Lost API documentation

Lost Members

March 10, 2010

Contents

Chapter 1

Setting up the lost framework

After obtaining the lost framework, for instance by checking it out from git, a little configuration is needed to get started.

The file `lost.pl` in the top-most directory of the copy of the framework. In the beginning of the file there are two important facts you may need to change,

```
lost_config(prism_command,'/opt/prism/bin/prism').
lost_config(lost_base_directory, '/home/cth/code/lost/').
```

The option `prism_command` should point to a the main PRISM executable binary.
`lost_base_directory` should be the full path of the directory (including trailing /) containing the `lost.pl` file.

To get started you can examine and run `example.pl` which is also located in the top-most directory.

1.1 Lost model conventions

Each model is located in its own subdirectory of the of the {`lost`}`/models/` directory. So for instance, the sample model called `sample_model1` is located in {`lost`}`/models/sample_model1/`.

To integrate into the framework each model must provide a file called `interface.pl`, which must be located in the same directory as the model. `interface.pl` can then implement various predefined predicates which serves as an entry poinit of using the model. Currently, the only supported predicate is `lost_best_annotation/3`.

In addition models are expected to store switch probabilities the subdirectory `parameters/` of the model directory. Switch parameter files should be given the extension `.prb`.

1.1.1 Model interface predicates

This section describes predicates, that when implemented by the `interface.pl` provided by a model, allows the model provide functionalities to the general framework.

lost_best_annotation(+ParameterFile,+InputFiles,+OutputFilename)
The framework calls this predicate to obtain a "best annotation" from the model. The model is free to provide this annotation in any way it sees fit. It is the models responsibility to save the annotation to `OutputFilename`.

`OutputFilename`: Full filename which the resulting "best annotation" should be saved to.

`ParameterFile`: Is the full name of a file containing the parameters the model should use. The model may disregard this argument if the it does not deal with parameters.

`InputFiles`: Is a list filename, each containing inputs to the model.

Note: I think it might be a good idea to add an "extra options" parameter to the lost_best_annotation, to enable a model to be parameterized.

Chapter 2

Lost shared APIs

To use the lost APIs, the file `lost.pl` located in the top-most {`lost`} directory must be consulted. Then, APIs, which are located in the {`lost`}`/shared` directory can be consulted using the goal,

```
lost_include_api(APIName).
```

where `APIName` is the name of a Prolog file located in the {`lost`}`/shared/` directory except the `.pl` extension.

2.1 interface.pl

The API provides the interface to lost models following the conventions described in section 1.1.

2.1.1 get_annotation_file/4

This API provides `get_annotation_file/4` which is used to retrieve the best annotation generated by a specified model with specified parameters and input sequences. If no such file currently exists, then the model will be run (e.g. the `lost_best_annotation/3` provided by the model will be called).

The generated annotation files are named according to a convention. All annotation files will be placed in the {`lost`}`/sequences/` directory. The `Filename` is construed according to the following convention:

```
{Modelname}_annot_{Id}.seq
```

The first time an annotation is generated the file `annotation.idx` will be created in this directory. This file serves as a database to map filenames of the generated annotation files to the (models ,inputs,probability parameters) that generated the particular annotations. This database file contains Prolog facts on the form,

```
fileid(Id,Filename,Model,SwitchParameters,InputFiles).
```

The annotation index is automatically maintained by `get_annotation/4` and should normally not be edited by hand.

If annotation for a particular run of a model is not present then `get_annotation_file/4` will start a new PRISM process that invokes the `lost_best_annotation` predicate provided be the model `interface.pl` file. By the contract of model conventions, the model will generate the annotation and save it to the file indicated by the provided filename.

2.2 Input-Output API

In this module (`io.pl`), severals predicates are defined to manipulate `*.seq` files :

- loading information from files that extracts from a file data information used as input of models (sequence annotation for example);
- saving information into a file;
- and maybe more.

2.2.1 Loading information from files

- `load_annotation_from_file(++Type_Info,++Options,++File,--Annotation)`: Generate from `File` a sequence of `Annotation`. It is assumed that `File` is composed of terms. `Type_Info` is used to specify what is the format of information in the file
  - `sequence` means that information is stored into a list. For example,

    ```
    data(Key_Index,1,10,[a,t,c,c,c..]).
    ```

  - `db` means that information is represented by a set of range that specified specific zone (coding region for example)

    ```
    gb(Key_Index,1,10).
    ```

  For each `Type_Info`, several options are available represented by the list `Options`. Options available for `sequence`:
  - [] (default): data list must be in position of the terms and all lists of data are append;
  - data_position(Num) permits to specify that the data lists the `Num`th argument of terms;
  - `range(Min,Max)` extracts from the list of the complete annotation the sublist from position `Min` to position `Max`;
  - all_lists: generate a list of each data list associated with a term. Warning: range(Min,Max) not support with this option.

File Example `toto.seq`:

```
data(Key_Index,1,5,[1,2,3,4,5]).
data(Key_Index,6,10,[6,7,8,9,10]).
data(Key_Index,11,15,[11,12,13,14,15]).
```

Results of request are:

```
| ?- load_annotation_from_file(sequence,[data_position(4)],R).
R = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] ?
| ?- load_annotation_from_file(sequence,[data_position(4),range(4,10)],R).
R = [4,5,6,7,8,9,10] ?
load_annotation_from_file(sequence,[data_position(4),all_lists],R).
R = [[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]] ?
```

Options available for `db`:

- [] (default): we suppose that the first and the second element of the term defined the range. By default, a list of 0-1 values is generated, 0 when you are outside a range and 1 you are inside;
- `in_db(Letter)` replaces the default value 1 by Letter into the generated list;
- `out_db(Letter)` replaces the default value 0 by Letter into the generated list;
- `range_position(Min,Max)` allows to specify the position into the terms of the minimal and maximal value;
- `range(Min,Max)` extracts from the list of the complete annotation the sublist from position `Min` to position `Max`;

File Example `toto.seq`:

```
gb(3,5).
gb(7,9).
gb(8,11). % Overlap ;)
```

Results of request are:

```
| ?- load_annotation_from_file(db,[],R).
R = [0,0,1,1,1,0,1,1,1,1,1] ?
| ?- load_annotation_from_file(db,[in_db(c),out_db(nc)],R).
R =  [nc,nc,c,c,c,nc,c,c,c,c,c]?
| ?- load_annotation_from_file(db,[range(3,7)],R).
R = [1,1,1,0,1] ?
| ?- load_annotation_from_file(db,[in_db(c),out_db(nc),range(8,16)],R).
R =  [c,c,c,nc,nc,nc,nc,nc]?
```

2.3 The accuracy API

Not implemented/included yet.

Index