

# CS5560 Knowledge Discovery Management

## Lab Assignment 4

**Lab Submission before November 28<sup>th</sup> 2018 (11:59 PM).**

Submit a short lab report including screenshots and data description to Turnitin . Post your GitHub URL with Source Code (No wiki) through Lab 4 form at <https://goo.gl/forms/MP2n7LQGmm2atsZo2>

### 1. Ontology Description

Trace the following Steps using the reference

#### 1. Determine the domain and scope of the ontology.

The domain of our ontology is Monarch Disease which is a semi-automatically constructed ontology that merges in multiple disease resources to yield a coherent merged ontology and its causes and treatments. Basically, for our Monarch Disease, there have two condition that can lead the disease, namely genetic causes and environmental causes. In addition, the ontology may contain some is that range of possible causes of Monarch Disease appears to be limited. For example, like gender.

#### 2. Consider reusing existing ontologies.

The reusing existing ontologies are some already existed ontologies that relevant to our ontology. Like "A Census of Disease Ontologies".

#### 3. Enumerate important terms.

For our ontology, like human, risk, cancer, diagnose, radiotherapy, patients, acromegaly, family, molecular, characteristics ...

#### 4. Define the classes & class hierarchy.

For our ontology, we set diseases, environment, gene, chemical ... as our classes. For the class hierarchy, we set some words that have some relatives with our classes.

For example:

- Disease

- human

- human disease causes and treatments.

#### 5. Define the properties of classes.

The properties of classes are property of sentences, like subject, object and verb.

For example:

The oncofoetal antigen 5T4 is a promising T cell target in the context of colorectal cancer.

**6. Define the facets of the slots.**

Facets are some words that seems like they belong to same mentioned under classes. For example, like our Monarch Disease which contain some diseases. They are mentioned to be as facets of slots and not act as subclasses.

## 7. Create instances.

Instances of ontology are different words that have same meaning. For example, cancer, tumor.

## Ontology Guide:

<http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html>

**2. Characterize the triplets based on the following Rules (minimum 5) Triplets:**

[illegible]

### a. Inverse Of

```
def inverseOf(inputf : RDD[String]): Unit =
{
    val trip1 = inputf.map(line => line.split(regex = ",").collect())
    val trip2 = trip1
    val pw = new PrintWriter(new File( pathname = "output/inverseOf.txt" ))

    trip1.foreach( t1 =>
    {
        trip2.foreach( t2 => {
            if (t1(0).equals(t2(2)) && t1(2).equals(t2(0)) && !t1(1).equals(t2(1)) )
                println(t1(0) + "," + t1(1) + "," + t1(2) + " inverse to " + t2(0) + "," + t2(1) + "," + t2(2))
            pw.write(t1(0) + "," + t1(1) + "," + t1(2) + " INVERSE TO " + t2(0) + "," + t2(1) + "," + t2(2) + "\n")
        })
    })
    pw.close()
}
```

The file is too large: 130.07 MB. Showing a read-only preview of the first 244 MB.

```

1 aim, assess, attitude INVERSE TO aim, assess, attitude
2 aim, assess, attitude INVERSE TO aim, assess, attitude
3 aim, assess, attitude INVERSE TO aim, assess, attitude
4 aim, assess, attitude INVERSE TO aim, assess, attitude
5 aim, assess, attitude INVERSE TO aim, assess, knowledge
6 aim, assess, attitude INVERSE TO aim, assess, knowledge
7 aim, assess, attitude INVERSE TO aim, assess, knowledge
8 aim, assess, attitude INVERSE TO aim, assess, knowledge
9 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
10 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
11 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
12 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
13 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
14 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudents
15 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudentsOfKarachi
16 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudentsOfKarachi
17 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudentsOfKarachi
18 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongCollegeStudentsOfKarachi
19 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
20 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
21 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
22 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
23 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
24 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudents
25 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudentsOfKarachi
26 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudentsOfKarachi
27 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudentsOfKarachi
28 aim, assess, attitude INVERSE TO aim, assess, knowledgeAmongFemaleCollegeStudentsOfKarachi
29 aim, assess, attitude INVERSE TO aim, assess, practice
30 aim, assess, attitude INVERSE TO aim, assess, practice
31 aim, assess, attitude INVERSE TO aim, assess, practice
32 aim, assess, attitude INVERSE TO aim, assess, practice
33 aim, assess, attitude INVERSE TO aim, assess, practiceOfBse
34 aim, assess, attitude INVERSE TO aim, assess, practiceOfBse
35 aim, assess, attitude INVERSE TO aim, assess, practiceOfBse
36 aim, assess, attitude INVERSE TO aim, assess, practiceOfBse
37 aim, assess, attitude INVERSE TO cancer, is, leading
38 aim, assess, attitude INVERSE TO cancer, is, leading
39 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
40 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
41 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
42 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
43 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
44 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool
45 aim, assess, attitude INVERSE TO conditions, make, diseaseSelf-examinationBestTool

```

## b. Symmetric Property

```

def symmetry(inputf : RDD[String]): Unit =
{
  val trip1 = inputf.map(line => line.split(regex = ",")).collect()
  val trip2 = trip1
  val pw = new PrintWriter(new File( pathname = "output/symmetry.txt" ))

  trip1.foreach( t1 =>
  {
    trip2.foreach( t2 => {
      if (t1(0).equals(t2(2)) && t1(2).equals(t2(0)) && t1(1).equals(t2(1)) )
        //println(t1(0) + " " + t1(1) + " " + t1(2) + " inverse to " + t2(0) + " " + t2(1) + " " + t2(2))
        pw.println(t1(0) + " " + t1(1) + " " + t1(2) + " SYMMETRIC TO " + t2(0) + " " + t2(1) + " " + t2(2) + "\n")
    })
  })
  pw.close()
}

```

Triplets.txt × transitive.txt × triplets × Triplets.txt × inverseOf.txt × symmetry.txt × irreflexive.txt

```

1 disease, is, Creating SYMMETRIC TO Creating, is, disease

```

## c. Transitive Property

```
def transitive(inputf : RDD[String]): Unit =
{
    val tripl = inputf.map(line => line.split(regex = ",")).collect()
    val trip2 = tripl
    val pw = new PrintWriter(new File(pathname = "output/transitive.txt"))

    tripl.foreach( t1 =>
    {
        trip2.foreach( t2 => {
            if (t1(2).equals(t2(0)) && t1(1).equals(t2(1)) && !t1(2).equals(t2(2))) {
                //println(t1(0) + "," + t1(1) + "," + t1(2) + " inverse to " + t2(0) + "," + t2(1) + "," + t2(2))
                println(t1(0) + "," + t1(1) + "," + t1(2) )
                println(t2(0) + "," + t2(1) + "," + t2(2) )
                println("transitive :")
                println(t1(0) + "," + t1(1) + "," + t2(2) )
                //println("\n")
            }
        })
    })
    pw.close()
}
```

Output/Triples.txt × transitive.txt × triplets × output\_of/Triples.txt × inverseOf.txt × symmetry.txt × irreflexive.txt ×

disease, is, cancer and cancer, is, causeOfDeath "TRANSITIVE TO" disease, is, causeOfDea

#### d. Property Chain Axiom

```
def propertyAxiom(inputf : RDD[String]): Unit =
{
    val tripl = inputf.map(line => line.split(regex = ",")).collect()
    val trip2 = tripl
    val trip3 = tripl
    val pw = new PrintWriter(new File(pathname = "output/propertyaxiom.txt"))

    tripl.foreach( t1 =>
    {
        trip2.foreach( t2 => {
            if (t1(2).equals(t2(0)) && t1(1).equals(t2(1)) && !t1(2).equals(t2(2))) {
                trip3.foreach(t3 => {
                    if (t1(0).equals(t3(0)) && t2(2).equals(t3(2)) && !t1(1).equals(t3(1))) {
                        println(t1(0) + "," + t1(1) + "," + t1(2) )
                        println(t2(0) + "," + t2(1) + "," + t2(2) )
                        println("property axiom :")
                        println(t3(0) + "," + t3(1) + "," + t3(2) )
                        //pw.write("\n")
                    }
                })
            }
        })
    })
    pw.close()
}
```

Output/Triples.txt × transitive.txt × triplets × output\_of/Triples.txt × inverseOf.txt × symmetry.txt × irreflexive.txt × propertyaxiom.txt ×

disease, is, cancer and cancer, is, causeOfDeath => disease, is, causeOfDeath

#### e. Irreflexive Property

```

def irreflexive(inputf : RDD[String]): Unit =
{
    val trip1 = inputf.map(line => line.split( regex = "," ),).collect()
    val trip2 = trip1
    val pw = new PrintWriter(new File( pathname = "output/irreflexive.txt" ))

    trip1.foreach( t1 =>
    {
        trip2.foreach( t2 => {
            if (t2(2).equals(t2(0)) && t1(0).equals(t2(0)) && !t1(2).equals(t2(2)) && t1(1).equals(t2(1)) ) {
                println(t1(0) + "," + t1(1) + "," + t1(2) )
                println(t2(0) + "," + t2(1) + "," + t2(2) )
                //pw.write("\n")
            }
        })
    })

    pw.close()
}

```

### 3. Record unique features noted from your ontology

1. Entity recognition creation
2. Subclasses creation
3. Synonyms creation
4. Top words creation