Universidad Técnica Particular de Loja

Ingeniería en Sistemas Informáticos y Computación

Sistemas Basados en Conocimientos

Proyecto Bimestral - Transformación, enlazado y almacenamiento de datos

- Fuertes Alejandro
- Román Andrés

Definición de URIs y datos origen

Para el siguiente trabajo bimestral hemos considerado el uso de varias URI's para generar el archivo RDF, nuestra fuente de información es la Wikipedia y la DBpedia, en donde podemos encontrar URI's como las que nombramos a continuación:

- **skos:Concept:** http://www.w3.org/2004/02/skos/core
- **owl:Thing:** http://www.w3.org/2002/07/owl
- **dbo:Document:** http://dbpedia.org/ontology/

Transformación y almacenamiento de datos RDF

Los datos encontrados en la wikipedia se encuentran en texto plano, por lo que es sumamente importante tabular estos datos para su posterior uso, es así que mediante varios procesos que realizamos convertimos la información o texto de la wiki en tripletas, las cuales son más fáciles de comprender para el sistema y nos permiten a su vez organizar la información.
A continuación, presentaremos las tablas de los datos tabulados generados para el sistema.

Resumen de datos recolectados

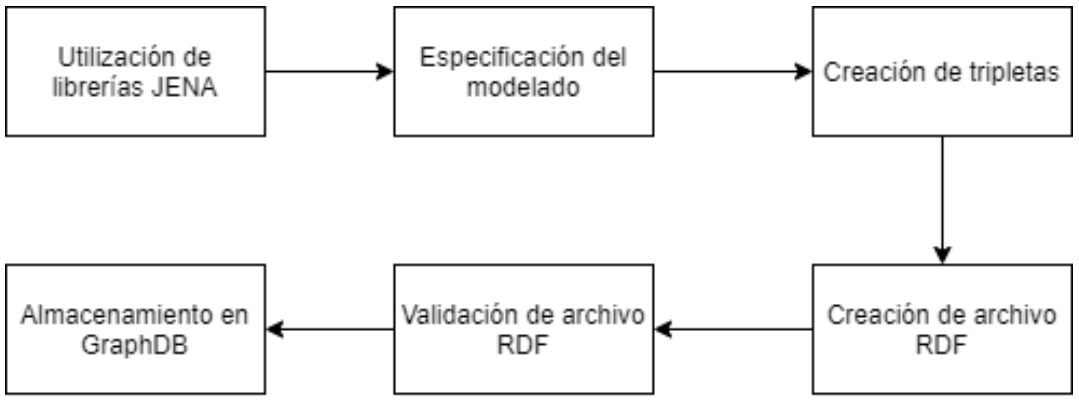
Clase	Número instancias
skos:Concept	652
owl:Thing	6723
dbo:Document	3285

Pre-procesamiento de datos

La primera limpieza de datos que se realizó fue sobre los subconceptos obtenidos en los 3 niveles de recorrido en DBPedia, para ello se recorrió la tabla de ***skos:Concept*** obtenida en un principio y mediante sentencias SQL se realiza selecciones únicas de los subconceptos para que no estén repetidos. Los datos obtenidos se deben limpiar quitando la url de DBpedia y dejando solo el contenido final para que pase a ser el atributo “skos:prefLabel”, para esto se utilizó códigos en python del siguiente Collab, siendo necesario eliminar columnas y realizar un parse de espacios en blanco y caracteres especiales.
La segunda limpieza de datos se realizó sobre los links de Wikipedia obtenidos, para ello se crea una tabla temporal con los IDs y una tabla para las redirecciones de esos mismos links. En este punto se hizo uso de la librería BeautifulSoup que analiza los documentos HTML y crea un árbol con todos sus documentos para extraer la información pertinente, utilizada sobre todo para temas de scrapy, en este caso para el scrapping de la API de Wikipedia. Para esta clase también fue necesario realizar eliminación de columnas trabajando el csv obtenido y haciendo un parse datos.

Transformación de datos

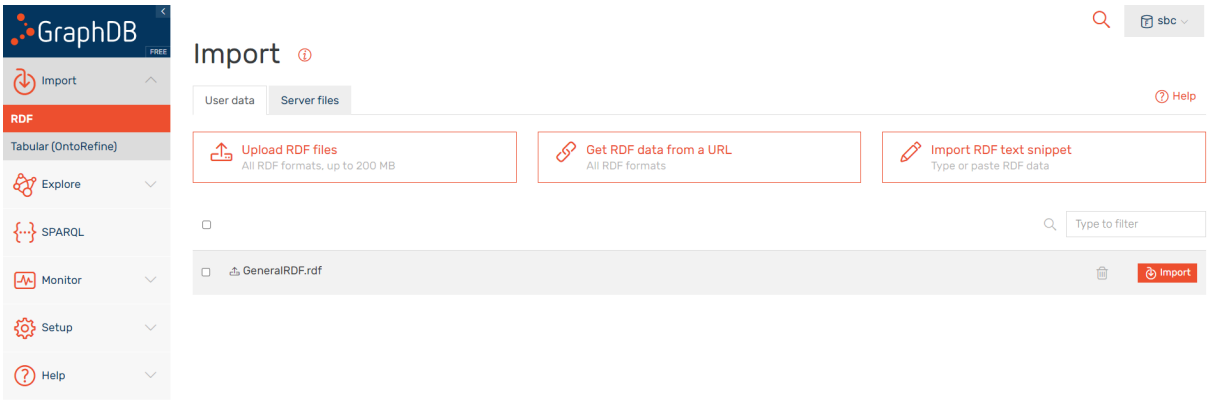
Ahora se procederá con la explicación del proceso con Jena para transformar los datos, explicada en el siguiente esquema:



El primer paso es hacer uso de las librerías que proporciona Jena para la extracción de datos y la producción de archivos en formato RDF, una vez importadas estas librerías se pasa a especificar el modelo que se ha definido para este trabajo, haciendo uso de los prefijos de cada ontología utilizada. El siguiente paso es codificar la creación de tripletas según las clases del modelo y las propiedades de cada una de ellas para que se pueda crear un archivo en formato RDF. El archivo obtenido será validado en la aplicación online de EasyRDF y almacenado en GraphDB que proporciona un clúster de alta disponibilidad.

Almacenamiento

Para almacenar los datos usaremos GraphDB que nos otorga capacidades como almacenamiento, organización y gestión de datos de forma semántica. Además nos da más opciones para trabajar con los archivos RDF.



Enlazado post-transformación

La realización de este apartado no es necesaria en nuestro caso ya que nuestro código implementado ya obtiene las categorías partiendo de un concepto realiza hasta 3 subniveles como se muestra en el código siguiente y en el siguiente archivo almacenado en el repositorio del proyecto.

```
from SPARQLWrapper import SPARQLWrapper, JSON, RDF, XML
from bd_vr import BDdatos
import codecs
import json
from datetime import date

db = BDdatos()
con = db.conectar()

# Variables to start the query process. To change de rootConcept.

rootConcept = ["http://dbpedia.org/resource/Category:Virtual_reality"] # set the root concept
e = "http://live.dbpedia.org/sparql"  # set the EndPoint: DBPedia Live


def getSubConcepts():
    """
        Get sub-soncepts of the root concept(s). Retrieve subconcepts located in three hops.
    """

    varP = ['c1', 'c2', 'c3', 'level']
    sparql = SPARQLWrapper(e)

    query = """select distinct ?level ?r ?c1 ?c2 ?c3
    {
        {VALUES (?r ?level) {(<%s> 1)}
            ?r ^skos:broader ?c1.}
        union
        {VALUES (?r ?level) {(<%s> 2)}
            ?r ^skos:broader ?c1. ?c1 ^skos:broader ?c2.}
        union
        {VALUES (?r ?level) {(<%s> 3)}
            ?r ^skos:broader ?c3. ?c3 ^skos:broader ?c2. ?c2 ^skos:broader ?c1.}
    }
    """
```

En este código se obtienen los subconceptos existentes que tiene el nodo raíz (https://dbpedia.org/page/Category:Virtual_reality) bajo la relación de skos:broader, en este caso se llegó hasta el tercer nivel de recorrido en DBPedia, obteniendo un total de 772 subconceptos. Para obtener estos subconceptos se utilizó una consulta SPARQL que recorre la red de conceptos cercanos al nodo raíz, bajo la relación de *skos:broader*. Además también se obtienen las url de resource de cada concepto encontrado a través de una consulta a la API de Wikipedia con los datos ya limpiados, obteniendo el link de DBPedia y de Wikipedia de cada uno de los conceptos y sus metadatos.

```
from SPARQLWrapper import SPARQLWrapper, JSON, RDF, XML
from datetime import date
from bd_vr import BDdatos

db = BDdatos()
con = db.conectar()

e = "http://live.dbpedia.org/sparql"
otable = 'DBSubconcepts_WikiPages'  # table to fill with wikipages data

cur = con.cursor()
# Create table:
cur.execute(u"""CREATE TABLE """ + otable + """ (`date` date, `concept` text, """ +
                  """ `dbrResource` text, `wikiPage` text, `wikiPageModified` date DEFAULT NULL, `max_outDegree` INT DEFAULT
                  """ ENGINE=InnoDB DEFAULT CHARSET=utf8;""")

db.saveDB(con)
db.closeDB(con)
```

Enlace a repositorio:

https://github.com/AndresRoman/proyecto_sbc