Readme – Alchemy Micro-tutorial for SMART

Install Alchemy

The installation of Alchemy is given in

http://alchemy.cs.washington.edu/user-manual/2Installation.html

1. Download Alchemy bundle from: http://alchemy.cs.washington.edu/
2. Decompress the bundle into a repository called `alchemy,`

In Linux:

1. Type `make depend; make` in the `alchemy/src` directory,

2. If g++ is missing, then install it, for example with the following command:

   sudo apt-get install build-essential

3. If flex is missing, then install it:

   sudo apt-get install flex

4. If flex is missing, then install it:

   sudo apt-get install bison

Once the compilation of the source is done, you can move to `alchemy/bin` directory to start playing with Alchemy (see next Section).

Use Alchemy

Once Alchemy is built, you can find:
1. an executable file “learnstruct” : it is used to learn patterns represented in a Markov Logic Network theory from a set of facts.
2. an executable file “learnwts”: it is used to learn the “godness” of some given patterns, for example from human programmers/analysts/designers.
3. an executable file “infer”: it is used to infer some conclusions (such as high-level events) from some facts (such as low-level events).

Using these executable files, Alchemy can be conveniently used in two phases:

1. A learning phase where, given some input data, we learn the patterns encoded in first-order logic,
2. An inference phase where we apply the patterns over new input data to derive or infer new output knowledge.

These two phases are quickly illustrated below.

Learning Phase

Structure Learning. Let 's first learn some patterns from some facts (e.g. from a database), run:

```
./learnstruct -i alert.mln -o learnedpatterns.mln -t facts.db
```

The file `facts.db` contains a sample of facts, extracted for example from a database.
The file `alert.mln` contains a declaration of predicates we are interested in. Optionally, some patterns can be given to help the reasoning engine.
The file `learnedpatterns.mln` contains the Markov Logic Theory, that is the patterns we were looking for.

Weight Learning. Another way to learn patterns learning can be achieved with weight learning. In weight learning, the input of Alchemy is a file .db (facts.db below) that contains data like rdf triples encoding low-level information, and another file .mln (alert.mln below) that contains a set of potential patterns. The output of weight learning is another file .mln that contains the patterns with associated weights: Each pattern is associated a weight representing the goodness of the patterns.

To test it, run:

```
./learnwts -i alert.mln -o learnedpatterns.mln -t facts.db -ne
alert -noAddUnitClauses
```

The file `facts.db` contains a sample of facts, extracted for example form a database.
The file `alert.mln` contains a declaration of predicates and potential patterns we are interested in.
The file `learnedpatterns.mln` contains the Markov Logic Theory, that is the patterns with the attached weights we were looking for.

Inference Phase

Once, we have some patterns learned by either weight learning or structure learning, we can apply them to new incoming data. Next we apply our patterns encoded in `learnedpatterns.mln` to other facts encoded in `factsForInference.db` in order to derive high-level information that will be written in `alert.results`. Here we will query for alerts So, in your terminal run the following command:

```
 ./infer -i learnedpatterns.mln -e factsForInference.db
-r alert.results -q alert  -maxSteps 1000
```

- The file `factsForInference.db` contains new facts.
- The file `learnedpatterns.mln` contains the Markov Logic Theory of the patterns we want to apply.
- `The` file `alert.results` contains the results. We queried for alerts.

`More can be found in:`

http://alchemy.cs.washington.edu/user-manual/2Installation.html