MASWS Assignment 1 Part 2

Dilja Rudolfsdottir s0966087 & Steven Eardley s0934142

1. Steps Taken in Converting to RDF

The purpose of this assignment was to take the UK Anonymised MOT tests and results data we outlined in Part 1 of this assignment and convert it into RDF format, or more specifically Turtle. We had intended to use `flat2n3.pl` - a Perl script from MIT's 'Simile' project[i] to convert our data but instead we ended up writing our own script to do the conversion. There were a few reasons behind this; first of all we would have had to spend a considerable amount of time figuring out all of the components in the Perl script since they were not immediately obvious, secondly that script did not output the data in Turtle format which means we would have had to do a second conversion. Instead we decided that the format of our data was actually rather simple to convert and a quicker way would perhaps just be to write our own program for converting the data. This allowed us to tailor the structure around our data, increasing efficiency. The code for the program is detailed in Section 7 at the end if this report, so the reader can get an idea of how it works.

Before any conversion could take place, the quantity of data was considered. The 730 MB gzip compressed file extracts to 2.5 GB of text entries, corresponding to 29,179,062 individual MOT tests. These entries have the following format:

```
145668806|28962612|2011-02-01|4|N|P|53989|PE|FORD|ESCORT FINESSE 16V|RED|P|1597|1998-12-02
```

We decided to tackle a month's worh of data – we chose February, and extracted these entries using the following command on Linux:

```
grep “2011-02-” test_result_2011.txt
```

This takes all entries including the partial date for February, a total of 2,882,093. Although there are two date fields in the MOT data, it is safe to use this method because they do not overlap – tested vehicles which were first registered (the last field) in 2011 will not have been tested in the same year – vehicles are only tested one or three years after manufacture.

To build a program to convert the data we first had to consider the desired output and study Turtle examples to try to get an idea of what our data would look like after the conversion had been completed. An e-commerce example[ii] from GoodRelations using the vso (Vehicle Sales Ontology) was a useful starting point.

The input our conversion program takes is therefore of the format above with newlines in between rows and the "pipe" character ("|") representing the different columns. The meanings of the above entries are defined in the user guide released with the data. Table 1 below shows how the data categories above are mapped to our defined predicates (discussed later), in correct order from left-to-right.

Category (from guide)	Type	RDF Prediacte	RDF Type
Test ID	integer (unique)	mot:testid	xsd:integer
Vehicle ID	integer	mot:vehicleid	xsd:integer
Test Date	date	mot:testdate	xsd:date
Test Class ID	string{“1”, “2”, “3”, “4A”, “5”, “5A”, “7”}	mot:testclassid	xsd:string
Test Type	string{“N”, “F”, “PM”, “PR”, “RF”, “PL”}	mot:testtype	xsd:string
Test Result	string{“P”, “F”, “PRS”, “ABA”, “ABR”, “R”}	mot:testresult	xsd:string
Test Mileage	integer	vso:mileageFromOdometer	gr:QuantitativeValueFloat
Post Code	string	mot:testlocation	dbpedia URI
Make	string	gr:hasManufacturer	dbpedia URI
Model	string	mot:vehiclemodel	xsd:string
Fuel Type	string	vso:fuelType	dbpedia URI
Cylinder Capacity	integer	vso:engineDisplacement	gr:QuantitativeValueFloat
First Use Date	string	vso:firstRegistration	xsd:date

Table 1: Summary of CVS equivalent to RDF conversion types

To start with, our conversion program takes as an input a text file of the above described format. The output is then a .ttl file in Turtle format with the completed conversion. First of all a number of prefixes are defined to make for better readability of the output file, as well as saving time when writing up the URIs. For every line in the file it splits the line at every "pipe" character and creates an array called 'objects' of length 14 for each of the 14 objects in each line. The program itself can not recognise which item in the array corresponds to which of the predicates described above so we define them for objects[0] to objects[13] like the following example:

```
out.write(String.format("\tmot:testlocation dbpedia:%s_postcode_area .\n", objects[7]));
```

The above line of code writes out a line in the output Turtle file which represents the two last components of a triple, the predicate and the object. The full triple is not in the line above since it is in an abbreviated form so the first component, the subject, is represented only in the first triple of this subject. The semi-colon at the end represents that the subject has more possible predicates and objects which follow after this one. In the example above the predicate is the Post Code where the URI is dbpdia:Postal_Code and the object is the URI dbpedia:X_postcode_area where X stands for the given postcode in each line (EH, PE, SO etc.). The program runs through every line in the input file in this way and outputs the corresponding triples, or abbreviated triples, in the correct Turtle format . The program also specifies and prints out the relations between each of the resources and their data types.

Some issues that arose while doing this conversion were for example deciding on which classes to divide the data into and how to choose a subject name to uniquely identify each instance. Our solution was to consider each entry as two distinct entities: the test and the vehicle. The test is instantiated with a 'blank node' labelled with the Test ID – uniqueness of the label is guaranteed because this is the primary key in the database:

```
_:72538084 a mot:Test ;
```

The same method also gives us instantiations of individual vehicles, using the `mot:Vehicle` class:

```
_:15988135 a mot:Vehicle ;
```

These are related to each test by the `mot:Test` class' `mot:testvehicle` predicate (subject added here for clarity, but is not written by our program):

```
_:72538084 mot:testvehicle _:15988135 ;
```

i.e. the test `_:72538084` was on the vehicle `_:15988135`.

To test the validity of our RDF output we used the tool available on rdfabout.com[iii] which accepts a snippet of RDF/XML or N3/Turtle and checks the syntax. The sample of RDF output from our program (as seen in Section 5) passes the tests. The tool also produces an output of 'the underlying triples' defined by the RDF file, which shows the data has been organised as was intended. This is presented in Section 8 below.

2. 3rd Party Schemas Used

There were two main 3rd party schemas that we chose to use: Vehicles Sales Ontology and Good Relations. Vehicle Sales Ontology, abbreviated as 'vso' in the Turtle conversion, was chosen since we needed to find a vocabulary that could describe car properties, such as the predicates Fuel Type, Make and First Use Date. While VSO used different terms for many of these predicates, such as *firstRegistration* for First Use Date and *engineDisplacement* for Cylinder Capacity those still represent the "thing in the world" that we are referring to.

The Good Relations (abbreviated as 'gr' in the conversion) vocabulary is designed to cover e-commerce, and the VSO vocabulary falls under GR. While the VSO vocabulary is very specific for vehicles the GR vocabulary can cover a much broader area of sales, production, services and many other things. In our conversion we only used the GR vocabulary for one predicate, the Make of a car, although in the GR case the predicate is called hasManufacturer, but this represents the same thing in the real world. The GR ontology also supplies various data types used in VSO, so these can be seen in our output.

While VSO and Good Relations were very useful for a lot of the car properties we required there were more general objects that we used DBpedia for. Those included Postcode, the different Fuel Types (Gasoline, Diesel, Electricity etc.) and the different car manufacturers (Peugeot, Ford, Honda etc.). Relating the manufacturers to DBpedia entries required making all but the first letter of each lower case, adding it to a `dbpedia:` prefix with the expectation that it will resolve. For example, the entry example above with Make "FORD" will be written to file as `dbpedia:Ford`, expands to http://dbpedia.org/resource/Ford and resolves to http://dbpedia.org/page/Ford_Motor_Company in a browser. All examples we tested produced a valid resource, so we expect this method is adequate.

3. New Vocabulary Created

While a lot of the vocabulary that were were interested in using was already available there were still some more task-specific things that we needed to create our own vocabulary for. Those were the predicates that were unique for the MOT Test, such as Test ID, Test Date, Test Type, Test Result and Vehicle ID. Since the MOT tests had not been converted into RDF before it is no surprise that those predicates were not already specified as URIs and while there are URIs for predicates such as "ID" those do not represent to the same predicates in the real world and therefore a new vocabulary had to be created. The full definition is included in Section 6 of this document.

For both tests and vehicles it was necessary to define a class – the former because it was novel and the latter because there was additional information required over that which could be expressed using the VSO, namely the Vehicle ID field and a Model which accepts a string value. In order to utilise the VSO the `mot:Vehicle` class was defined as a subclass of `vso:MotorizedRoadVehicle` – the class which contains cars, motorbikes, buses, vans and so on. `mot:Vehicle` is therefore defined as a 'vehicle which has been tested by MOT'.

When defining the predicates we ensured that their names were as close as possible to those defined by the user guide supplied with the raw data. This makes it straightforward to understand what each is for. Table 2 summarises the meanings of predicates defined for the MOT Schema.

Class	Property	Description
`mot:Test`	-	A class to define the result of an MOT test
	`mot:testid`	The ID of the test
	`mot:testvehicle`	The `mot:Vehicle` the test was on
	`mot:testdate`	The date of the test
	`mot:testclassid`	The class of vehicle in the test
	`mot:testtype`	The type of test carried out
	`mot:testresult`	The outcome of the test
	`mot:testlocation`	The area the test was in
`mot:Vehicle`	-	A class to define a Vehicle tested by MOT
	`mot:vehicleid`	The ID of the vehicle
	`mot:vehiclemodel`	The vehicle's model

Table 2: The meaning of vocabulary defined for MOT tests.

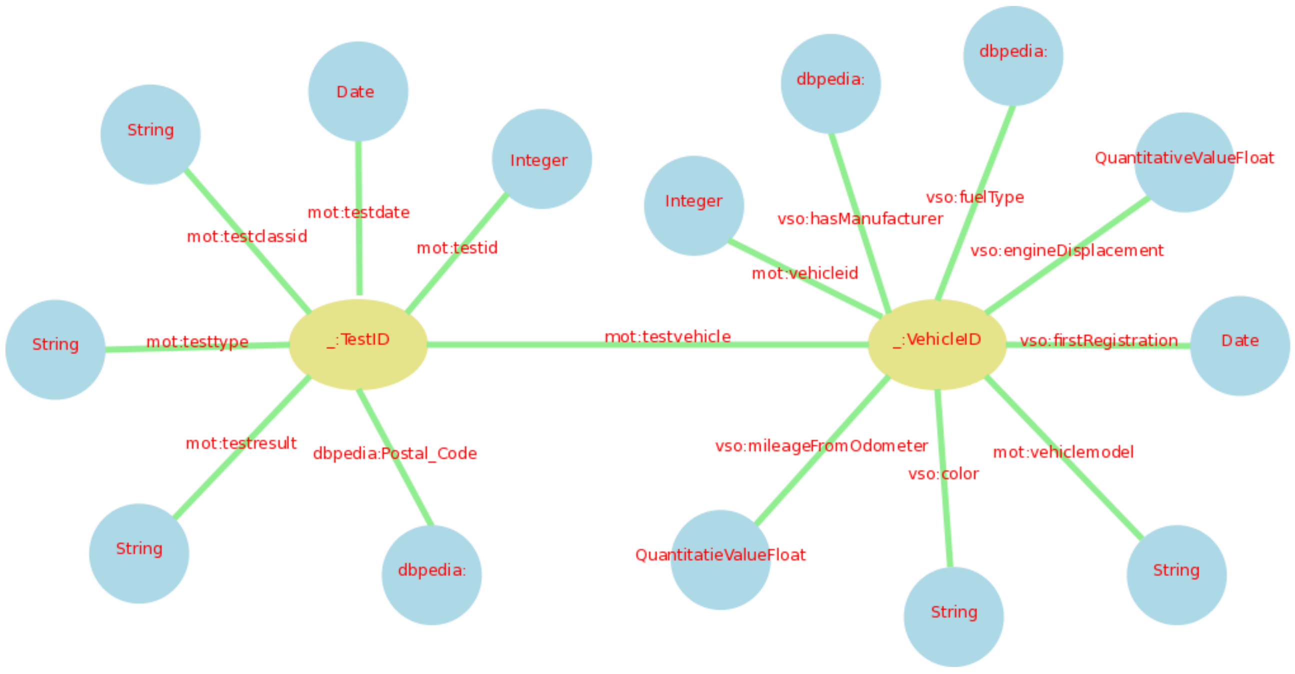
Table 2 shows the entire extent of properties of the mot:Test class, but not for mot:Vehicle – there are numerous in use which are inherited from `vso:MotorizedRoadVehicle`. Table 1 above shows how the remainder of these fitted in with the MOT data.

We first considered using the `vso:Automobile` class, but had to go a class up the hierarchy (to `vso:MotorizedRoadVehicle`) since can't guarantee all are cars – the data additionally contains vans and motorbikes. Even when considering the MOT Test Class ID data (motorbikes are tested under class 1, cars under 4) it is not possible to effectively classify all vehicles into the subclasses of `vso:MotorizedRoadVehicle`: there is no way of discerning the other subclasses at that level – `vso:BusOrCoach`, `vso:Truck`, `vso:Van` - these may all be tested under MOT class 4.

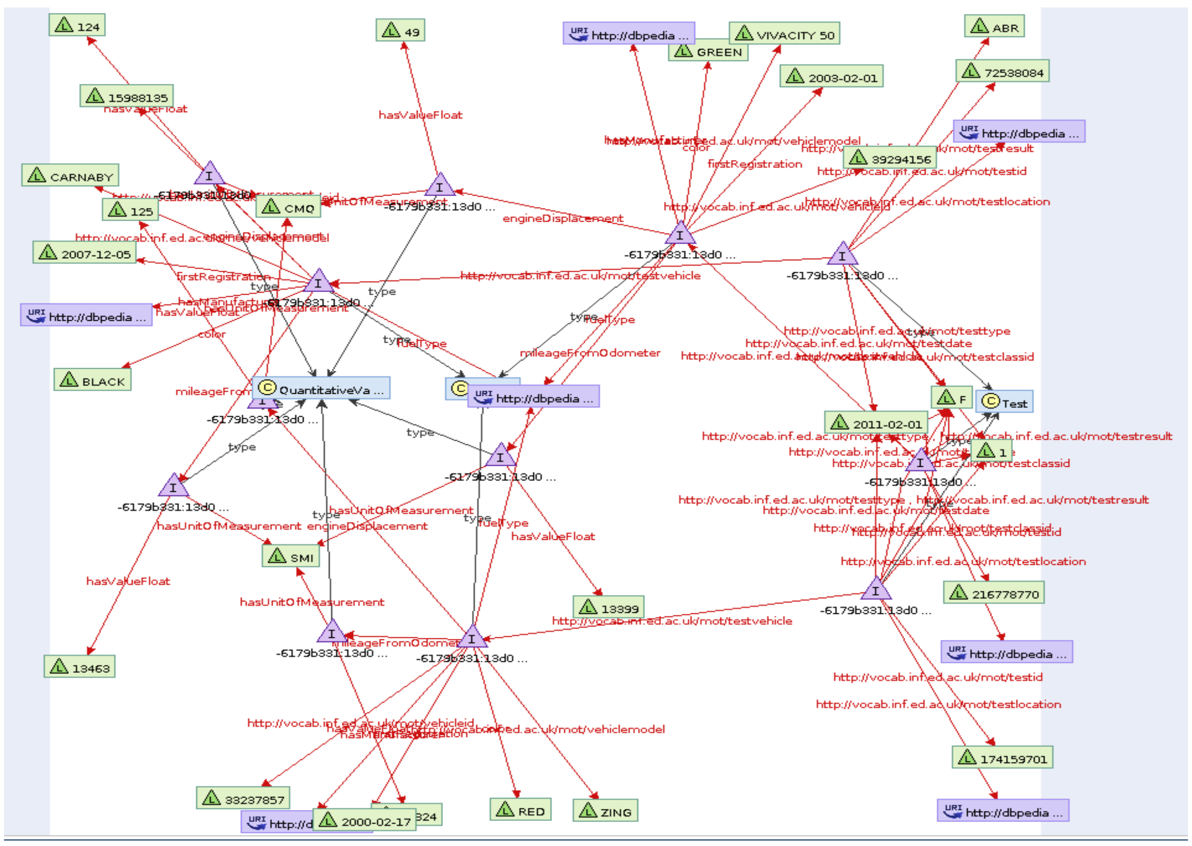
4. Visualisation of RDF Schema

When it came to visualising the RDF Schema we took the same approach as when converting the data. Using available visualisation toolkits seemed to be over-complicating the matter when creating our own visualisation seemed relatively straight forward. Attempts to use Cytoscape[iv], Jena and various on-line tools resulted in disaster. We therefore decided to create our own visualisation graph on the DIA Diagram Editor, shown below. In this graph a yellow eclipse stands for a Subject, a green line stands for a Predicate and a light blue circle stands for an Object. Subject and a Predicate are labelled with their correct names while the Objects are labelled with the data types those objects can take.

As we can see from the graph the dataset has two classes, one which is uniquely identified by the subject `mot:testid` and another which is uniquely identified by the subject `mot:vehicleid`. The `mot:Test` class has all of the properties from the actual test, many of which we had to create our own URIs for, while the `mot:Vehicle` class has has all of the properties of the vehicle. The `mot:vehicleid` is yellow because it is the Subject but it is also an Object in one relation, the one that connect `mot:testid` to `mot:vehicleid` via the `mot:testvehicle` predicate.



One tool that almost worked was RDF-Gravity, automatically producing the following graph for our sample in the following section (once converted to RDF/XML):



5. RDF Output

The following is a sample from our results when converting February's MOT test data. It details the prefixes and three tests on three different vehicles. The `mot` prefix is a placeholder, since we haven't made the schema we defined publicly available. The entire file for February's data contains 63,406,052 lines of text and is 2.3 GB in size uncompressed. A downloadable (compressed to 137.34 MB) version can be found at: https://www.dropbox.com/s/x9dds0630bv6vax/RDF-Output.ttl.gz

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix vso: <http://www.heppnetz.de/ontologies/vso/ns#> .
@prefix gr:  <http://purl.org/goodrelations/v1#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix mot: <http://vocab.inf.ed.ac.uk/mot/> .

_:72538084 a mot:Test ;
        mot:testid "72538084"^^xsd:integer ;
        mot:testvehicle _:15988135 ;
        mot:testdate "2011-02-01"^^xsd:date ;
        mot:testclassid "1"^^xsd:string ;
        mot:testtype "F"^^xsd:string ;
        mot:testresult "ABR"^^xsd:string ;
        mot:testlocation dbpedia:KT_postcode_area .
_:15988135 a mot:Vehicle ;
        vso:mileageFromOdometer [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "13463"^^xsd:float ;
                gr:hasUnitOfMeasurement "SMI"^^xsd:string] ;
        mot:vehicleid "15988135"^^xsd:integer ;
        gr:hasManufacturer dbpedia:Piaggio ;
        mot:vehiclemodel "CARNABY"^^xsd:string ;
        vso:color "BLACK" ;
        vso:fuelType dbpedia:Gasoline ;
        vso:engineDisplacement [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "124"^^xsd:float ;
                gr:hasUnitOfMeasurement "CMQ"^^xsd:string ] ;
        vso:firstRegistration "2007-12-05"^^xsd:date .

_:174159701 a mot:Test ;
        mot:testid "174159701"^^xsd:integer ;
        mot:testvehicle _:33237857 ;
        mot:testdate "2011-02-01"^^xsd:date ;
        mot:testclassid "1"^^xsd:string ;
        mot:testtype "F"^^xsd:string ;
        mot:testresult "F"^^xsd:string ;
        mot:testlocation dbpedia:WN_postcode_area .
_:33237857 a mot:Vehicle ;
        vso:mileageFromOdometer [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "30824"^^xsd:float ;
                gr:hasUnitOfMeasurement "SMI"^^xsd:string] ;
        mot:vehicleid "33237857"^^xsd:integer ;
        gr:hasManufacturer dbpedia:Kymco ;
        mot:vehiclemodel "ZING"^^xsd:string ;
        vso:color "RED" ;
        vso:fuelType dbpedia:Gasoline ;
        vso:engineDisplacement [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "125"^^xsd:float ;
                gr:hasUnitOfMeasurement "CMQ"^^xsd:string ] ;
        vso:firstRegistration "2000-02-17"^^xsd:date .

_:216778770 a mot:Test ;
        mot:testid "216778770"^^xsd:integer ;
        mot:testvehicle _:39294156 ;
        mot:testdate "2011-02-01"^^xsd:date ;
        mot:testclassid "1"^^xsd:string ;
        mot:testtype "F"^^xsd:string ;
        mot:testresult "F"^^xsd:string ;
        mot:testlocation dbpedia:CB_postcode_area .
_:39294156 a mot:Vehicle ;
        vso:mileageFromOdometer [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "13399"^^xsd:float ;
                gr:hasUnitOfMeasurement "SMI"^^xsd:string] ;
        mot:vehicleid "39294156"^^xsd:integer ;
        gr:hasManufacturer dbpedia:Peugeot ;
        mot:vehiclemodel "VIVACITY 50"^^xsd:string ;
        vso:color "GREEN" ;
        vso:fuelType dbpedia:Gasoline ;
        vso:engineDisplacement [ a gr:QuantitativeValueFloat ;
                gr:hasValueFloat "49"^^xsd:float ;
                gr:hasUnitOfMeasurement "CMQ"^^xsd:string ] ;
        vso:firstRegistration "2003-02-01"^^xsd:date .
```

6. MOT Schema Specification

This file also passes validation using the same tool as tested the RDF output.

```
# MASWS Assignment 2
# Diljá Rudolfsdóttir & Steven Eardley 2013

@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix vso:    <http://purl.org/vso/ns#> .
@prefix sch:    <http://schema.org/> .

# Placeholder for our MOT vocab
@prefix mot:    <http://vocab.inf.ed.ac.uk/mot/> .

# Define a class for an MOT Test
mot:Test    rdf:type    rdfs:Class ;
    rdfs:label "MOT Test" ;
    rdfs:comment """Defines an MOT Test result.""" .

mot:testid  rdf:type    rdf:Property .
mot:testid  rdfs:range  xsd:integer .
mot:testid  rdfs:domain mot:Test .

mot:testvehicle  rdf:type    rdf:Property .
mot:testvehicle  rdfs:range  mot:Vehicle .
mot:testvehicle  rdfs:domain mot:Test .

mot:testdate    rdf:type    rdf:Property .
mot:testdate    rdfs:range  xsd:date .
mot:testdate    rdfs:domain mot:Test .

mot:testclassid    rdf:type    rdf:Property .
mot:testclassid    rdfs:range  xsd:string .
mot:testclassid    rdfs:domain mot:Test .

mot:testtype    rdf:type    rdf:Property .
mot:testtype    rdfs:range  xsd:string .
mot:testtype    rdfs:domain mot:Test .

mot:testresult    rdf:type    rdf:Property .
mot:testresult    rdfs:range  xsd:string .
mot:testresult    rdfs:domain mot:Test .

mot:testlocation    rdf:type    rdf:Property .
mot:testlocation    rdfs:range  sch:AdministrativeArea .
mot:testlocation    rdfs:domain mot:Test .


# Define a class for a vehicle
mot:Vehicle rdf:type    rdfs:Class ;
    rdfs:subClassOf vso:MotorizedRoadVehicle ;
    rdfs:label  "Vehicle tested by MOT" ;
    rdfs:comment    """Defines a vehicle which has been tested in an MOT Test -
a sublass of the vso: MotorizedRoadVehicle""" .

mot:vehicleid  rdf:type    rdf:Property .
mot:vehicleid  rdfs:range  xsd:integer .
mot:vehicleid  rdfs:domain mot:Vehicle .
```

```
mot:vehiclemodel    rdf:type    rdf:Property .
mot:vehiclemodel    rdfs:range  xsd:string .
mot:vehiclemodel    rdfs:domain mot:Vehicle .
```

7. Conversion Code

```
/* MASWS Assignment 2

Diljá Rudolfsdóttir & Steven Eardley 2013
*/

import java.io.*;
import java.util.*;

public class rdfConverter{

public static void main(String args[]) {
    try{

        FileInputStream fstream = new FileInputStream(args[0]);

        BufferedReader br = new BufferedReader(new InputStreamReader(fstream));
        String strLine;

        FileWriter fstream3 = new FileWriter("RDF-Output.ttl", false);
        BufferedWriter out = new BufferedWriter(fstream3);

        out.write("@prefix dbpedia: <http://dbpedia.org/resource/> .\n");
        out.write("@prefix vso:
<http://www.heppnetz.de/ontologies/vso/ns#> .\n");
        out.write("@prefix gr:  <http://purl.org/goodrelations/v1#> .\n");
        out.write("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n");
        out.write("@prefix mot: <http://vocab.inf.ed.ac.uk/mot/> .\n\n");

        while (((strLine = br.readLine()) != null)) {
            String [] objects = strLine.split("\\|");

            // Create a class for the Test
            out.write(String.format("_:%s a mot:Test ;\n", objects[0]));

            // Add test properties
            out.write(String.format("\tmot:testid \"%s\"^^xsd:integer ;\n",
objects[0]));

            out.write(String.format("\tmot:testvehicle _:%s ;\n", objects[1]));

            out.write(String.format("\tmot:testdate \"%s\"^^xsd:date ;\n",
objects[2]));

            out.write(String.format("\tmot:testclassid \"%s\"^^xsd:string ;\n",
objects[3]));

            out.write(String.format("\tmot:testtype \"%s\"^^xsd:string ;\n",
objects[4]));

            out.write(String.format("\tmot:testresult \"%s\"^^xsd:string ;\n",
objects[5]));

            out.write(String.format("\tmot:testlocation dbpedia:%s_postcode_area
.\n", objects[7]));
```

```
            // Create a class for the vehicle
            out.write(String.format("_:%s a mot:Vehicle ;\n", objects[1]));

            out.write(String.format("\tvso:mileageFromOdometer [ a
gr:QuantitativeValueFloat ;\n\t\tgr:hasValueFloat
\"%s\"^^xsd:float ;\n\t\tgr:hasUnitOfMeasurement \"SMI\"^^xsd:string] ;\n",
objects[6]));

            out.write(String.format("\tmot:vehicleid \"%s\"^^xsd:integer ;\n",
objects[1]));

            out.write(String.format("\tgr:hasManufacturer dbpedia:%s ;\n",
(objects[8].charAt(0)) + objects[8].substring(1).toLowerCase()));

            out.write(String.format("\tmot:vehiclemodel \"%s\"^^xsd:string ;\n",
objects[9]));

            out.write(String.format("\tvso:color \"%s\" ;\n", objects[10]));

            String fuelType;
            char sw = objects[11].charAt(0);
            switch (sw) {
                case 'P':  fuelType = "dbpedia:Gasoline";
                        break;
                case 'D':  fuelType = "dbpedia:Diesel_fuel";
                        break;
                case 'E':  fuelType = "dbpedia:Electricity";
                        break;
                case 'S':  fuelType = "dbpedia:Steam";
                        break;
                case 'N':  fuelType = "dbpedia:Liquefied_natural_gas";
                        break;
                case 'F':  fuelType = "dbpedia:Fuel_cell";
                        break;
                case 'C':  fuelType = "dbpedia:Compressed_natural_gas";
                        break;
                case 'L':  fuelType = "dbpedia:Liquefied_petroleum_gas";
                        break;
                case 'O':  fuelType = "Other";
                        break;
                default: fuelType = "None Found";
                        break;
            }
            out.write(String.format("\tvso:fuelType %s ;\n", fuelType));

            out.write(String.format("\tvso:engineDisplacement [ a
gr:QuantitativeValueFloat ;\n\t\tgr:hasValueFloat
\"%s\"^^xsd:float ;\n\t\tgr:hasUnitOfMeasurement \"CMQ\"^^xsd:string ] ;\n",
objects[12]));

            out.write(String.format("\tvso:firstRegistration \"%s\"^^xsd:date .\
n\n", objects[13]));
        }
        System.out.println("\nSuccess! Written to RDF-Output.ttl\n");
        out.close();
        fstream.close();
    }

    catch (Exception e) {// Catch exception if any
    System.err.println("Error: " + e.getMessage());
    }
    }
}
```

8. The Underlying Triples (using rdfabout.com validator)

```
_:72538084 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Test>.

_:72538084 <http://vocab.inf.ed.ac.uk/mot/testid>
"72538084"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testvehicle> _:15988135.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testdate> "2011-02-
01"^^<http://www.w3.org/2001/XMLSchema#date>.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testclassid>
"1"^^<http://www.w3.org/2001/XMLSchema#string>.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testtype>
"F"^^<http://www.w3.org/2001/XMLSchema#string>.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testresult>
"ABR"^^<http://www.w3.org/2001/XMLSchema#string>.
_:72538084 <http://vocab.inf.ed.ac.uk/mot/testlocation>
<http://dbpedia.org/resource/KT_postcode_area>.
_:15988135 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Vehicle>.
_:bnode541280000 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode541280000 <http://purl.org/goodrelations/v1#hasValueFloat>
"13463"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode541280000 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"SMI"^^<http://www.w3.org/2001/XMLSchema#string>.
_:15988135 <http://www.heppnetz.de/ontologies/vso/ns#mileageFromOdometer>
_:bnode541280000.
_:15988135 <http://vocab.inf.ed.ac.uk/mot/vehicleid>
"15988135"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:15988135 <http://purl.org/goodrelations/v1#hasManufacturer>
<http://dbpedia.org/resource/Piaggio>.
_:15988135 <http://vocab.inf.ed.ac.uk/mot/vehiclemodel>
"CARNABY"^^<http://www.w3.org/2001/XMLSchema#string>.
_:15988135 <http://www.heppnetz.de/ontologies/vso/ns#color> "BLACK".
_:15988135 <http://www.heppnetz.de/ontologies/vso/ns#fuelType>
<http://dbpedia.org/resource/Gasoline>.
_:bnode1060000064 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode1060000064 <http://purl.org/goodrelations/v1#hasValueFloat>
"124"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode1060000064 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"CMQ"^^<http://www.w3.org/2001/XMLSchema#string>.
_:15988135 <http://www.heppnetz.de/ontologies/vso/ns#engineDisplacement>
_:bnode1060000064.
_:15988135 <http://www.heppnetz.de/ontologies/vso/ns#firstRegistration> "2007-
12-05"^^<http://www.w3.org/2001/XMLSchema#date>.
_:174159701 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Test>.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testid>
"174159701"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testvehicle> _:33237857.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testdate> "2011-02-
01"^^<http://www.w3.org/2001/XMLSchema#date>.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testclassid>
"1"^^<http://www.w3.org/2001/XMLSchema#string>.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testtype>
"F"^^<http://www.w3.org/2001/XMLSchema#string>.
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testresult>
"F"^^<http://www.w3.org/2001/XMLSchema#string>.
```

```
_:174159701 <http://vocab.inf.ed.ac.uk/mot/testlocation>
<http://dbpedia.org/resource/WN_postcode_area>.
_:33237857 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Vehicle>.
_:bnode641366912 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode641366912 <http://purl.org/goodrelations/v1#hasValueFloat>
"30824"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode641366912 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"SMI"^^<http://www.w3.org/2001/XMLSchema#string>.
_:33237857 <http://www.heppnetz.de/ontologies/vso/ns#mileageFromOdometer>
_:bnode641366912.
_:33237857 <http://vocab.inf.ed.ac.uk/mot/vehicleid>
"33237857"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:33237857 <http://purl.org/goodrelations/v1#hasManufacturer>
<http://dbpedia.org/resource/Kymco>.
_:33237857 <http://vocab.inf.ed.ac.uk/mot/vehiclemodel>
"ZING"^^<http://www.w3.org/2001/XMLSchema#string>.
_:33237857 <http://www.heppnetz.de/ontologies/vso/ns#color> "RED".
_:33237857 <http://www.heppnetz.de/ontologies/vso/ns#fuelType>
<http://dbpedia.org/resource/Gasoline>.
_:bnode122646848 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode122646848 <http://purl.org/goodrelations/v1#hasValueFloat>
"125"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode122646848 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"CMQ"^^<http://www.w3.org/2001/XMLSchema#string>.
_:33237857 <http://www.heppnetz.de/ontologies/vso/ns#engineDisplacement>
_:bnode122646848.
_:33237857 <http://www.heppnetz.de/ontologies/vso/ns#firstRegistration> "2000-
02-17"^^<http://www.w3.org/2001/XMLSchema#date>.
_:216778770 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Test>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testid>
"216778770"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testvehicle> _:39294156.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testdate> "2011-02-
01"^^<http://www.w3.org/2001/XMLSchema#date>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testclassid>
"1"^^<http://www.w3.org/2001/XMLSchema#string>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testtype>
"F"^^<http://www.w3.org/2001/XMLSchema#string>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testresult>
"F"^^<http://www.w3.org/2001/XMLSchema#string>.
_:216778770 <http://vocab.inf.ed.ac.uk/mot/testlocation>
<http://dbpedia.org/resource/CB_postcode_area>.
_:39294156 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vocab.inf.ed.ac.uk/mot/Vehicle>.
_:bnode1824013824 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode1824013824 <http://purl.org/goodrelations/v1#hasValueFloat>
"13399"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode1824013824 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"SMI"^^<http://www.w3.org/2001/XMLSchema#string>.
_:39294156 <http://www.heppnetz.de/ontologies/vso/ns#mileageFromOdometer>
_:bnode1824013824.
_:39294156 <http://vocab.inf.ed.ac.uk/mot/vehicleid>
"39294156"^^<http://www.w3.org/2001/XMLSchema#integer>.
_:39294156 <http://purl.org/goodrelations/v1#hasManufacturer>
<http://dbpedia.org/resource/Peugeot>.
_:39294156 <http://vocab.inf.ed.ac.uk/mot/vehiclemodel> "VIVACITY
50"^^<http://www.w3.org/2001/XMLSchema#string>.
```

```
_:39294156 <http://www.heppnetz.de/ontologies/vso/ns#color> "GREEN".
_:39294156 <http://www.heppnetz.de/ontologies/vso/ns#fuelType>
<http://dbpedia.org/resource/Gasoline>.
_:bnode1305293760 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://purl.org/goodrelations/v1#QuantitativeValueFloat>.
_:bnode1305293760 <http://purl.org/goodrelations/v1#hasValueFloat>
"49"^^<http://www.w3.org/2001/XMLSchema#float>.
_:bnode1305293760 <http://purl.org/goodrelations/v1#hasUnitOfMeasurement>
"CMQ"^^<http://www.w3.org/2001/XMLSchema#string>.
_:39294156 <http://www.heppnetz.de/ontologies/vso/ns#engineDisplacement>
_:bnode1305293760.
_:39294156 <http://www.heppnetz.de/ontologies/vso/ns#firstRegistration> "2003-
02-01"^^<http://www.w3.org/2001/XMLSchema#date>.
```

i http://simile.mit.edu/wiki/SIMILE:About (accessed 23/02/13)
ii http://www.ebusiness-unibw.org/wiki/VSO (accessed 23/02/13)
iii http://www.rdfabout.com/demo/validator/ (accessed 22/02/13)
iv http://www.cytoscape.org/ (accessed 23/02/12)