Stojković, "Analysis of artificial intelligence expert systems for power transformer condition monitoring and diagnostics," vol. 149, pp. 125-136, 2017.

[13] Y. A. Pabón Guerrero and L. J. Rojas Bolaños, "MODELO ONTOLÓGICO PARA MANEJO DE PERFILES DE USUARIO EN LA IoT," p.7, 2017.

[14] L. García and L. J. I. d. S. Carlos, "Estudio del impacto técnico y económico de la transición de internet al internet de las cosas (IoT) para el caso Colombiano," p. 16, 2015.

[15] R. K. J. G. I. Sethi, San Jose, "The role of telecommunications in smart Cities," p. 2, 2016.

[16] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridgingwireless sensor networks into internet of things," in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2010, pp. 347-352: Ieee.

[17] L.-H. Yen and W.-T. J. C. C. Tsai, "The room shortage problem of tree-based ZigBee/IEEE 802.15. 4 wireless networks," vol. 33, no. 4, pp. 454-462, 2010.

[18] F. Wu, T. Wu, and M. R. Yuce, "Design and Implementation of a Wearable Sensor Network System for IoT-Connected Safety and Health Applications," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 87-90: IEEE.

[19] D. G. Costa, C. Duran-Faundez, D. C. Andrade, J. B. Rocha-Junior, and J. P. J. S. Just Peixoto, "Twittersensing: An event-based approach for wireless sensor networks optimization exploiting social media in smart city applications," vol. 18, no. 4, p. 1080, 2018.

[20] T. Muhammed, R. Mehmood, and A. Albeshri, "Enabling reliable and resilient IoT based smart city applications," in *International Conference on Smart Cities, Infrastructure, Technologies and Applications*, 2017, pp. 169-184: Springer.

[21] A. Cama-Pinto, E. De la Hoz, and D. Cama-Pinto, "Las redes de sensores inalámbricos y el internet de las cosas," pp. 168-170, 2012.

[22] S. Suakanto, S. H. Supangkat, and R. Saragih, "Smart city dashboard for integrating various data of sensor networks," in *International Conference on ICT for Smart Society*, 2013, pp. 1-5: IEEE.

[23] K. Guo, Y. Lu, H. Gao, and R. J. S. Cao, "Artificial intelligence-based semantic internet of things in a user-centric smart city," vol. 18, no. 5, p. 1341, 2018.

[24] T. Berners-Lee, J. Hendler, and O. J. S. a. Lassila, "The semantic web," vol. 284, no. 5, pp. 34-43, 2001.

[25] D. M. B. Torres, A. C. Romero, and J. S. G. J. P. Sanabria, "Web semántica, más de una década de su aparición," vol. 8, no. 1, pp. 61-69, 2017.

[26] O. B. Sezer, E. Dogdu, M. Ozbayoglu, and A. Onal, "An extended iot framework with semantics, big data, and analytics," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 1849-1856: IEEE.

[27] A. Bernstein, J. Hendler, and N. J. C. o. t. A. Noy, "A new look at the semantic web," vol. 59, no. 9, pp. 35-37, 2016.

[28] M. R. León, D. D. R. Ochoa, and A. J. B. J. T. C. Barajas, "Tendencias en computación: Web Semántica y Computación Cognitiva," vol. 12, no. 1, pp. 19-26, 2018.

[29] L. Morán, "Web semántica y educación en el nivel superior: una experiencia sobre su uso para la enseñanza y el aprendizaje," pp. 37-53, 2017.

[30] C. M. Keet, *An introduction to ontology engineering*. Maria Keet, 2018, pp. 2-7.

[31] A. C. García and J. F. M. Ortega, "Modelo de información común de un middleware semántico para Internet de la Energía," pp. 33-34, 2016.

[32] F. Manola, E. Miller, and B. J. W. C. r. McBride, "RDF primer," vol. 10, no. 1-107, p. 6, 2004.

[33] R. Angles and R. J. I. L. A. T. García, "Transforming RDF Data into Property Graphs," vol. 18, no. 01, pp. 130-137, 2020.

[34] M. B. M. Arciniega and V. S. J. O. Faggioni, "Modelo ontológico para la representación de datos académicos y su publicación con tecnología semántica," vol. 32, no. 10, pp. 267-282, 2016.

[35] M. Ramírez-Mora, "Desarrollo de un modelo de usuario basado en ontologías," p. 37, 2016.

[36] W. W. W. Consortium, "OWL 2 web ontology language document overview," 2012.

[37] M. A. Paredes Valverde, "Interfaces del lenguaje natural para la consulta y recuperación de información de bases de conocimiento basadas en ontologías," Facultad de Informática, UNIVERSIDAD DE MURCIA, 2017.

[38] S. Zhong, Z. Fang, M. Zhu, and Q. Huang, "A geo-ontology-based approach to decision-making in emergency management of meteorological disasters," 2017.

[39] D. Socías Segura, "Ontología Turística Geográfica," Enginyeria Técnica en Informàtica de Sistemes, s.f.

[40] T. Tambassi, "From a geographical perspective: spatial turn, taxonomies and geo-ontologies," in *The Philosophy of Geo-Ontologies*: Springer, 2018, pp. 27-36.

[41] G. C. Federico Fonseca "Geo-Ontologies," vol. Chapter 12, 2013.

[42] A. Haller *et al.*, "The SOSA/SSN ontology: a joint WeC and OGC standard specifying the semantics of sensors observations actuation and sampling," vol. 1, pp. 1-19, 2018.

[43] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. J. J. o. W. S. Lefrançois, "SOSA: A lightweight ontology for sensors, observations, samples, and actuators," vol. 56, pp. 1-10, 2019.

[44] T. Elsaleh, S. Enshaeifar, R. Rezvani, S. T. Acton, V. Janeiko, and M. J. S. Bermudez-Edo, "IoT-Stream: A lightweight ontology for internet of things data streams and its use with data analytics and event detection services," vol. 20, no. 4, p. 953, 2020.

[45] F. Lécué and J. Z. Pan, "Predicting knowledge in an ontology stream," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013: Citeseer.

[46] Amazon. (2020). *¿Qué son los datos de streaming?. [En línea] Disponible en: https://aws.amazon.com/es/streaming-data/ [Consultado el 5 de junio de 2020]*.

[47] F. Gao, M. I. Ali, and A. J. c. Mileo, "Semantic Discovery and Integration of Urban Data Streams⋆," vol. 7, p. 16, 2014.

[48] M. Fossati, E. Dorigatti, and C. J. S. W. Giuliano, "N-ary relation extraction for simultaneous T-Box and A-Box knowledge base augmentation," vol. 9, no. 4, pp. 413-439, 2018.

[49] A. Lawan and A. J. a. p. a. Rakib, "The semantic web rule language expressiveness extensions-a survey," 2019.

[50] E. J. h. w. w. o. T. r.-s.-q. Prud'hommeaux, "SPARQL query language for RDF, W3C recommendation," 2008.

[51] M. Koubarakis and K. Kyzirakos, "Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL," in *Extended Semantic Web Conference*, 2010, pp. 425-439: Springer.

[52] A. F. Dia, Z. Kazi-Aoul, A. Boly, and Y. Chabchoub, "C-SPARQL extension for sampling RDF graphs streams," in *Advances in Knowledge Discovery and Management*: Springer, 2018, pp. 23-40.

[53] A. Bolles, M. Grawunder, and J. Jacobi, "Streaming SPARQL-extending SPARQL to process data streams," in *European Semantic Web Conference*, 2008, pp. 448-462: Springer.

[54] J. P. C. Pérez, "Ontology-based access to sensor data streams," Universidad Politécnica de Madrid, 2013.

[55] D. F. Barbieri, D. Braga, S. Ceri, E. D. VALLE, and M. J. I. J. o. S. C. Grossniklaus, "C-SPARQL: a continuous query language for RDF data streams," vol. 4, no. 01, pp. 3-25, 2010.

[56] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic, "EP-SPARQL: a unified language for event processing and stream reasoning," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 635-644.

[57] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and G. A. Ruberti, "Ontology-Based Data Access and Integration," ed, 2018.
[58] E. Kharlamov *et al.*, "Towards analytics aware ontology based access to static and streaming data," in *International Semantic Web Conference*, 2016, pp. 344-362: Springer.
[59] S. A. Gómez and P. R. Fillottrani, "Completitud de los métodos de acceso a datos basado en ontologías: enfoques, propiedades y herramientas," in *XIX Workshop de Investigadores en Ciencias de la Computación*, 2017.
[60] C. Nikolaou, E. V. Kostylev, G. Konstantinidis, M. Kaminski, B. C. Grau, and I. J. a. p. a. Horrocks, "The bag semantics of ontology-based data access," 2017.
[61] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A knowledge-based approach for real-time iot data stream annotation and processing," in *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, 2014, pp. 215-222: IEEE.
[62] M. I. U. Fassler and A. S. C. Barahona, "ANÁLISIS DE DATA MINING PARA LA TOMA DE DECISIONES EN LA UNIDAD DE NIVELACIÓN Y ADMISIÓN A NIVEL UNIVERSITARIO."
[63] M. M. Gaber *et al.*, "Internet of Things and data mining: From applications to techniques and systems," vol. 9, no. 3, p. e1292, 2019.
[64] G. VERDUZCO REYES, E. Bautista Thompson, J. A. Ruiz Vanoye, and A. Fuentes Penna, "Modelos de tecnologías del Big Data Analytics y su aplicación en salud," 2017.
[65] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, S. J. J. o. K. S. U.-C. Belfkih, and I. Sciences, "Big Data technologies: A survey," vol. 30, no. 4, pp. 431-448, 2018.
[66] S. Mansalis, E. Ntoutsi, N. Pelekis, Y. J. S. A. Theodoridis, and D. M. T. A. D. S. Journal, "An evaluation of data stream clustering algorithms," vol. 11, no. 4, pp. 167-187, 2018.
[67] S. Guha and N. Mishra, "Clustering data streams," in *Data stream management*: Springer, 2016, pp. 169-187.
[68] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*, 2003, pp. 81-92: Elsevier.
[69] A. Zhou, F. Cao, W. Qian, C. J. K. Jin, and I. Systems, "Tracking clusters in evolving data streams over sliding windows," vol. 15, no. 2, pp. 181-214, 2008.
[70] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. J. J. o. E. A. Sohler, "Streamkm++ a clustering algorithm for data streams," vol. 17, pp. 2.1-2.30, 2012.
[71] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*, 2006, pp. 328-339: SIAM.
[72] L.-x. Liu, H. Huang, Y.-f. Guo, and F.-c. Chen, "rDenStream, a clustering algorithm over an evolving data stream," in *2009 international conference on information engineering and computer science*, 2009, pp. 1-4: IEEE.
[73] C. Ruiz, E. Menasalvas, and M. Spiliopoulou, "C-denstream: Using domain knowledge on a data stream," in *International Conference on Discovery Science*, 2009, pp. 287-301: Springer.
[74] J. Ren and R. Ma, "Density-based data streams clustering over sliding windows," in *2009 Sixth international conference on fuzzy systems and knowledge discovery*, 2009, vol. 5, pp. 248-252: IEEE.
[75] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel, "Density-based projected clustering over high dimensional data streams," in *Proceedings of the 2012 SIAM international conference on data mining*, 2012, pp. 987-998: SIAM.
[76] J. Lin and H. Lin, "A density-based clustering over evolving heterogeneous data stream," in *2009 ISECS international colloquium on computing, communication, control, and management*, 2009, vol. 4, pp. 275-277: IEEE.
[77] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 133-142.
[78] C. Jia, C. Tan, and A. Yong, "A grid and density-based clustering algorithm for processing data stream," in *2008 Second International Conference on Genetic and Evolutionary Computing*, 2008, pp. 517-521: IEEE.
[79] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. J. A. T. o. K. d. f. D. Zhang, "Density-based clustering of data streams at multiple resolutions," vol. 3, no. 3, pp. 1-28, 2009.
[80] J. Ren, B. Cai, and C. J. J. o. C. I. T. Hu, "Clustering over data streams based on grid density and index tree," vol. 6, no. 1, pp. 83-93, 2011.
[81] X. H. Dang, V. Lee, W. K. Ng, A. Ciptadi, and K. L. Ong, "An EM-based algorithm for clustering data streams in sliding windows," in *International conference on database systems for advanced applications*, 2009, pp. 230-235: Springer.
[82] P. Domingos, G. J. J. o. C. Hulten, and G. Statistics, "A general framework for mining massive data streams," vol. 12, no. 4, pp. 945-949, 2003.

[83] A. Bifet *et al.*, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Proceedings of the First Workshop on Applications of Pattern Analysis*, 2010, pp. 44-50: PMLR.
[84] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 935-940.
[85] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "Meka: a multi-label/multi-target extension to weka," 2016.
[86] G. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.
[87] T. O. J. N. a. i. m. l. Ayodele, "Types of machine learning algorithms," vol. 3, pp. 19-48, 2010.
[88] F. Osisanwo *et al.*, "Supervised machine learning algorithms: classification and comparison," vol. 48, no. 3, pp. 128-138, 2017.
[89] P. Druzhkov, V. J. P. R. Kustikova, and I. Analysis, "A survey of deep learning methods and software tools for image classification and object detection," vol. 26, no. 1, pp. 9-15, 2016.
[90] J. Read, F. Perez-Cruz, and A. Bifet, "Deep learning in partially-labeled data streams," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 954-959.
[91] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," vol. 17, pp. 25-32, 2012.
[92] B. Alessandro, B. Martin, F. Martin, L. Sebastian, and M. Stefan, "Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model," ed: Springer, 2013.
[93] R. Battle and D. J. S. W. Kolas, "Enabling the geospatial semantic web with parliament and geosparql," vol. 3, no. 4, pp. 355-370, 2012.
[94] D. M. Strong, Y. W. Lee, and R. Y. J. C. o. t. A. Wang, "Data quality in context," vol. 40, no. 5, pp. 103-110, 1997.
[95] N. G. Weiskopf and C. J. J. o. t. A. M. I. A. Weng, "Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research," vol. 20, no. 1, pp. 144-151, 2013.
[96] P. Gonzalez-Gil, A. F. Skarmeta, and J. A. Martinez, "Towards an ontology for iot context-based security evaluation," in *2019 Global IoT Summit (GIoTS)*, 2019, pp. 1-6: IEEE.
[97] Y. Raimond, "A distributed music information system," Queen Mary, University of London, 2008.
[98] K. Belhajjame *et al.*, "Prov-o: The prov ontology: W3c recommendation 30 april 2013," 2013.
[99] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," ed: Stanford knowledge systems laboratory technical report KSL-01-05 and …, 2001.
[100] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, K. J. P. Taylor, and U. Computing, "IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics," vol. 21, no. 3, pp. 475-487, 2017.
[101] J. Fonseca, P. Guillemin, M. Bauer, L. Frost, and G. J. v. Privat, "Context information management (CIM); NGSI-LD API," vol. 1, pp. 1-159, 2018.
[102] L. Vangelista, A. Zanella, and M. Zorzi, "Long-range IoT technologies: The dawn of LoRa™," in *Future access enablers of ubiquitous and intelligent infrastructures*, 2015, pp. 51-58: Springer.
[103] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan, "Evaluation of LoRa and LoRaWAN for wireless sensor networks," in *2016 IEEE SENSORS*, 2016, pp. 1-3: IEEE.

ANEXOS

Anexo I. Estado de las variables en diferentes tipos de cortocircuitos.

Cortocircuito monofásico tierra

$$\begin{bmatrix}\bar{v}_a\\ \bar{v}_b\\ \bar{v}_c\end{bmatrix}=\begin{bmatrix}0\\ \\ \end{bmatrix}\quad\begin{bmatrix}\bar{i}_a\\ \bar{i}_b\\ \bar{i}_c\end{bmatrix}=\begin{bmatrix}0\\ 0\\ \end{bmatrix}$$

$v_0+v_1+v_2=0$

$i_0=i_1=i_2=i_a/3$

Cortocircuito bifásico a tierra

$$\begin{bmatrix}\bar{v}_a\\ \bar{v}_b\\ \bar{v}_c\end{bmatrix}=\begin{bmatrix}\\ 0\\ 0\end{bmatrix}\quad\begin{bmatrix}\bar{i}_a\\ \bar{i}_b\\ \bar{i}_c\end{bmatrix}=\begin{bmatrix}0\\ \\ \end{bmatrix}$$

$i_0+i_1+i_2=0$

$v_0=v_1=v_2=v_a/3$

Cortocircuito bifásico

$$\begin{bmatrix}\bar{v}_a\\ \bar{v}_b\\ \bar{v}_c\end{bmatrix}=\begin{bmatrix}\\ \bar{v}_c\\ \bar{v}_b\end{bmatrix}\quad\begin{bmatrix}\bar{i}_a\\ \bar{i}_b\\ \bar{i}_c\end{bmatrix}=\begin{bmatrix}0\\ -\bar{i}_c\\ -\bar{i}_b\end{bmatrix}$$

$i_0+i_1+i_2=0$

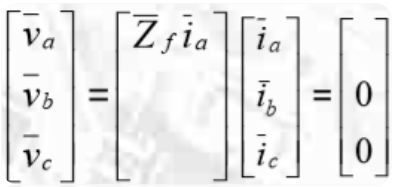
$v_2=v_1 \quad i_0=0$

Cortocircuito trifásico

$$\begin{bmatrix}\bar{v}_a\\ \bar{v}_b\\ \bar{v}_c\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix}\quad\begin{bmatrix}\bar{i}_a\\ \bar{i}_b\\ \bar{i}_c\end{bmatrix}=\begin{bmatrix}\\ \\ \end{bmatrix}$$

$v_0=v_1=v_2=0$

Cortocircuito monofásico a tierra con impedancia Zf



$$\begin{bmatrix} \bar{v}_a \\ \bar{v}_b \\ \bar{v}_c \end{bmatrix} = \begin{bmatrix} \bar{Z}_f \bar{i}_a \end{bmatrix} \begin{bmatrix} \bar{i}_a \\ \bar{i}_b \\ \bar{i}_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$v_0+v_1+v_2 = Z_f i_a$$

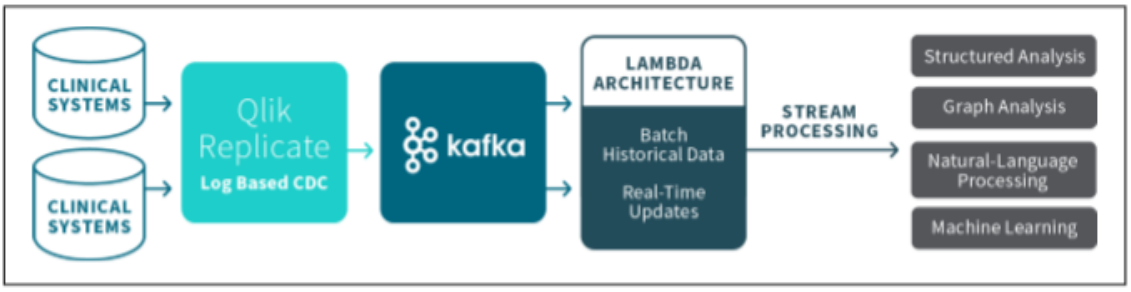
$$i_0=i_1=i_2= i_a/3$$

Anexo II. Extensiones SWRL.

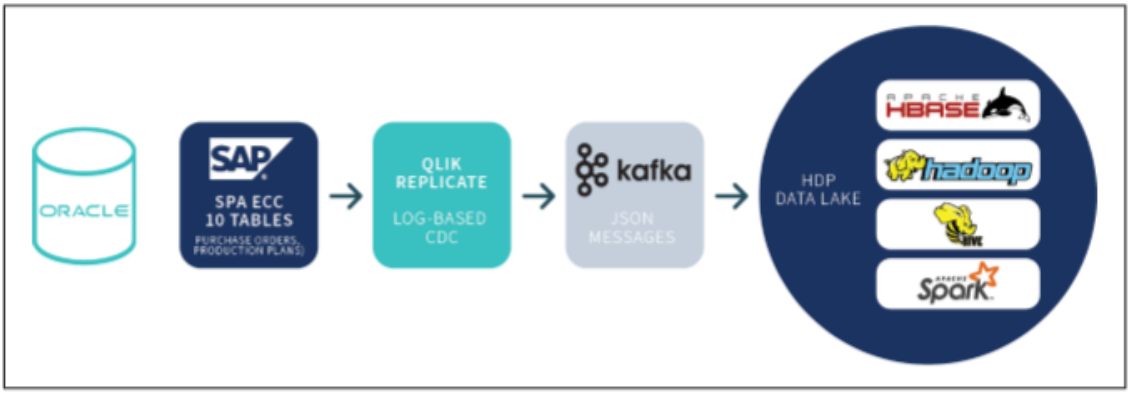
Extensiones de SWRL	Sintaxis agregada	Semántica agregada	Tipo de Extensión
Fuzzy-SWRL (f-SWRL)	clase: C(x)*w propiedad: P(x,y)*w Donde: w ∈ [0,1]	Introduce el valor verdadero "w", para especificar el grado de confidencialidad par los socios en la clase de propiedades	Extensión Fuzzy
Vague-SWRL	Introduce el segundo grado de peso, w_2 para la sintaxis f-SWRL	Denota el segundo grado de socio: w y w_2 se especifica el salto de los socios en la clase de propiedades	Extensión Fuzzy
SWRL Fuzzy (SWRL-F)	Introduce el operador fuzzy: *fuzzymatch*(?x, 'fuzzyvalue')	Coincide las correspondientes variables fuzzy con los valores Fuzzy designados desde los Fuzzy set	Extensión Fuzzy
Fuzzy no monótona SWRL (f-NSWRL)	Fuzzy peso (w), 'Not' and '¬' operadores	Igual a F-SWRL y Extensión de Negación	Fuzzy-no monotónica Extensión
Bayes-SWRL	Probabilidad de peso (p-variable, p)	La variable p (p) hace coincidir la probabilidad	Extensión probabilística

	Clase :C(x)*p_x Propiedad: P(x,y)*p_{xy} Donde: p [0,1]	de afirmaciones de clase o propiedad con valores predefinidos en una tabla de probabilidad condicional (CPT).	
Negación	Operadores 'not' and '¬'	Negación de conceptos existentes y afirmación de hechos negativos.	Extensión no monótona
Cuantificadores SWRL	Operador existente (∃X) Operador NotExists (@X) en antecedentes Operador NotExists (@X) en consecuentes par todo (∀X)	Afirma nuevas instancias de la variable x cuantificada Chequea los hechos pedidos	Extensión no monótona
Dominio	Operadores de dominio: *dominance*(R_x,R_y), R= identificador de regla	Implica: Regla R_x tiene más prioridad en el orden de ejecución que R_y	Ordenado Reglas (Prioridad)
Mutex	Operador mutex: *Mutex*(R_x,R_y), R= identificador de regla	Implica: Regla R_y no será ejecutada en el evento mientras R_x sea ejecutada	Exclusión de Regla
SWRL Extendido (XSWRL)	Operador existencial '!'	Crea nuevos individuos que satisfacen la variable x	Extensión no monótona existencial
SWRL-FOL/ CIF-SWRL	El 'Para todo (∀)' y 'Existencial (∃)' operadores	Impone lógica de primer orden(FOL) operaciones cuantificadoras en las variables SWRL	Extensión FOL
Extensión Epsilon	Constructor ϵXφ(X)	Crea nuevos individuos	Extensión de cuantificación
SWRL Open-Math	La matemática constructiva: *swrlbext:mathext*	Usa las funciones de OpenMath para crear adicionales Built-ins	Match Ext Avanzada
SWRL Temporal	Operaciones de duración, add/subtract operaciones	Implementa temporal operaciones comp predicados en datos valid-time	Extensión Built-in
SWRLM	Complex Mathematical operations (evaluate, square-root, natural log) ej. swrlm : eval(?area, "width * height", ?width, ?height)	Implementa matemáticamente funciones como predicados	Extensión Built-in
SWRLX	Existencial Built-in ej. swrlx:makeOWLThings(?x, ?y)	Crea nuevos individuos de tipo owl: Thing	Extensión Built-in

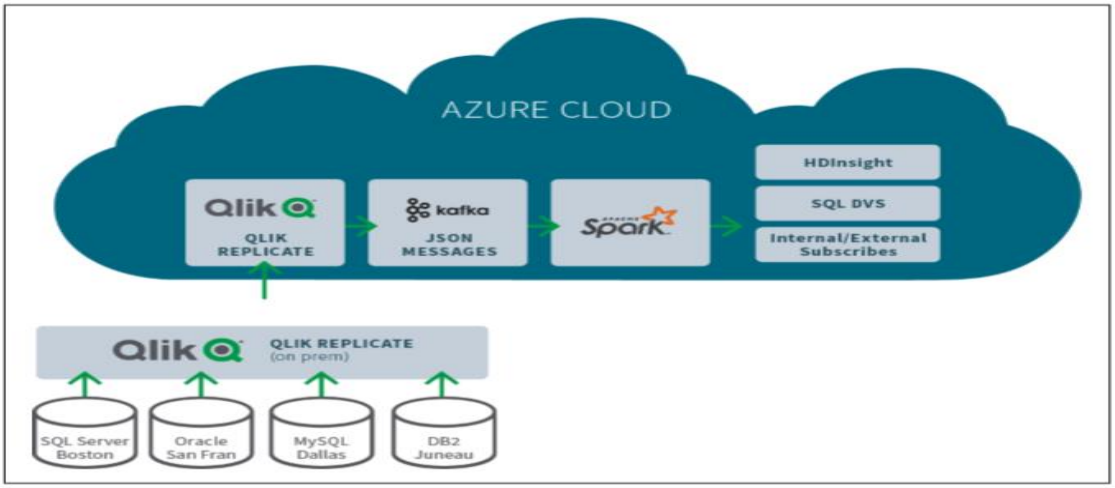
Anexo III. Arquitecturas Stream Data.



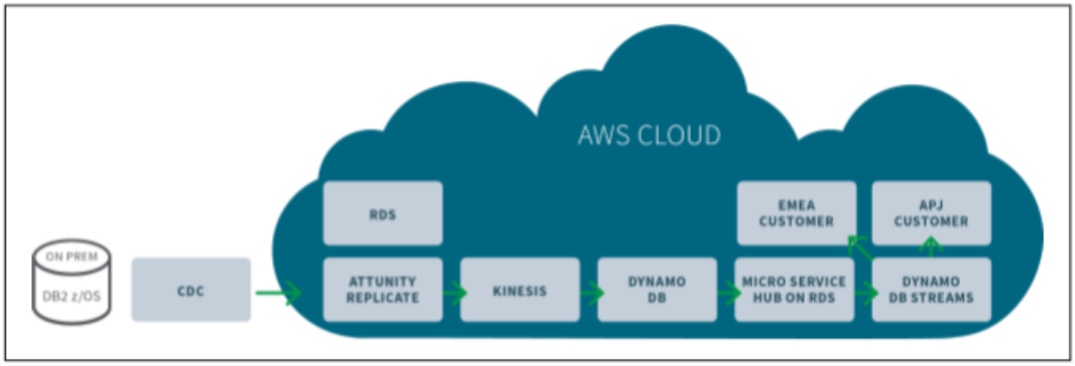
1. Arquitectura de datos de Kafka streaming para la arquitectura Lambda basado en la nube.



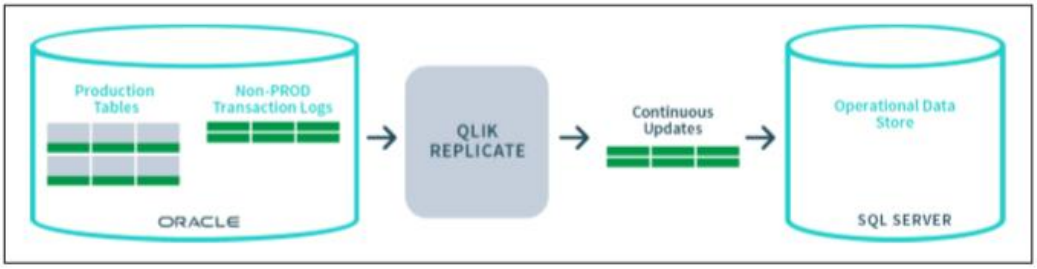
2 Arquitectura de datos de streaming para data lake.



3 Arquitectura de datos para streaming basados en la nube y arquitectura data lake.

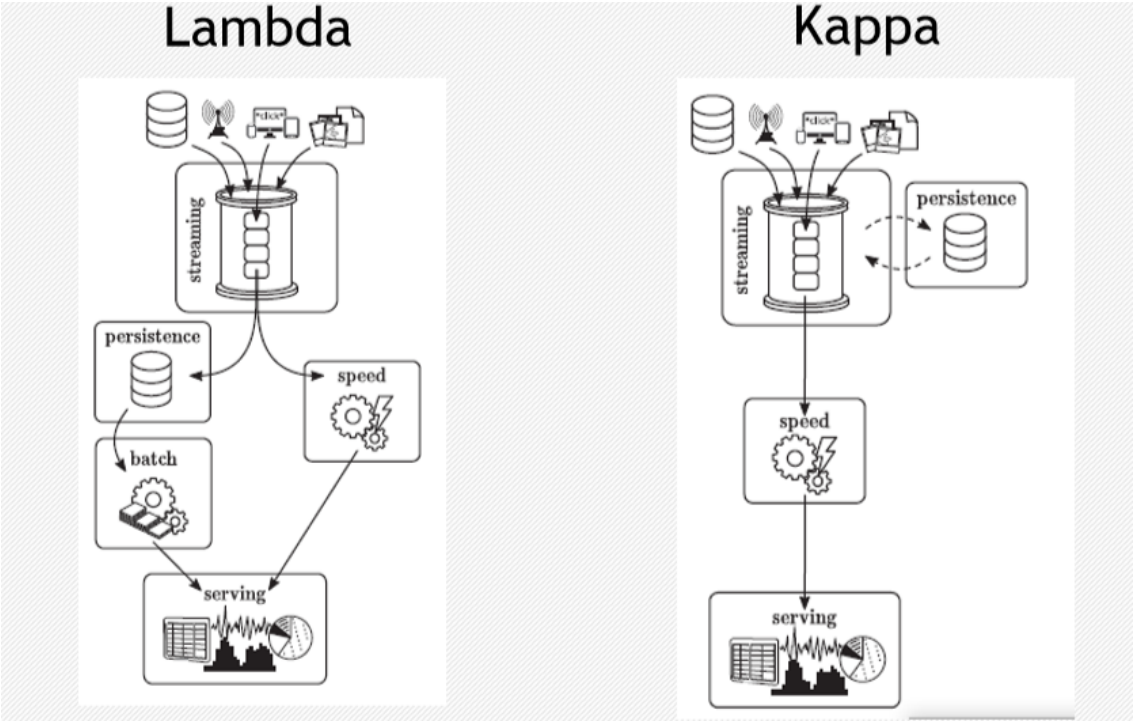


4 Arquitectura de datos para microservicios respaldados en la nube.



5 Arquitectura de datos para real-time Operational Data Store (ODS).

Anexo IV. Arquitectura Lambda vs Kappa.



Anexo V. Algoritmos para el análisis de datos.

Propósito	Algoritmo de análisis
Filtrado Colaborativo	***Filtrado colaborativo basado en el usuario***.- Permiten generar un perfil de usuario en base a su comportamiento digital y luego encontrar otros usuarios con gustos similares. ***Filtrado colaborativo basado en Elemento*.-** Permite que un usuario encuentre elementos similares que gustaron a otro usuario.
Clasificación	***Naive Bayes***.- Es un clasificador estadístico, basado en el teorema de Bayes. Ayuda a predecir si una entidad pertenece a una clase. Puede manejar grandes cantidad de datos con rapidez y precisión. ***Bosque aleatorio***.- Clasificador para grandes volúmenes de datos, introduce la aleatoriedad en cada clasificador individual. Parte de lo que se conoce como árbol de decisión. ***Modelos ocultos de markov***.- Permite encontrar parámetros desconocidos (ocultos) a partir de parámetros observables. Se puede ver como un doble proceso estocástico. ***Perceptron multicapa***.- Se considera un algoritmo que realiza un aprendizaje por corrección de error. Permite un alto grado de conectividad en sus capas o neuronas. Puede usarse para predecir estancias hospitalarias de pacientes.
Agrupamiento (clustering)	***K-Means clustering***.- Permite dividir un conjunto de datos en subconjuntos (clusters). Generalmente, se emplea la distancia

	Euclidiana para establecer la cercanía del cluster a un centroide y de esta manera identificar a qué clúster pertenece un elemento. ***Fuzzy k-Means***.- Los elementos de un conjunto de datos tienen un grado de pertenencia a un clúster. Los elementos no son dicotómicos al no estar sujetos a pertenecer de forma definitiva a un clúster. ***Spectral Clustering***.- Usa la matriz de similitud de datos para reducir dimensionalidad. Los datos son representados como un grafo, la unión de los vértices tiene un peso. A mayor peso, se interpreta como mejor similitud, menor peso, menor similitud.

Anexo VI. Algoritmo K-Means.

```
import numpy as np    # Se utiliza para hacer cálculos científicos
import pandas as pd   # Ayuda en el análisis de datos
import matplotlib.pyplot as plt  # Función "pyplot" de matplotlib. Impresión de
gráficos en alta calidad
from sklearn.cluster import KMeans    # Importar la función KMeans que se encuentra
en el módulo "cluster" del paquete "sklearn"

cortocircuito = pd.read_csv('MUCHAS.csv')    # Importa el archivo a analizar
cortocircuito_variables =
cortocircuito.drop(['Sensor','_id','StreamStart','StreamEnd','Zf','lat','long','alt
','frequency','idl','idp','Point'], axis =1)  #Creamos una nueva variable la cual
contenga los datos del archivo csv, seleccionando solo las columnas que me hacen
falta
cortocircuito_norm = ((cortocircuito_variables -
cortocircuito_variables.min())/(cortocircuito_variables.max() -
cortocircuito_variables.min())) #normalizo los valores

wcss = []   # Búsqueda de la cantidad optima de cluster
for i in range (1,11) :
    kmeans = KMeans(n_clusters = i, max_iter=300)
    kmeans.fit(cortocircuito_norm)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11), wcss)
plt.title("Codo de Jambú")
plt.xlabel('Número de Clusters')
plt.ylabel('WCSS')   # WCSS es un indicador de que tan similares son los individuos
dentro de los clusters


clustering = KMeans(n_clusters= 2, max_iter= 300) #Crea el modelo
clustering.fit(cortocircuito_norm) #Aplica el modelo a la base de datos
cortocircuito['KMeans_Clusters'] = clustering.labels_ #Los resultados del
clustering se guardaran en labels dentro del modelo


from sklearn.decomposition import PCA # Para visualizar los clusters que se
formaran
pca = PCA(n_components=2)
pca_cortocircuito = pca.fit_transform(cortocircuito_norm)
```

```
pca_cortocircuito_df = pd.DataFrame(data= pca_cortocircuito,
columns=['Componente_1','Componente_2'])
pca_nombres_cortocircuito = pd.concat([pca_cortocircuito_df,
cortocircuito[['KMeans_Clusters']]], axis=1)

fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Componente 1',fontsize = 15)
ax.set_ylabel('Componente 2',fontsize = 15)
ax.set_title('Componentes Principales', fontsize = 20)

color_theme = np.array(["blue", "green","orange","red","yellow","grey"])
ax.scatter(x = pca_nombres_cortocircuito.Componente_1, y =
pca_nombres_cortocircuito.Componente_2,
           c = color_theme[pca_nombres_cortocircuito.KMeans_Clusters], s= 50)
plt.show()
cortocircuito.to_csv("add_clusters.csv")
```

Anexo VII. Grafo R2RML utilizado.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix CSV: <http://localhost:8890/schemas/CSV/> .
@prefix csv-stat: <http://localhost:8890/CSV/stat#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix scovo: <http://purl.org/NET/scovo#> .
@prefix aowl: <http://bblfish.net/work/atom-owl/2006-06-06/> .
@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
@prefix iot-stream: <http://purl.org/iot/ontology/iot-stream#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix qoi: <https://w3id.org/iot/qoi#> .
@prefix qu: <http://purl.oclc.org/NET/ssnx/qu/qu#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#TriplesMapOBSERVATION_csv> a rr:TriplesMap; rr:logicalTable
[ rr:tableSchema "CSV" ;
rr:tableOwner "DBA" ;
rr:tableName "OBSERVATION_csv" ];
rr:subjectMap [ rr:termType rr:IRI  ;
rr:template "http://localhost:8890/CSV/observation_csv/ID/{ID}#this";
```

```
rr:class CSV:OBSERVATION_csv; rr:graph <http://localhost:8890/CSV#> ];
rr:predicateObjectMap [ rr:predicateMap [ rr:constant CSV:id ] ;
rr:objectMap [ rr:column "ID" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant CSV:_id ] ;
rr:objectMap [ rr:column "_id" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant iot-stream:windowStart ] ;
rr:objectMap [ rr:column "StreamStart" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant iot:stream: windowEnd ] ;
rr:objectMap [ rr:column "StreamEnd" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant sosa:Sensor ] ;
rr:objectMap [ rr:column "Sensor" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:va ] ; rr:objectMap [
rr:column "Va" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:vb ] ; rr:objectMap [
rr:column "Vb" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:vc ] ; rr:objectMap [
rr:column "Vc" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:ia ] ; rr:objectMap [
rr:column "Ia" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:ib ] ; rr:objectMap [
rr:column "Ib" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:ic ] ; rr:objectMap [
rr:column "Ic" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:zf ] ; rr:objectMap [
rr:column "Zf" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant geo:lat ] ; rr:objectMap [
rr:column "lat" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant geo:long ] ; rr:objectMap [
rr:column "long" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant geo:alt ] ; rr:objectMap [
rr:column "alt" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant CSV:frequency ] ;
rr:objectMap [ rr:column "frequency" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:idl ] ; rr:objectMap [
rr:column "idl" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:idp ] ; rr:objectMap [
rr:column "idp" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant geo:Point ] ; rr:objectMap [
```

```
rr:column "Point" ]; ] ;
rr:predicateObjectMap [ rr:predicateMap [ rr:constant aa:KmeansCluster ] ;
rr:objectMap [ rr:column "KMeans_Clusters" ]; ] .
```

Anexo VIII. Agregar marcadores en Folium.

```
import folium
import pandas as pd

coordenadas = pd.read_csv("querry.csv" )
zulueta_map = folium.Map(location=[22.371873,-79.566525], zoom_start = 14)

folium.Marker(location=[22.368198,-79.574087],
popup="Planta_Eléctrica",icon=folium.Icon(color='red')).add_to(zulueta_map)

for index, franchise in coordenadas.iterrows():
    location = [franchise['lat'], franchise['long']]
    folium.Marker(location, popup = f'Sensor:{franchise["Sensor"]}\n
Cortocircuito:{franchise["Tipo_Falla"]}\n Altitud(m):{franchise["alt"]}\n
ID_Línea:{franchise["idl"]}\n ID_Poste:{franchise["idp"]} \n
Inicio:{franchise["StreamStart"]} \n
Final:{franchise["StreamEnd"]}').add_to(zulueta_map)
zulueta_map.save("zulueta_map.html")
```

Anexo IX. Mapa de calor.

```
from folium.plugins import HeatMap
import folium
import pandas as pd

coordenadas = pd.read_csv("querry.csv" )

zulueta_map = folium.Map(location=[22.371873,-79.566525], zoom_start = 14)
folium.Marker(location=[22.368198,-79.574087],
popup="Planta_Eléctrica",icon=folium.Icon(color='red')).add_to(zulueta_map)

for index, franchise in coordenadas.iterrows():
    location = [franchise['lat'], franchise['long']]
    folium.Marker(location, popup = f'Sensor:{franchise["Sensor"]}\n
Cortocircuito:{franchise["Tipo_Falla"]}\n Altitud(m):{franchise["alt"]}\n
ID_Línea:{franchise["idl"]}\n ID_Poste:{franchise["idp"]} \n
Inicio:{franchise["StreamStart"]} \n
Final:{franchise["StreamEnd"]}').add_to(zulueta_map)
```

```
nrb1=HeatMap(data=coordenadas[['lat','long']].groupby(['lat','long']).sum().reset_i
ndex().values.tolist(), radius=40, max_zoom=50).add_to(zulueta_map)
nrb1.save("Heatmap.html")
```

Anexo X. Unión de los marcadores.

```
import folium
import pandas as pd
from folium.plugins import MarkerCluster

coordenadas = pd.read_csv("querry.csv" )
zulueta_map = folium.Map(location=[22.371873,-79.566525], zoom_start = 14)
market_clusters = MarkerCluster().add_to(zulueta_map)

folium.Marker(location=[22.368198,-79.574087],
popup="Planta_Eléctrica",icon=folium.Icon(color='red')).add_to(zulueta_map)

for index, franchise in coordenadas.iterrows():
    location = [franchise['lat'], franchise['long']]
    folium.Marker(location, popup = f'Sensor:{franchise["Sensor"]}\n
Cortocircuito:{franchise["Tipo_Falla"]}\n Altitud(m):{franchise["alt"]}\n
ID_Línea:{franchise["idl"]}\n ID_Poste:{franchise["idp"]} \n
Inicio:{franchise["StreamStart"]} \n
Final:{franchise["StreamEnd"]}').add_to(market_clusters)
zulueta_map.add_child(market_clusters)
zulueta_map.save("market_clusters.html")
```