Oscar4 Java API: chemical name dictionaries

Egon Willighagen

Published October 28, 2010

Citation

Willighagen, E. (2010). Oscar4 Java API: chemical name dictionaries. In *chem-bla-ics*. chem-bla-ics. https://doi.org/10.59350/866tq-qv177

Keywords

Oscar, Java, Chebi

Abstract

Besides getting Oscar used by ChEBI (hopefully via Taverna), my main task in my three month Oscar project is to refactor things to make it more modular, and remove some features no longer needed (e.g. an automatically created workspace environment). Clearly, I need to define a lot of new unit tests to ensure my assumptions on how to code works are valid.

chem-bla-ics

Besides getting Oscar used by ChEBI (hopefully via Taverna), my main task in my three month Oscar project is to refactor things to make it more modular, and remove some features no longer needed (e.g. an automatically created workspace environment). Clearly, I need to define a lot of new unit tests to ensure my assumptions on how to code works are valid.

So, what are the API requirements set out? These include (but are not limited to):

- have reasonable defaults
- being able to add custom dictionaries
- easily change the chemical entity recogniser
- plugin text normalization (see Peter's post on UNICODE)

This week I worked on the dictionary refactoring, and talked with Lezan about the ChemicalTagger and trying to get this based on the newer Oscar code (I think we'll be able to finish that today). So, I cleaned up some code I did in the first week, and introduced a Oscar class providing a Java API to the Oscar functionality.

So, to get started with Oscar in your application, you only need to do:

```
Oscar oscar = new Oscar(
  this.getClass().getClassLoader()
);
oscar.loadDefaultDictionaries();
Map<NamedEntity,String> structures =
  oscar.getNamedEntities(
    "Ingredients: acetic acid, water."
  );
```

The ClassLoader is needed because the Oscar class will not generally know how to load custom classes.

You can add additional dictionaries, by implementing the IChemNameDict interface and one or more of IInChIProvider, ISMILESProvider, and ICMLProvider. For example, adding the OPSIN dictionary would extend the above code to:

```
Oscar oscar = new Oscar(
  this.getClass().getClassLoader()
);
oscar.loadDefaultDictionaries();
oscar.getChemNameDict().register(
  new OpsinDictionary()
);
Map<NamedEntity,String> structures =
  oscar.getNamedEntities(
    "Ingredients: acetic acid, water."
  );
```

chem-bla-ics

And, I think the `oscar.getChemNameDict()` method will be renamed to something like `oscar.getDictionaryRegistry()` really soon.