TALLER 4 Análisis y Enriquecimiento de Información

Erik Baumert, Carlos Carvajal, Areli Moreno

Análisis de Información sobre BigData
Universidad de los Andes, Bogotá, Colombia
{ e.baumert, cj.carvajal, am.morenof }@uniandes.edu.co

Fecha de presentación: mayo 23 de 2017

Tabla de contenido

1 Introducción 1
2 Tecnología usada y Forma de acceso a la aplicación Web resultante 1
3 Resultados de la Recolección de datos 2
4 Estrategia de relacionamiento de la información 2
4.1 Extracción de entidades 2
4.2 Enriquecimiento semántico 3
5 Análisis de resultados 4
6 Conclusiones 4

1 Introducción

Este documento describe brevemente los métodos, tecnologías y resultados del taller.

2 Tecnología usada y Forma de acceso a la aplicación Web resultante

Para llevar a cabo el proceso de recolección de los datos se construyeron los scripts en Python que se describen brevemente en la siguiente sección. El almacenamiento y consulta de la información se realizó en MongoDB. En el proceso de extracción de entidades y relacionamiento de la información se utilizó la Api de *Meaning Cloud*: *Topics Extraction API* 2.0, el lenguaje SPARQL y scripts de Python. La interfaz gráfica se realizó en AngularJS, haciendo uso de *Google Maps Api* para la geolocalización de información.

La arquitectura diseñada para resolver el problema es similar a la presentada en los talleres anteriores, en la cual se expone la información a través de un servicio REST, en este caso se realiza una petición POST a través de un objeto JSON con la estructura {"text":""}, que es el texto que el cliente ingresa por medio de la interfaz gráfica. El valor enviado es el criterio de búsqueda de preguntas en la base de MongoDB. La respuesta del servicio es una lista de preguntas de la base de datos de mongo, cada pregunta tiene dos campos: *summary* (La pregunta realizada) y *entities*, donde esta última es una lista Dentro de *entities* con los tres tipos de objetos que se describen en la tabla 1, asociados a la información correspondiente a Dbpedia. El servicio se expone con la ejecución del script `appLinkedData.py`.

Las fuentes y los métodos usados para la recolección de datos se describen a continuación:

- Personajes del entretenimiento nacidos y muertos en el día: Para la obtención de estos datos se construyó el script `imdbreader.py` que hace uso de las librerías `datetime`, `pymongo`, `feedparser` y `re`. El proceso sigue el algoritmo de la figura1. La base de datos obtenida y almacenada en MongoDB se llama `db.dates`. IMDB 5000 *Movie Dataset.*
- El script `kaggleReader.py` usa las librerías `csv`, `pymongo` y `progressbar`. En este caso se lee el archivo .CSV de Kaggle fila por fila, y se crea un JSON con cada una, que incluye los atributos en MongoDB (db.metadata). Se asigna el tipo de dato de acuerdo a su naturaleza. Finalmente, las 5000 filas resultantes se importan a través de la conexión con MongoDB en una sola ejecución del script, monitoreando el proceso a través de una barra de

progreso. El nombre de la base obtenida es `db.metadata`.

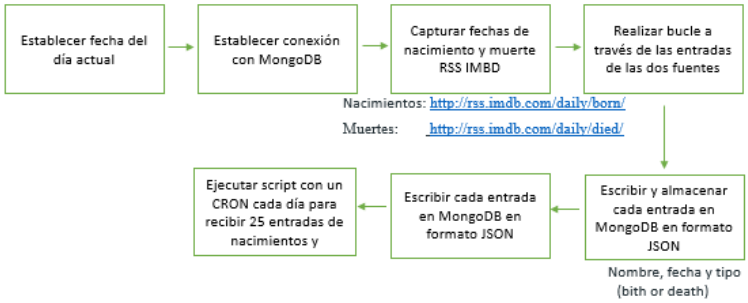

Figura 1. Procedimiento para recolección fechas de nacimiento y muerte

- Preguntas de *Movies & TV Stack Exchange*: Para esta tarea se construyó el script `moviereader.py`, en el cual se descarga el *feed RSS* con las 30 preguntas más recientes de la página. Dado que la información en *Movies & TV Stack Exchange* se actualiza constantemente, el script descarga la información automáticamente 3 veces al día, almacenando las preguntas en MongoDB. En cada ejecución se compara el ID de las preguntas descargadas con las previamente almacenadas de forma que solo se registran las preguntas que aún no se encuentren en la base de MongoDB. Los datos se almacenaron en el archivo `db.questions`.
- DBpedia: Los daros de esta fuente se usaron para realizar el enriquecimiento semántico de las entidades identificadas en las preguntas de *Movies & TV Stack Exchange.* El procedimiento para su recolección se expone en la sección 4.2*xchange.*

3 Resultados de la Recolección de datos

En la tabla 1 se presentan los resultados generales de la recolección de datos. Los datos de dbpedia se almacenaron añadiendo los campos descritos, en la base de preguntas db.questions.

4 Estrategia de relacionamiento de la información

La metodología para relacionar la información parte de la extracción de entidades de los textos asociados a las preguntas de *Movies & TV Stack Exchange*. Posteriormente, se realiza un enriquecimiento semántico de las entidades a partir de la asignación de variables obtenidas de DBpedia.

4.1 Extracción de entidades

Inicialmente se consideraron 3 extractores de entidades: *Yahoo, Ambiverse* y *Meaning Cloud*. La documentación y recursos relacionados con el extractor de Yahoo se encuentran desactualizados y no fue posible realizar una prueba. Ambiverse provee un *Api rest* de aplicación sencilla, sin embargo, presenta restricciones de uso en relación al número de peticiones. Por su parte, *Topics Extraction API* 2.0 de *Meaning Cloud* ofrece un extractor de entidades que permite hasta 20000 peticiones por mes bajo una figura gratuita a partir de la creación de una cuenta. Esta Api proporciona información más completa que los productos anteriores, que incluye Entidades: personas, organizaciones, lugares, URIs, números de teléfono, Conceptos: palabras clave significativas en el texto, Expresiones de tiempo, Expresiones monetarias, Expresiones de cantidad, Citas y Relaciones.

Para realizar este proceso se construyeron los scripts de Python `movies_entity_extractor.py` y `meaning_cloud_entity_extractor.py`. el primero hace uso del segundo para realizar la conexión con la Api de *Meaning Cloud* y almacenar los resultados en MongoDB como se muestra en la figura 2.

	Nombre de la base en MongoDB	Variables almacenadas	Número de registros
Personajes del entretenimiento nacidos y muertos en el día. RSS IMBD	db.dates	*name*: Nombre del personaje *date*: Fecha asociada *type*:Tipo de fecha capturada: "*birth*": Fecha de nacimiento, "*death*": Fecha de muerte	599
Preguntas de *Movies & TV Stack Exchange*	db.questions	*id:* Id de la pregunta *authors*: *href*: Numero de usuario del autor, *name*: nombre o seudónimo del autor, *published*: Fecha y hora de publicación *title* o *summary*: Texto con la pregunta realizada, *category*: Categoria según el tema de la pregunta, *summary*: Resumen de la pregunta	524
IMDB 5000 *Movie Dataset*	db.metadata	*id*: de la película *director_name*: Director de la película, *movie_tittle*: Título, *num_voted_users*: Número de usuarios que la han votado *duration*: Duración de la película, *actor1_name*: Nombre del protagonista, *actor2_name* : Nombre coprotagonista *genres*: Géneros de la pelicula, *plot_keywords*: palabras clave	5043
DBpedia	db.questions	**Persona** *name*: El nombre de la entidad, obligatorio, *relatives*: Familiares *partners*: Parejas y esposas ,*birthplace*: Lugar de nacimiento, opcional, *birthdate*: Fecha de nacimiento. **Localización** *name*: El nombre de la entidad, obligatorio, *type*: 1, obligatorio. capital: Capital del pais, *country*: País donde se encuentra la ciudad, opcional, *lon*: Longitud geoposición, opcional, *lat*: Latitud geoposicion. **Organización** *name*: El nombre de la entidad, obligatorio. *type*: 2, obligatorio, *dbptype*: Tipo de organización según DBpedia, opcional, *semtype*: Tipo de organización según extractor de entidades, opcional, *city:* Ciudad donde se encuentra la organización, opcional, *country*: País donde se encuentra la organización, opcional.	524

Tabla 1. Resultados del procesamiento de recolección de información

Figura 2. Algoritmo `movies_entity_extractor.py`

4.2 Enriquecimiento semántico

Como se mencionó en la sección 2, para hacer el enriquecimiento semántico se hizo uso de DBpedia, la cual provee un api de consulta mediante SPARQL, y para su implementación se usó la librería para python llamada SPARQLWrapper. *SPARQL Protocol and RDF Query Language*, es un lenguaje estandarizado para la consulta de grafos RDF que permite el acceso a información disponible en plataformas web como DBpedia. Incluye operaciones la recuperación de sentencias RDF y para la creación, modificación y borrado de datos. Para este proceso se construyeron dos scripts en Python

`semantic enricher.py` y `db pedia client.py`. El primero hace uso del segundo, el cual recibe una de las entidades identificadas en el paso descrito en la sección anterior y a través de la librería `SPARQLWrapper`, se conecta con la base de DBpedia y realiza la búsqueda de las variables asociadas a esa entidad. El algoritmo básico para realizar este proceso se presenta en la figura 3.

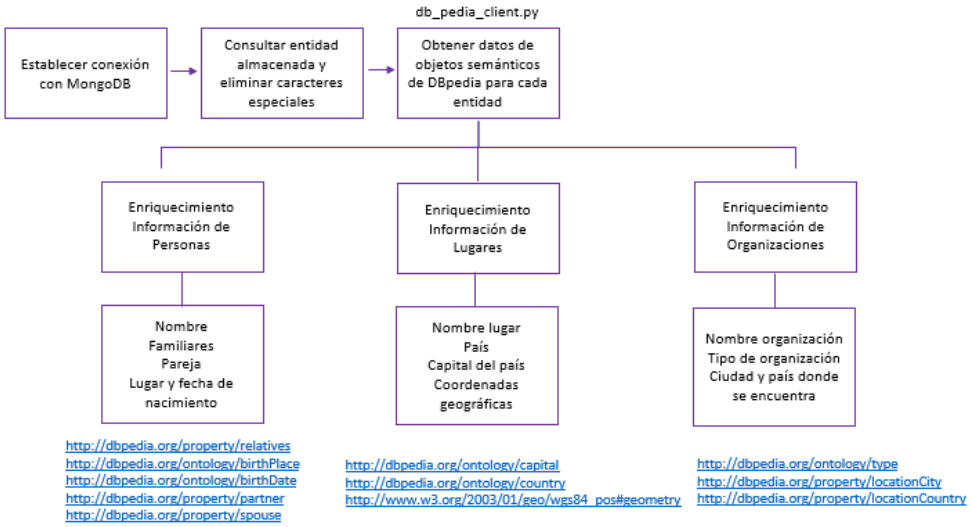

Figura 2. Algoritmo `semantic_enricher.py`

5 Análisis de resultados

Como resultado se obtuvo una base de datos con 533 preguntas, que se relacionaron a través de entidades con variables tomadas de DBpedia. Se observa (tabla 2) que para 2% de las preguntas no se pudo identificar ninguna entidad. Por otro lado, es posible hablar de que existe un porcentaje de "ruido" si se observan las entidades para las cuales no se obtuvo información de DBpedia. Un ejemplo de ello sería la entidad como "Junior", es un nombre que no tiene mayores características de identificación por lo cual no será posible encontrar variables asociadas en DBpedia. Todas aquellas entidades para las que no se encontró información e DBpedia, se denominaron ruido, y se encontró que 37% de las preguntas tienen por lo menos una entidad ruido. Se puede decir, que de la información que se consulta en la interfaz gráfica, el 61% es información fiable, en términos del contenido que ofrece.

	Cantidad	**Porcentaje**
Preguntas sin entidades	9	2%
Preguntas Ruido	196	37%
Total preguntas	**533**	

Tabla 2. Resumen información recolectada

6 Conclusiones

- Se identificó a la *Topics Extraction API* 2.0 como la mejor dentro de las opciones por su facilidad en la aplicación, la completitud de la información y el número de peticiones que se pueden realizar.
- En los procedimientos de relacionamiento semántico existe un desface o error que se produce porque no existe una lógica directa entre las entidades. En ese caso los análisis semánticos presentan una desventaja porque para garantizar una confiabilidad más alta de la información sería necesario establecer estándares semánticos y relaciones de equivalencia entre conceptos. En el caso del ejercicio, sin embargo, se obtuvo un importante porcentaje de información que se logró relacionar.
- Los retos de esta actividad son realizar un proceso que involucre más de dos fuentes para relacionar, y realizar un proceso escalable para la solución.