**Changes to Pipleline 1**

In first Assignment, I did little modification in belong to and has position jape rules. I included one more feature for usage of showing same as relationship for person.

**RDF Schema**

This schema contains totally five classes

1. **Position of Person -** This class represent the annotations related to Position of person annotation which are captured through the JAPE rules.
2. **University Name -** This class depicts the annotations of University name. this annotation are also captured in pipeline before for using.
3. **Person Name -** This class represent the name annotation captured during processing. It is the subclass of “**FOAF class**”.
4. **Organizational Unit –** This class represents the organizational unit of Universities like “Faculty of Fine and Arts”.

This schema contains three Object Properties

1. **Belong\_to –** This object property captures belong to relation find in documents. It has two properties which are Domain and Range. Domain for this property is Person Name. Range for this object property University Name.
2. **Has\_Pos –** This object property finds all the persons position which are happened in document. It has two properties which are Domain and Range. Domain for this property is Person Name. Range for this property is Person position.
3. **Org\_Unit –** This object property contains the name of the organizational units.

**RDF Export**

* **Person Name –** First I get the class from the schema file and this is class is already a subclass of FOAF class. For each and every detected name, I create a new instance of person class.
* **University Name -** First get this class from defined schema and for detected every make the new instance of university name class.
* **Position of person -** Get the class from the defined schema. After for detected every position make the instance of type position of person.
* **Organizational Unit –** Fetch this class from the schema, and make the instance of this class for every detected organizational unit entities.
* **Belong\_to –** this Object property has two properties. First I make the instance of this object property. After by fetching the detected annotation of belong to, I get the feature dom, ran out of it, and assigned to domain and range property.
* **Has\_Position -** This Object property has two other property which is domain and range. After getting the has position annotation from the documents, I get the feature dom, ran out of it, and assigned to domain and range property.
* **Org\_Unit –** This Object property, I get from the defined schema and for every annotations detected in document, I assign new instance of org\_unit type.

**Linking stategy**

* For University name, I get the annotation called University name and get the name feature out of it. I convert this name into well-defined DBPedia URI which can be recognizable. After, for linking the generated DBPedia URL with University name URI, I used owl same As property.
* For Person Name, I made new PR for extracting the same referred through coreference chain. And put that name into belong to annotation. In last processing step, I put Owl same As notation to second detected name which is referring to first name person.

**Triplestore implementation** – For storing triple, I am writing the generated triples into RDF file. In this case, whole pipeline is running for every document. Therefore, I check that model or RDF file is previously generated or not. If the RDF file is generated before, then I add the newly generated instance and properties to existing RDF file.

* **Total number of generated triples are approximately 9454**

**SPARQL End point implementation**

For setting up the SPARQL end point, I have used FUSEKI server. I load the RDF on Fuseki local server, and server start listening on 3030 ports. After going on local host, we can fire the SPARQL query to get desired results

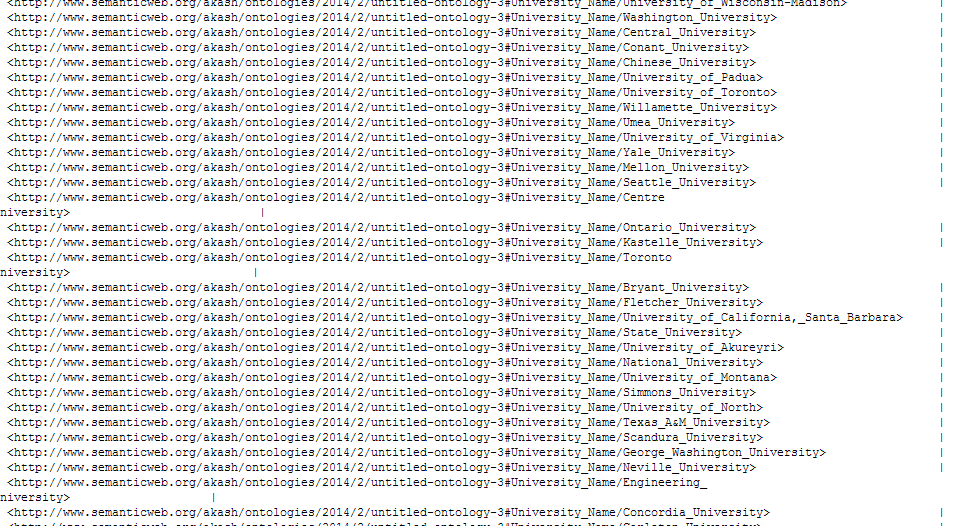
**SPARQL Queries**

1. **Find all Universities Name –**

SELECT ?s WHERE {

?s ?p <http://www.semanticweb.org/akash/ontologies/2014/2/untitled-ontology-3#University\_Name>

}



1. **Find the name of all university employee with their positions**

SELECT ?name ?pos WHERE {

?s ?p <http://www.semanticweb.org/akash/ontologies/2014/2/untitled-ontology-

3#Has\_Position> .

?s <http://www.w3.org/2000/01/rdf-schema#domain> ?name.

?s <http://www.w3.org/2000/01/rdf-schema#range> ?pos

}

****

1. **SPARQL query across two domains**

**SELECT ?obj WHERE {**

**{<http://www.semanticweb.org/akash/ontologies/2014/2/untitled-ontology-**

**3#University\_Name/University\_of\_Montana> ?predi ?obj**

**}**

**UNION{**

**<http://www.semanticweb.org/akash/ontologies/2014/2/untitled-ontology-**

**3#University\_Name/University\_of\_Montana>**

**<http://www.w3.org/2002/07/owl#sameAs> ?o**

**SERVICE <http://dbpedia.org/sparql>**

**{**

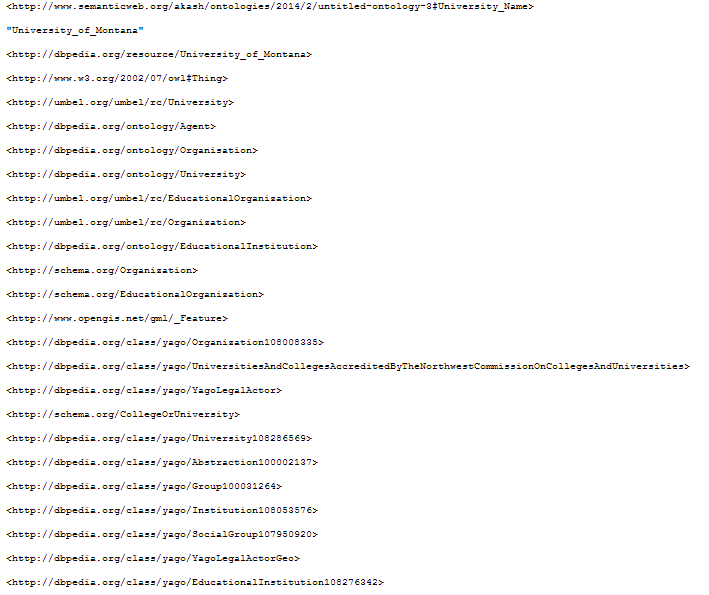
**?o ?predi ?obj**

**}**

**}**

**}**

**Screenshot of result of third query**

****

**Link Evaluation -**

1. Total number of university has been extracted is 240, and 181 out of 240 links have pages on DBPedia. So, coverage of links are 75 %.
2. I choose 25 links from different texts, 17 out of 25 is correctly linked to their true DBPedia page. Therefore, accuracy of links are 68%

**Improvement in linking strategy**

For better linking, I would like to get the locations from the surrounding contexts. And, if more than one location has been selected, I would like to assign rank according to place which is most likely to be the best candidate. After that annotate the name, it would be the better option for improvement.