Project – Syntactic and Semantic Matcher

Jyoti Yadav jyoti@kth.se
Larry Poon, larryp@kth.se

Project Setup: I created maven project structure from the existing provided structure. All jar files are added as maven dependency in pom.xml file so that it is easy to build.

Task 1: Syntactic matching: In order to parse wsdl files using JAXB I created file xmlsoap.xsd which contains schema for wsdl file. It was downloaded from http://schemas.xmlsoap.org/wsdl/ and saved to file xmlsoap.xsd. Then I used maven jaxb plugin to generate java classes from xmlsoap.xsd file. I also used the same plugin to generate the java classes for output.xsd file so that I could use jaxb to write the matcher output. All the generated classes can be found under **webserviceproject/target/generated-sources/jaxb/com/jyoti/webservice** path.

Now I used jaxb to load to load all wsdl definitions. For each wsdl file I created wsdl definitions and from these definitions I extracted operation list, input and output messages for each operation. This was the most difficult part and I took some online help (mentioned in references) to do this.

Now for each wsdl file I compared the input operations and their parameters with the output operations and parameters of all other wsdl files to find the score and then save the result to output file using jaxb. It used EditDistance to calculate the score between input and output elements.

Task 2: Task 2 just creates the average operation score of all elements matching score. It also creates service score as average of all operation scores. This was easy as most of the work was done in previous task and it included only sum and average calculation.

Task 3: Semantic matching. The main difference between this from task 2 is that it uses different algorithm to calculate the matching score. Also we parse the ontology (i.e. modelReference) attribute from wsdl definitions. Semantic matcher overrides the getMatchingScore method and uses owl classes from provided OntologyManager to find the degree of matching. It uses different methods to find different degrees of matching. For e.g exact if two ontology attributes are same, isSubclassOf if reasoner finds that the two ontology attribute's owl classes are subclasses etc.

Instructions: All the source code is in maven project format. You will need to change path to owl files and point them to correct location in Test_MyOntManager.java and MatcherLauncher.java. In order to run it we first need to import it in Intellij and run **mvn clean install**. Maven will first generate all jaxb java classes and will then compile source and build jar. The main class, which needs to be run is com.jyoti.ws.project. MatcherLauncher. This runs both Syntactic and Semantic Matcher and creates output files in directory **webserviceproject/src/main/output/**

We took reference for the jaxb parser and how to flatten complex type from the link given below.

References:

https://www.tutorialspoint.com/wsdl/wsdl_definition.htm
https://www.mkyong.com/java/jaxb-hello-world-example/
https://github.com/mattec92/KTH/tree/master/ID2208%20-%20Programming%20Web-Services/Project