MATAWS : Multimodal Approach for Automatic WS Semantic Annotation
User Guide for MATAWS 1.0.0

written by Cihan Aksoy

caksoy@uekae.tubitak.gov.tr

Galatasaray University, Computer Science Department, Istanbul, Turkey

TÜBİTAK - The Scientific and Technological Research Council of Turkey, Gebze/Kocaeli, Turkey

License

Mataws - Multimodal Approach for Automatic WS Semantic Annotation

Copyright 2010-2011 Cihan Aksoy & Koray Mançuhan

This file is part of Mataws - Multimodal Approach for Automatic WS Semantic Annotation.

Mataws - Multimodal Approach for Automatic WS Semantic Annotation is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

Mataws - Multimodal Approach for Automatic WS Semantic Annotation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Mataws - Multimodal Approach for Automatic WS Semantic Annotation. If not, see http://www.gnu.org/licenses/.

Table of contents

1 Introduction

MATAWS [1, 2] is an automatic semantic annotation tool for Web service (WS) collections. This document gives an overview for the tool and describes how to install and use it.

2 Prerequisites

All dependencies required by MATAWS are found in the provided bundle. In the next section, we will show how they should be organized. However, some basic needs should be satisfied to run MATAWS properly:

- A host computer with main memory whose size is more than your WS collection,
- Version 6.0 or more recent of a Java Runtime Environment (JRE) (or Java Development Kit (JDK) for developers), which can be obtained from the relevant download area at http://www.oracle.com

If the above resources are available, you are ready to pass to the installation of the tool. But if the size of your WS collection is more than the default heap size of the Java Virtual Machine (JVM), you have to set it up in order to increase the heap size.

The default maximum heap size of the JVM is 64MB for the 32 bits version and %30 more for the 64 bits one. You should increase the maximum heap size if the size of your collection is larger than the default heap size. You can perform that by adding the following arguments into the launcher file (described in the next sections):

```
-Xms128m -Xmx512m
```

The first parameter sets the initial heap size, whereas the second one sets its maximal size. You can increase the maximum heap size by considering the size of your WS collection.

3 Overview

This section briefly gives details about the way MATAWS implements the annotation process. It takes a collection of WSDL files as input and generates a collection of OWL-S files as output. Figure 1 gives an insight of its modular structure, which includes five different components. Among these components, two are using external APIs (Associator and Output Component), one is using an external API for a

small part of its role (Preprocessor Component), whereas the three remaining ones were completely developed by us in Java.

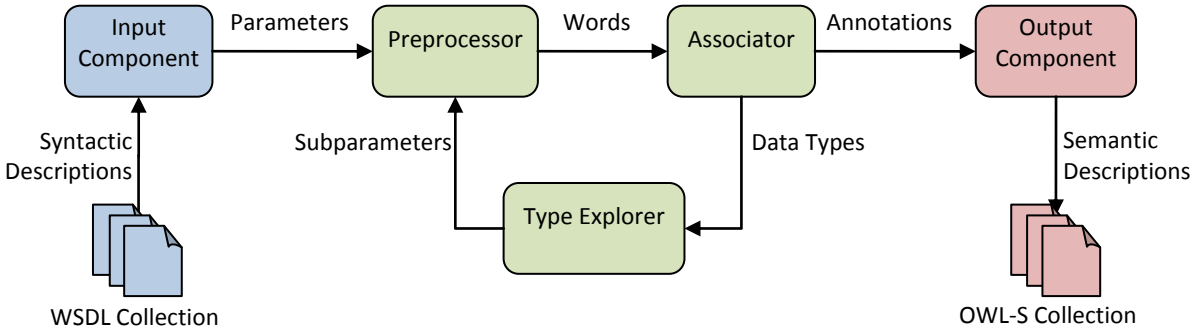

Figure 1 : Architecture of MATAWS [1]

The *Input Component* is in charge for extracting the set of all operation parameters defined in the considered collection of WSDL files. We designed a parser called as SINE (NEXT from now on) [3] able (among other things) to retrieve the parameter names, type names and type structures (in the case of complex types). The *Output Component* is used after the annotation process to generate a collection of OWL-S files corresponding to annotated versions of the input WSDL files. For this purpose, we selected the Java *OWL-S API* [4], which provides a programmatic read/write access to OWL-S service descriptions. Note we plan to add support for WSDL-S and SAWSDL by using other appropriate APIs.

The three remaining components correspond to the core of the annotation process. After the input component has parsed the WSDL files, it fetches parameters information to the *Preprocessor*. This one originally focuses on the parameter names, decomposing, normalizing, cleaning them so that they can be treated by the *Associator*. This component is based on the inference engine Sigma [5], whose role is to associate an ontological concept to a word. If Sigma is successful and manages to return a concept, this one is associated to the considered parameter. After all the parameters of a WS have been annotated, the *Output Component* is used to generate an OWL-S file with both the information contained in the original WSDL file and the selected concepts. However, for various reasons, it is not always possible for Sigma to find a suitable concept for every parameter. In this case, the *Type Explorer* accesses some properties related to the parameter data type, to obtain what we call subparameters. These are then fetched to the Preprocessor and the core processing starts again. In case of repeated annotation failure, this process can be repeated recursively until success or lack of subparameters. This is from where MATAWS takes its "multimodal" property.

4 Release Package Content

The MATAWS release is archived as `MATAWS-1.0.0.zip` and can be found on the lab website at http://bit.gsu.edu.tr/compnet. After downloading and unpacking, you will obtain a bundle structured as shown in Figure 2. It contains libraries in the `lib` folder, including `MATAWS-1.0.0.jar` and other necessary libraries.

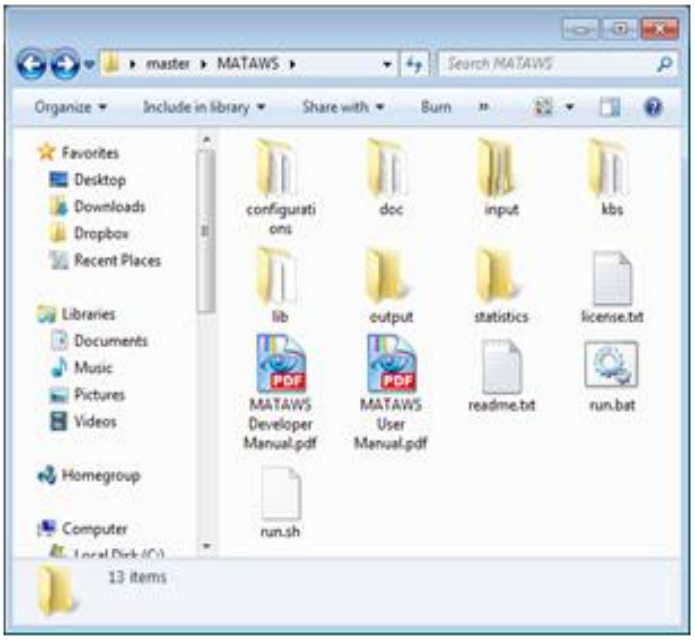

Figure 2: The directory MATAWS

The configuration files are placed under two different folders. Files used by MATAWS to detect stop words and abbreviations during the annotation process are found in the `configuration` folder. Files that hold mappings between the WordNet lexicon [6] dictionary and the SUMO ontology [7] are located under the `kbs` folder. Also, the `input`, `output` and `statistics` folders are meant to contain the collection to be annotated, the annotated collection (i.e. the result of the annotation) and some statistics describing the process, respectively.

5 Usage

After having unpacked the bundle in a suitable place for you, you should only make some configurations before using MATAWS.

5.1 Configuration

As explained in the previous section, there are several configuration folders in the bundle and they are all used for various purposes during the annotation process. The files in `configuration` must be defined in function of the properties of the WS collection to be annotated. This folder contains two files: `Abbreviations.txt` and `StopWords.txt`. They should be modified before starting an annotation

process for a new WS collection by analyzing first manually the parameters of the WS collection.

The file `Abbreviations.txt`, contains a list of the abbreviations used in parameter names, associated to the full version of the word. Since Sigma proposes concepts only for words, and since it is not able to find an ontological concept for a given abbreviation, it is up to the user to associate an appropriate word to each relevant abbreviation.

Before using this tool, you should analyze the parameters of your WS collection, determine what the abbreviations are, and insert not only the abbreviations, but also the corresponding full words into the file. Figure 3 shows an example of this file.

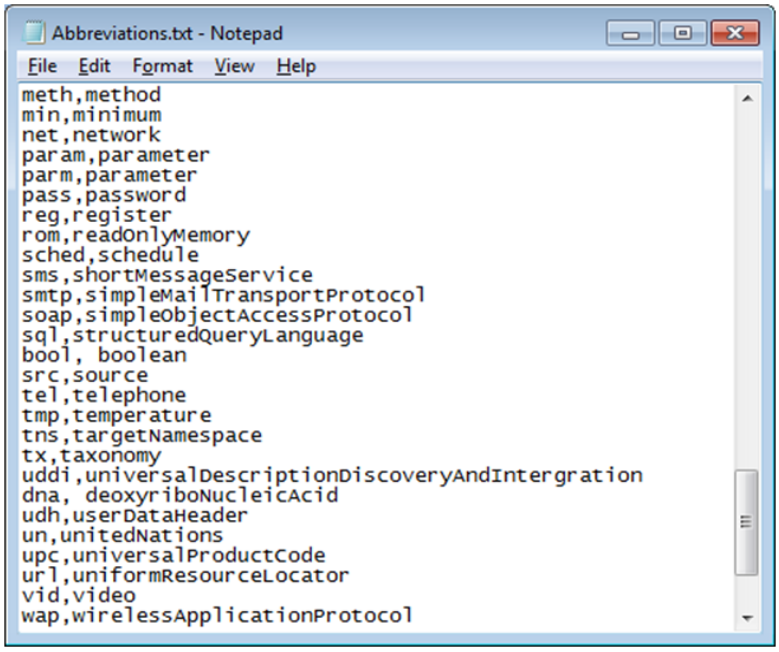

Figure 3: Abbreviation file example

The other configuration file contains stop words appearing in the parameter names. These words do not have a relevant meaning relatively to the collection of interest. That is why our tool detects and removes them from the words list sent to Sigma (even if this one might be able to annotate them). The `StopWords.txt` file is used to define them.

Before annotating your collection, you should analyze the parameters of your WS collection, determine the stop words, and insert them into the file. Figure 4 displays an example of this file.

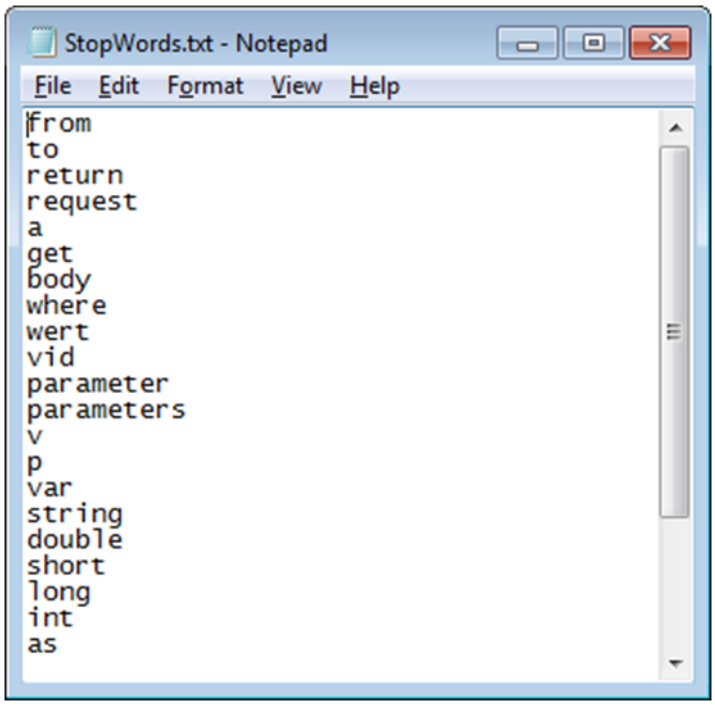

Figure 4: Stop words file example

5.2 Execution and Results

When you configure the environment, you need only to place your WS collection (or WS collections if there is more than one) in the `input` folder, and launch the program. There are two batch files to launch the program, select the one corresponding to your OS. After launching, the tool will take the WS collection(s), process it/them, and put the resulting semantic WS collection(s) into the `output` folder. Also, it will prepare a file including the results of the annotation process. This file is placed in the `statistics` folder.

In the `output` folder, you may observe that the type of the semantic WS files of the collection is OWL-S. MATAWS generates a separate OWL-S file for each WS operation, due to constraints present in some of the libraries used for this version. This limitation may cause more than one OWL-S file generation for a given WSDL file (if it contains several operations). The tool automatically reproduces the folder structure of the input collection, and adds one specific folder to store the different OWL-S files describing each service.

As mentioned before, MATAWS records the annotation processes during the execution. Not only the ontological concepts for a given parameter may be identified, but also an overall result for all parameters of the collection may be examined with the text output placed in the `statistics` folder. In the text output, you will notice that more than one concept may be associated to a given parameter. This is a limitation of MATAWS. After having broken a name down to several words, it might

find one concept for one of these words. We did not implement any automatic process allowing to combine these concepts, for now. We consequently write arbitrarily the first concept in the OWL-S file, but the whole list is available in the text output. We will cope with obtaining a comprehensive concept as a future work. Table 1 explains the terms of statistical part of the text output file.

Notion	Meaning
Total number of parameters	Total number of parameters contained in the WS collection
Number of annotated parameters	Number of parameters which were successfully annotated
Number of non-annotated parameters	Number of parameters the program failed to annotate
Percent of annotated parameters	Proportion of parameters successfully annotated
Number of unique parameters	Number of distinct parameters contained in the WS collection
Number of unique annotated parameters	Number of distinct parameters that were successfully annotated
Number of unique non-annotated parameters	Number of distinct parameters the program failed to annotate
Percent of unique annotated parameters	Proportion of distinct parameters which were successfully annotated
Total number of words	Total number of words identified during the decomposition process
Number of annotated words	Number of words that were successfully annotated
Number of non-annotated words	Number of words the program failed to annotate
Percent of annotated words	Proportion of words which were successfully annotated by the program
Number of different words	Number of distinct words in the collection
Number of different annotated words	Number of distinct successfully annotated words in the collection
Number of different non-annotated words	Number of distinct words the program failed to annotate
Percent of unique annotated words	Proportion of distinct words successfully annotated by the program

Table 1: Description of the statistics file outputted by MATAWS

When you look into the folders `input`, `output` and `statistics` found in the bundle, you will notice that an example is already presented. The `input` folder contains 26 WSDL files of a real world WS collection [8]. The `output` folder contains the result of their annotation: 113 OWL-S files. The `statistics` folder contains the file `statistics_0.txt` with the list of the collection parameters, their

decomposed forms, and corresponding concepts. At the end, you can see the statistical results of the annotation process described above.

References

1. Aksoy, C., et al., *MATAWS: A Multimodal Approach for Automatic WS Semantic Annotation*, in *3rd International Conference on Networked Digital Technologies*2011.
2. Aksoy, C. and K. Mancuhan, *Annotation automatique de descriptions de services web*, 2010, Galatasaray University: Istanbul, TR.
3. Cherifi, C., Y. Rivierre, and J.-F. Santucci, *WS-NEXT, A Web Services Network Extractor Toolkit*, in *5th International Conference on Information Technology*2011.
4. Sirin, E. and B. Parsia, *The OWL-S Java API*, in *3rd International Semantic Web Conference*2004.
5. Pease, A. *Sigma Knowledge Engineering Environment*. 2008 [cited 2010 December]; Available from: http://sigmakee.sourceforge.net/.
6. Miller, A.G., et al. *Wordnet*. 2005 [cited 2010 December]; A lexical database for English]. Available from: http://wordnet.princeton.edu/.
7. Niles, I. and A. Pease, *Towards a Standard Upper Ontology*, in *International Conference on Formal Ontology in Information Systems*, C. Welty and B. Smith, Editors. 2001: Ogunquit, US-ME.
8. Hess, A. *ASSAM (Automated Semantic Service Annotation with Machine learning) WSDL Annotator*. 2004 [cited 2010 December]; Available from: http://www.andreas-hess.info/projects/annotator/index.html.