Probabilistic Soft Logic (PSL) Overview and Tutorial

Installation & Prerequisites:

For Probabilistic Soft Logic (PSL), a Java installation is required. PSL is compatible with **Java 8 or later versions**. Thus, it is important to check whether the system satisfies this requirement before starting the installation process. For macOS, Java installation is quite straightforward using Homebrew with the command brew install java.

Writing PSL Rules:

In PSL, **weights** are assigned to logical rules, which specify the level of enforcement of the rules during inference. A higher weight means greater enforcement. A rule has a **body** and a **head**, separated by the symbol "->". Several conditions in the body are combined using the symbol "&".

In PSL, **hinge loss potentials** are used, and adding ^2 to a rule specifies a squared hinge loss, which results in smoother optimization during inference. Rules that end with a period (.) and do not have a weight are considered **hard constraints**, which must be satisfied.

Example rules:

```
1.0: Knows(A, B) & Likes(A, X) -> Likes(B, X) ^2
0.5: ~Likes(A, B) -> ~Likes(B, A) ^2
Knows(A, B) = Knows(B, A).
```

Setting Up Data and Project Files:

Users can look into source code of PSL by cloning its official repository:

- https://github.com/linqs/psl

In addition, example projects are available in this separate repository on Github:

- https://github.com/linqs/psl-examples

These examples will automatically download all the necessary dependencies, making them great for learning and testing. For those who are more familiar with the command line interface, the PSL CLI JAR file can be downloaded from **Maven Central**.

A typical PSL project will include a model file (for example, model.psl) that contains the rules and predicates, as well as a data configuration file.

Predicates and Data Configuration:

The predicates in PSL can be either closed or open predicates:

- **Closed predicates** are completely observed and do not change during inference.
- **Open predicates** are inferred by PSL according to the rules and evidence given.

The observation files are represented as text files with tab separation, where each line is a grounded atom optionally followed by a truth value between 0.0 and 1.0 (for example, Alice Pizza 0.9). The *targets* section specifies the predicates that PSL has to learn, and the truth section specifies the ground truth labels for evaluation or learning the weights of the rules.

Example predicate and data configuration:

```
predicates:
  Knows/2: closed
  Likes/2: open

observations:
  Knows: knows_obs.txt
  Likes: likes_obs.txt

targets:
  Likes: likes_targets.txt

truth:
  Likes: likes_truth.txt
```

Running Inference:

After the rules and data have been determined, the PSL tool moves on to the **Most Probable Explanation (MPE) inference** phase. In this phase, the tool computes the inferred truth values for the target predicates and saves them in an output directory. The output files contain ground atoms and inferred values ranging from **0.0 to 1.0**.

The inference phase can be performed by the PSL CLI using this following command:

```
java -jar psl-cli.jar --infer --model model.psl --data model.data --output results/
```

Example Inference Results:

Using the example model and data files, PSL is able to make new inferences about relationships based on existing evidence. For instance, if Alice likes pizza and sushi and knows Bob, PSL is able to infer that Bob likely likes the same things. If Bob also knows Carol, PSL is able to infer that Carol has weak but non-zero preference values based on the transitive relationship.

Sample observation files:

```
# knows_obs.txt
Alice Bob
Bob Carol

# likes_obs.txt
Alice Pizza 1.0
Alice Sushi 0.8
```

Sample inferred results:

```
Bob Pizza 0.91
Bob Sushi 0.74
Carol Pizza 0.63
```

This shows how the PSL will propagate information through relational structure while accounting for uncertainty.

Summary:

Probabilistic Soft Logic is a flexible method that integrates logical rules with probabilistic inference. By assigning weights to logical rules and employing soft truth values, Probabilistic Soft Logic enables structured inference under conditions of uncertainty and relational dependencies. The integration of expressiveness and optimization makes Probabilistic Soft Logic a viable method for real-world inference tasks.