User Manual of WILL 0.1

Shavlik Group

University of Wisconsin-Madison
http://www.cs.wisc.edu/machine-learning/shavlik-group/

August 15, 2019

Contents

1 Overview

WILL (Wisconsin Inductive Logic Learner) is an ILP engine. Inductive Logic Programming (ILP) induces general rules from specific facts and suggested rule structures.

WILL is based heavily on the Aleph implementation of ILP. [4] The Aleph implementation works by starting with the most specific clause (called the bottom clause) and iteratively evaluating more inclusive versions of this clause, selecting the best version at each step. The best version is the one which most accurately covers the positive examples provided while not covering the provided negative examples.

2 The WILL algorithm

The WILL ILP algorithm implementation is similar to that of Aleph, with some variations. The basic algorithm is illustrated in Algorithm 1.

Algorithm 1 BASIC WILL ALGORITHM

```
1: while not stoppingCondition do // This is the "outer loop"
2:    Select an example e from exampleSet
3:    Find best covering clause based on e // This operation is the "inner loop"
4:    Add best covering clause to the final theory
5:    Remove covered examples from exampleSet
6: Return the final theory
```

The stoppingCondition might be that a certain number of examples are covered by the theory, or that the theory has reached a maximum number of clauses.

There are some possible variations on what constitutes the "best covering clause", but it is generally the most general clause that covers some minimum percent of positive examples while excluding some percent of negative examples.

3 Getting started with WILL

Before you begin, you will need:

- The WILL jar file
- Data files

3.1 Data files naming convention and format

The data files have a very specific naming convention. They must exist in a directory with some name *prefix*, and the files must be named:

- *prefix*_bk.txt, which contains background knowledge
- *prefix*_pos.txt, which contains positive examples
- *prefix*_neg.txt, which contains negative examples
- *prefix*_facts.txt, which contains additional facts
- *prefix*_bkRel.txt which contains additional advice (optional file)

Example data files are provided in the `SampleTestbeds` directory. This document uses the "trains" testbed.

Java-style `//` comments can be used to add comments to the ends of lines in the data files files.

3.1.1 Background knowledge file (*prefix*_bk.txt)

The background knowledge file is structured similarly to that used in Aleph. It consists mostly of mode declarations and "known" rules we give to the system.

The following examples are based on a simple data set, where the goal is to devise a theory that can be used to identify eastbound trains. [2] Mode declarations specify legal formats for predicate heads generated by WILL. A mode declaration might look like:

```
mode: eastbound(+train).
```

Such a rule tells WILL that it is legal to produce a rule such as: