Using the SEALS Client's Oracle in Interactive Matching

Daniel Faria

May 26, 2016

Abstract

This tutorial explains how matching systems may use the Oracle class from the SEALS client in the Interactive Matching track of the Ontology Alignment Evaluation Initiative.

For instructions on how to wrap your system for the SEALS platform, please refer to the SEALS Client Tutorial.

1 Introduction

The Oracle class of the SEALS client simulates a (single) user that validates mappings. Matching systems can query the Oracle about mapping candidates they generated, as they would a user, and the Oracle will indicate whether those mappings are correct or not (according to the reference alignment of the corresponding matching task).

To simulate the fact that a user may make erroneous validations (depending to his level of expertise about the domain of the ontologies) the Oracle will randomly return erroneous responses, with uniform probability specified at the start of the run (as an input parameter of the SEALS client). Note that, because the Oracle is simulating a single user, asking about the same mapping multiple times to dilute the error is not possible — the Oracle will reply in the same manner it replied previously, if asked about a mapping more than once.

To simulate the fact that a user will generally be able to provide an estimate of his expertise or confidence about the matching task, systems can ask the Oracle about its confidence (the complement of the error rate). This will be constant throughout the matching task.

The Oracle also simulates the scenario where a user is asked which of a set of competing mappings are correct, rather than to validate individual mappings. When queried about a set of up to three competing mappings, the Oracle will count this as a single user interaction, rather than an interaction for each mapping. An example of a set of three competing mappings is: *Muscle* <=> *Muscle*, *Muscle* <=> *Muscle Tissue*, *Striated Muscle Tissue* <=> *Muscle Tissue*. Formally, we can say that a set of mappings is competing if, for every pair of mappings in the set, at least one of the mapped entities is shared by the pair.

2 Interacting with the Oracle class

2.1 Importing the Oracle class

The Oracle class is contained in the SEALS client, and can be imported through:

```
import eu.sealsproject.omt.client.interactive.Oracle;
```

Note that, because the evaluation will be done through the SEALS client, you need not include it in the classpath of your matching tool.

2.2 Simple Check

The simplest way of asking for feedback about a mapping from the Oracle is through the following method:

```
static boolean check(String uri1, String uri2, String rel)
```

where *uri1* is the URI of the source ontology's entity (class, property, or individual) in string form, *uri2* is the URI of the target ontology's entity in string form, and *rel* is the mapping relation, also in string form ("=", "<", or ">").

If the specified mapping is present in the reference alignment, the Oracle will return *true* with probability equal to $1 - e$, where e is the error rate, and *false* with probability e. Otherwise, the Oracle will return *false* with probability $1 - e$, and *true* with probability e.

2.3 Check with Relation

As an alternative to the previous method, systems can use the following method:

```
static boolean check(String uri1, String uri2, Relation rel)
```

where *rel* is given as an instance of the Relation enum (Relation.EQUIVALENCE, Relation.SUBSUMES, or Relation.SUBSUMED_BY) rather than in string form.

Using this alternative requires importing also the Relation enum:

```
import eu.sealsproject.omt.client.Relation;
```

Again, if the specified mapping is present in the reference alignment, the Oracle will return *true* with probability equal to $1 - e$, where e is the error rate, and *false* with probability e. Otherwise, the Oracle will return *false* with probability $1 - e$, and *true* with probability e.

2.4 Check Mapping

Yet another alternative is the following method:

```
static boolean check(Mapping m)
```

where m is an instance of class Mapping, which can be constructed using either of the following constructors:

```
Mapping(String sourceURI, String targetURI, String rel)
Mapping(String sourceURI, String targetURI, Relation rel)
```

Using this alternative requires importing the Mapping class:

```
import eu.sealsproject.omt.client.interactive.Mapping;
```

and if you use the second constructor, also the Relation enum, as specified in subsection 2.3.

Once more, if the specified mapping is present in the reference alignment, the Oracle will return *true* with probability equal to $1 - e$, where e is the error rate, and *false* with probability e. Otherwise, the Oracle will return *false* with probability $1 - e$, and *true* with probability e.

2.5 Check Set of Mappings

The final option for querying the Oracle is the following method:

```
static Set<Mapping> check(Set<Mapping> maps)
```

where *maps* is a set of any number of instances of class Mapping (as per subsection 2.4).

For each Mapping m in *maps*, the method will call $check(m)$. However, to simulate the scenario where the user is asked to choose between competing mappings (as explained in section 1) if *maps* contains 2 or 3 Mappings, and all are competing, the method will count this as a single interaction. If *maps* contains 3 Mappings, but only 2 of them are competing, the method will count this as two interactions. Otherwise, the method will count one interaction for each Mapping in *maps*.

The method will return the subset of Mappings *maps* for which the call $check(m)$ returns *true*, as detailed in subsection 2.4.

2.6 Oracle Confidence

Systems can also ask the Oracle about its confidence (the complement of the error) through the following method:

```
static double confidence()
```

Note that the confidence will be constant throughout each matching task — it will not reflect the classification of individual mappings, but only the overall reliability of the simulated user.