# Consultas SparQl a Estructuras RDF

BRYAN ASMAT, DANILO BLAS, VICTOR CHAVEZ, WILLIAM GUADALUPE
Universidad Nacional de Ingeniería
Lima,Perú
basmatf@uni.pe, iblass@uni.pe , vchavezb@uni.pe , wguadalupeq@uni.pe

*Resumen*—**En el presente trabajo se implementaron consultas de tipos SparQL que permiten hacer la búsqueda en distintas fuentes de datos como estructuras RDF,la creación de esta estructura, asi como un resumen de los pasos que realizamos para la ejecución del proyecto.**

**Para iniciar el proyecto se usó la base de conocimiento de la Wikidata realizando consultas online https://query.wikidata.org/ con lo cual seguiremos los siguientes pasos para que se entienda como se ha creado la estructura RDF. Accedemos al enlace mencionado anteriormente.Podemos usar las consultas con extensión .rq que estan almacenadas en el repositorio de este proyecto: Si usamos el contenido del archivo onlineQuerySELECT.rq copiaremos la información y la usaremos en el servicio de consultas de la Wikidata nos devuelve la tabla que hemos construido mediante las consultas que hemos hecho. Si usamos el contenido del archivo onlineQueryCONSTRUCT.rq copiaremos la información y la usaremos en el servicio de consultas de la Wikidata nos devuelve como serían las tripletas en una nueva estructura RDF que vamos a manejar. Descargamos el archivo TSV detallado que se genera al usar el archivo onlineQueryCONSTRUCT.rq y lo modificamos para que tenga el formato Turtle a usar en el presente proyecto. El archivo RDF en formato turtle una vez generado y se encuentra en el siguiente en el directorio RDF con el siguiente nombre RDF.ttl. Hasta ahora los pasos que hemos hecho es para extraer una estructura RDF nueva de Wikidata realizado consultas SparQL para obtener datos de investigaciones y trabajos científicos, digamos obras, que se basen sobre el tema de Ciencias de la Computación. De estos datos hemos extraído las siguiente características:**

**Titulo,Autor,Tema, Idioma, Revista, Fecha de Publicación, URL.**

## I. INTRODUCCIÓN

La información en la web se puede representar bajo un formato de datos llamado RDF. El consorcio World Wide Web (W3C) propuso una especificación llamada SparQL(lenguaje de consultas) para manipular grafos compuesstos por tripletas RDF en la web y en repositorios de datos, asi como extraer datos mediante consultas a relaciones desconocidas y relaizar búsquedas con combinaciones complejas de base de datos.

La estructura de los resultados de una consulta SPARQL es una tabla donde cada fila es un resultado y cada columna se corresponde con una de las variables de la consulta.

Una consulta SparQL consta de tres partes:

- Coincidencia de patrones: optional, union, nesting,filtering.
- Modificadores de solución: projection, distinct, order, límit, offset.
- Parte de salida: construcción de nuevas tripletas.

Las búsquedas se realizan a través de un SparQL Endpoint, que es la interfaz de entrada y salida de estas bases de datos y que permitirá nuestras consultas. Dicho de otra forma, los Endpoints de SparQL no son más que una dirección web (URL) que acepta peticiones de búsqueda SparQL y devuelve resultados, las formas de consulta SparQL son select, construct, ask y describe.

**¿Cuáles son los desafíos de diseño en SparQL?**

SparQL debe tener en cuenta las características distintivas de RDF:

- Debería poder extraer información de grafos RDF interconectados.
- Debe ser coherente con la semántica de mundo abierto de RDF.
- Debe ofrecer la posibilidad de agregar información opcional si está presente.
- Debería ser capaz de interpretar correctamente grafos RDF con un vocabulario con semántica predefinida.
- Debería ofrecer algunas funcionalidades para navegar en un grafo RDF.

## II. ¿QUE ES LA WEB SEMÁNTICA?

La Web Semántica vendría a ser una extensión de la Web actual dotada de significado, esto es, un espacio donde la información tendría un significado bien definido, de manera que pudiera ser interpretada tanto por agentes humanos como por agentes computarizados.

La web Semántica ha sido impulsada, entre otros, por el creador de la web, Tim Berners Lee en el 2000 para que la información sea reunida de forma que los buscadores puedan ”comprender”, más allá de limitarse a colocar los documentos en una lista

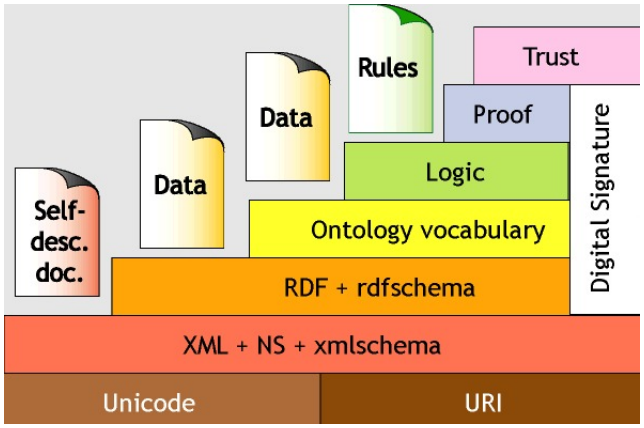


Figura 1. Web semántica [7]

## III. URLs, URIs, IRIs

### *III-A. URLs*

El LRU es una cadena de caracteres con la que se asigna una dirección única a cada uno de los recursos de información disponibles en Internet. Existe un URL único para cada página de cada uno de los documentos de la WWW

### *III-B. URIs*

Un identificador de recursos uniforme es una cadena de caracteres que identifica los recursos de una red de forma unívoca. La diferencia respecto a un localizador de recursos uniforme (URL) es que estos últimos hacen referencia a recursos que, de forma general, pueden variar en el tiempo.

Normalmente estos recursos son accesibles en una red o sistema. Los URI pueden ser localizador de recursos uniforme (URL),(URN)

### *III-C. IRIs*

Es un estándar de protocolo de Internet que se basa en el protocolo Identificador uniforme de recursos (URI) al expandir en gran medida el conjunto de caracteres permitidos.

## IV. Estructura RDF

RDF es un método general para descomponer cualquier tipo de conocimiento en trozos pequeños, con algunas reglas acerca de la semántica o significado, de esas piezas.

El punto es tener un método tan simple que puede expresar cualquier hecho, y a la vez tan estructurada que las aplicaciones informáticas pueden hacer cosas útiles con él. Attention Is All You Need [1]

RDF representa la información en una declaración triple -también llamada tripleta- que sigue la estructura sujeto-predicado-objeto Aqui citaremos las paginas [1]

**Ejemplo**

<La ronda de noche><fue creada por><Rembrandt>

En este ejemplo**<La ronda de noche>** y el objeto **<Rembrandt>** se pueden considerar como dos nodos en un grafo donde el predicado **<fue creada por>** define una arista o la relación entre estos .

Figura 2. Encoder-Decoder del Transformer. Fuente The Transformer: Attention Is All You Need. [1]

## V. Buscando RDF con SPARQL

SPARQL nos permite traducir datos en grafo, intensamente enlazados, en datos normalizados en formato tabular, esto es, distribuidos en filas y columnas, que se pueden abrir en formato como .csv .json .ttl ,etc o también se pueden ver en programas de visualización

Resulta útil pensar las consultas SPARQL como un conjunto de oraciones con espacios en blanco. La base de datos tomará esta consulta y encontrará cada conjunto de oraciones que encaje correctamente en estos espacios en blanco, devolviéndonos los valores coincidentes como una tabla.

## VI. Consultas SPARQL

SPARQL, acrónimo de SPARQL *Protocol and RDF Query Language*, es un lenguaje estandarizado para hacer consultas en estructuras RDF. Es una tecnología clave en el desarrollo de la web semántica que se constituyó como recomendación oficial del W3C el 15 de enero de 2008, siendo actualizado a la versión 1.1 en 2013 [1]. En un principio SPARQL únicamente incorpora funciones para la recuperación sentencias RDF. Sin embargo, algunas propuestas también incluyen operaciones para el mantenimiento (creación, modificación y borrado) de datos.

### *VI-A. Especificaciones*

SPARQL 1.0:

- SPARQL como lenguaje de consultas para RDF, cubre la sintaxis de las consultas (QL).
- SPARQL como protocolo para RDF especifica cómo un programa debe pasar consultas SPARQL a un servicio de procesamiento de consultas SPARQL y cómo este servicio las retorna.
- Formato XML de resultados de consultas SPARQL, describe un formato simple XML para que los procesadores de consultas lo utilizen al devolver resultados.

Sparql 1.1:

- La especificación de consulta federada describe como una simple consulta puede traer data de multiples fuentes, esto hace más fácil construir aplicaciones que toman ventaja de entornos distribuidos.
- La especificación de actualización es la diferencia clave entre SPARQL 1.0 y 1.1 porque lleva a

SPARQL de ser un lenguaje de consulta a algo que puede agregar datos a un conjunto de datos, reemplazarlos y eliminarlos también.
- La especificación de Descripción del servicio describe cómo un programa cliente puede preguntarle a un motor SPARQL exactamente qué características admite.
- La especificación del formato JSON de los resultados de la consulta describe un equivalente JSON del formato XML de los resultados de la consulta.
- La especificación de los formatos CSV y TSV de los resultados de la consulta describe los equivalentes de valores separados por comas y tabuladores del formato XML de los resultados de la consulta.
- La especificación del Protocolo HTTP de Graph Store amplía el Protocolo SPARQL con una API similar a REST para la comunicación entre un cliente y un procesador SPARQL sobre gráficos o conjuntos de tripletas.

### *VI-B. Estructura RDF a utilizar*

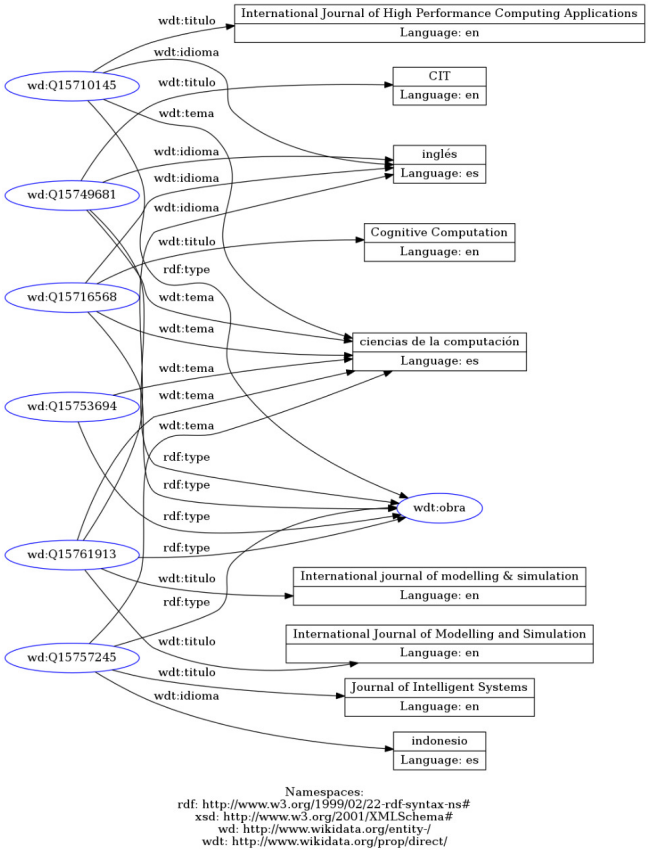


Figura 3. Transformer XL: EGrafo generado, con 30 tripletas de la estructura RDF utilizada [2]

De acuerdo al artículo, las principales diferencias entre el Transformer y el Transformer Universal consistn en que el Transformer Universal aplica el encoder para un número variable de pasos para cada token de entrada/salida (T pasos), mientras que el Transformador aplica exactamente 6 capas de encoder/decoder, respectivamente. Así mismo, el Transformer Universal utiliza una representación de entrada ligeramente diferente: incluye un embedding de paso de tiempo además de la codificación posicional.

Los números variables de pasos presentes en el Transformer Universal se logran mediante el uso de Adaptive Computation Time, un mecanismo propuesto por Alex Graves [6] que permite la aplicación del encoder y decoder un número variable de veces.

### *VI-C. Transformer-XL*

El transformer-XL es uno de los primeros modelos exitosos en abordar el problema de poseer una longitud fija en la secuencia de entrada. En el artículo Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context [7] se propone este método novedoso para el modelado de lenguaje, que permite que una arquitectura transformer aprenda dependencia a largo plazo, a través de un mecanismo de recurrencia, más allá de una longitud fija sin alterar la coherencia temporal.

Se introduce un mecanismo recurrente a nivel de segmento que permite que el modelo reutilice estados ocultos anteriores en el momento del entrenamiento, abordando tanto los problemas del contexto de longitud fija como la fragmentación del contexto. En otras palabras, la información histórica se puede reutilizar y se puede extender tanto como lo permita la memoria de la GPU.

Para reutilizar adecuadamente los estados ocultos, los autores proponen un mecanismo llamado codificaciones posicionales relativas que ayuda a evitar la confusión temporal. Los modelos actuales no pueden distinguir la diferencia posicional entre entradas en diferentes segmentos en diferentes capas. La codificación de posición relativa soluciona este problema al codificar el sesgo de información posicional en los estados ocultos, que difiere de otros enfoques que realizan esto como el nivel de entrada.

Figura 4. Transformer XL: Entrenamiento y Evaluación. Fuente: Transformer XL [7]

El transformer-XL reduce el puntaje de perplejidad SoTA anterior en varios conjuntos de datos como text8, enwiki8, One Billion Word y WikiText-103. Los autores afirman que el método es más flexible, más rápido durante la evaluación (aceleración de 1874 veces), se generaliza bien en conjuntos de datos pequeños y es eficaz para modelar secuencias cortas y largas.

Así mismo, los autores proponen una nueva métrica llamada Relative Effective Context Length que propor-

ciona una manera justa de comparar modelos que se prueban con mayores longitudes de contexto.

### *VI-D. XLNet*

XLNet es un modelo presentado en el artículo XLNet: Generalized Autoregressive Pretraining for Language Understanding [8] que es un lenguaje autoregresivo que genera la probabilidad conjunta de una secuencia de tokens basada en la arquitectura del Transformer con recurrencia. Este modelo introduce una variante de modelado de lenguaje llamada modelado de lenguaje de permutación. Los modelos de lenguaje de permutación están entrenados para predecir un token dado el contexto anterior como el modelo de lenguaje tradicional, pero en lugar de predecir los tokens en orden secuencial, predice los tokens en un orden aleatorio.

Además de usar el modelado de lenguaje de permutación, XLNet mejora BERT al usar el Transformer XL como su arquitectura base. XLNet utiliza las dos ideas clave de Transformer XL: embeddings posicionales relativas y el mecanismo de recurrencia. Los estados ocultos del segmento anterior se almacenan en caché y se congelan mientras se realiza el modelado del lenguaje de permutación para el segmento actual. Como todas las palabras del segmento anterior se usan como entrada, no es necesario conocer el orden de permutación del segmento anterior.

### *VI-E. BERT*

Este modelo [9],publicado el 2018, esta conformado por una pila de bloques Transformers, la cual esta preentrenada en un corpus de dominio general con 800 millones de palabras. Este modelo obtiene un mejor rendimiento que modelos anteriores a su publicación debido a su naturaleza bidireccional donde la atención se centra en toda la secuencia.

### *VI-F. RoBERTa*

Es un modelo introducido en Facebook. El enfoque BERT robustamente optimizado RoBERTa [10], es un reentrenamiento de BERT con una metodología de entrenamiento mejorada, un 1000 % más de datos y potencia de cálculo. Para mejorar el procedimiento de entrenamiento, RoBERTa elimina tareas como Next Sentence Prediction (NSP) del entrenamiento previo de BERT e introduce el enmascaramiento dinámico para que el token enmascarado cambie durante las épocas de entrenamiento.

RoBERTa supera a BERT y XLNet en los resultados de referencia de GLUE.

### *VI-G. DistilBERT*

Este modelo aprende una versión destilada (aproximada) de BERT, que retiene el 95 % de rendimiento pero utiliza solo la mitad del número de parámetros. Espec'ificamente, no tiene embeddings de tipo token, pooler y retiene solo la mitad de las capas del BERT de Google.

DistilBERT [11], utiliza una técnica llamada destilation, que se aproxima a BERT de Google. La idea es que una vez que se ha entrenado una gran red neuronal, sus distribuciones de salidas completas se pueden aproximar usando una red más pequeña. Una de las funciones clave de optimización utilizadas para la aproximación posterior en las estadísticas bayesianas es la divergencia de Kulback-Leiber.

## VII. Conclusiones

- Se ha realizado clasificación multiclase utilizando la librería simpletransformers y modelos preentrenados, basados en BERT, obteniendo buenos resultados en métricas de clasificación, especialmente del modelo distilBERT, que tiene un menor número de paramétros que BERT y que utiliza el aprendizaje `teacher-students`, donde se entrena a una red de `students` para imitar la distribución de salida completa de la red del `teacher` (su conocimiento) y retiene el rendimiento de BERT.
- En la tarea de traducción, se realizaron comparaciones entre distintos modelos seq2seq y los modelos transformers, donde estos últimos obtuvieron mejores resultados. Sin embargo el modelo Transformer XL modificado no logró superar al Transformer original.

## Referencias


[1] https://www.w3.org/TR/rdf-sparql-query/.
[2] Página web para generar grafos a partir de una estructura RDF: .
[3] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E.Hinton. Layer Normalization (2016). Disponible en https://arxiv.org/abs/1607.06450.
[4] Kishore Papineni , Salim Roukos , Todd Ward , Wei-jing Zhu (2002). BLEU: a Method for Automatic Evaluation of Machine Translation, pp. 311-318. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).
[5] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. Universal Transformers (2018). ICLR 2019. Disponible en https://arxiv.org/abs/1807.03819
[6] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks (2017). Disponible en https://arxiv.org/abs/1603.08983.
[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context (2019). Disponible en https://arxiv.org/abs/1901.02860.
[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding (2019). Disponible en https://arxiv.org/abs/1906.08237.
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018). Disponible en https://arxiv.org/abs/1810.04805.
[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov- RoBERTa: A Robustly Optimized BERT Pretraining Approach (2019). Disponible en https://arxiv.org/abs/1907.11692.
[11] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2019). Disponible en https://arxiv.org/abs/1910.01108.

[12] Pytorch-Transform:Implementación de transformer para NLP. https://pytorch.org/hub/huggingface_pytorch-transformers/.
[13] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Jamie Brew. Transformers: State-of-the-art Natural Language Processing (2019), HuggingFace Inc. Disponible en https://arxiv.org/abs/1910.03771.
[14] SimpleTransformers https://pypi.org/project/simpletransformers/.
[15] Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification(2015). Disponible en https://arxiv.org/abs/1509.01626.
[16] Desmond Elliott, Stella Frank, Khalil Sima'an, Lucia Specia. Multi30K: Multilingual English-German Image Descriptions(2016). Disponible en https://arxiv.org/abs/1605.00459.
[17] TransformerXL from Scratch notebook.