Temporal Relational Theory Learning: The TRAIL System Manual

https://github.com/nkatzz/ORL

November 16, 2020

Nikos Katzouris,
Institute of Informatics & Telecommunications,
National Center for Scientific Research "Demokritos"
nkatz@iit.demokritos.gr

Contents

List of Figures

1 Introduction

This document is intended to be used as a manual for TRAIL, a software package that provides a common interface to a number of & algorithms for Temporal Relational Theory Learning. These algorithms are designed for learning logical theories from data of temporal nature, based on techniques from the fields of Inductive Logic Programming (ILP) [6] & Statistical Relational Learning (SRL) [5]. The learnt models may be used for reasoning with time & events, in applications such as Complex Event Recognition (CER) [4].

CER systems detect occurrences of *complex events* in streaming input, defined as spatio-temporal combinations of *simple events* (e.g. sensor data), using a set of complex event patterns. Since such patterns are not always known beforehand, while existing ones may frequently need to be updated to reflect change in the incoming data characteristics, machine learning algorithms for discovering such patterns from data are highly useful. Several challenges are involved in this type of learning. Scalability is a major requirement, since data collected in applications of temporal nature usually come in large volumes, or even in open-ended data streams. Also, such algorithms should be resilient to noise & uncertainty, which are ubiquitous in temporal data sources [1], while taking into account commonsense phenomena [12], which often characterize dynamic application domains. Finally, Valuable existing expert knowledge about the domain may greatly facilitate learning & reasoning for CER if taken into account. This calls for expressive, yet highly efficient from an operational perspective, event pattern specification languages that allow to easily encode domain principles into usable background knowledge for learning & reasoning.

The learning algorithms contained in TRAIL attempt to address some of these challenges. Following work in logic-based approaches to CER [3, 2], which use first-order logic as a unifying representation language for events, complex event patterns and background knowledge, TRAIL is also based on logic. This allows for direct connections to machine learning techniques from the fields of ILP and SRL. A lot of work behind most of the learning algorithms in TRAIL has been devoted to scalability, with a focus on theory revision techniques, incremental & online learning.

The following algorithms are implemented in TRAIL:

- LearnRevise, which is based on ideas from non-monotonic ILP and incorporates machinery from the XHAIL [13] algorithm, in addition to theory revision techniques. LearnRevise operates in a "one-shot" mode, generalizing from a (typically small) excerpt of data, representative of some application domain. LearnRevise is able to either learn a theory from scratch or revise an existing one.

- ILED, originally introduced in [9], an algorithm that lifts the non-monotonic learning & theory revision techniques on which LearnRevise relies to an incremental learning setting, in order to foster scalability. ILED learns incrementally, by iteratively processing small data "snapshots" (each such snapshot may be e.g. a small data chunk that LearnRevise uses for one-shot learning). The ILED algorithm introduced in [9] was designed for learning sound hypotheses (i.e. theories that correctly account for the entirety of the training data). It is thus not capable to deal with noise, which often makes it less useful in real-life applications. In contrast, although closely related to the original ILED algorithm, the ILED version included in TRAIL abandons the requirement for soundness and instead attempts to learn a theory that minimizes the training error, using theory revision techniques and an iterative hill-climbing search in the space of theories.

- OLED, an algorithm introduced in [10], which learns logical theories in an online fashion, i.e. in a single-pass over a training set.

- WOLED, an algorithm introduced in [8], which learns learns rules' structure & weights in an online fashion. WOLED is also capable of probabilistic inference with the weighted rules.

TRAIL is based on Answer Set Programming (ASP) [11] and all learning algorithms contained in this software package rely on the Clingo[1] [7] answer set solver.

This document contains some material on Inductive Logic Programming, Statistical Relational Learning and Answer Set Programming, but it is by no means an adequate introduction to any of these fields. The interested reader should refer to (e.g.) the books on these topics mentioned above ([6, 5, 11]), or to some other of the numerous resources that are available. The research papers mentioned earlier (and in what follows in this document) may also be helpful. More resources on CER may be found at the website of NCSR's Complex Event Recognition group[2]. **All examples used throughout this document may be downloaded from...**

The rest of the document is structured as follows: In Section...

[1] https://potassco.org/clingo/

[2] http://cer.iit.demokritos.gr/

2 Installation

In order to build TRAIL from source you need to have a JDK installed (e.g. OpenJDK, or OracleJDK), version 8 or higher. You will also need SBT, the Scala Build Tool[3] (version 1.3.9 has been used with the code). TRAIL relies on the Clingo ASP solver[4]. At the time of writing the latest Clingo release is v5.4.0. TRAIL won't work with Clingo version prior to v5.x.x. Due to the fact that Clingo's API is also used in TRAIL, Python 2.7 is also required to use the software.

2.1 Building TRAIL from Source

The source code comes with an installation script, which takes care of several dependencies that are necessary for Clingo v5.4.0, as well as Clingo itself. The script also installs LoMRF[5], a library for Markov Logic Networks, which is used by some algorithms in TRAIL, mostly in order to compare Markov Logic-based to ASP-based tools for statistical relational learning. Using the script is the easiest way to install TRAIL. It has been tested on Ubuntu 18.04, but it should work on other Linux distributions that use apt for package management. If you encounter problems running the script it is recommended that you first try to install Clingo manually, by building it from source. Then comment-out the part of the script that is related to installing Clingo and re-run the installation script to get LoMRF. To build Clingo from source follow the instructions provided from the Clingo team[6], keeping in mind that it is **mandatory** to build Clingo while enabling Python support (Clingo's Python API).

Clone or download the source code from `https://github.com/nkatzz/ORL` and cd into the root directory.

Remark 2.1. In what follows we refer to the root directory where the TRAIL source code has been cloned into by **`$TRAIL_HOME`**. ■

To run TRAIL's installation script (sudo privileges required):

```
> cd $TRAIL_HOME/install-scripts
> ./install.sh
```

The script will create a "`dependencies`" directory in the root directory of the source code, where Clingo will be located. When running TRAIL the `dependencies` directory is assumed by default to be the location where the Clingo tools reside. In particular, the `clingo` executable file that will be used by TRAIL by default will be located in `dependencies/clingo/build/bin`. To make sure that Clingo has been built properly you can cd into that directory and type `clingo --version` in a terminal. The result should look similar to the one below:

```
> clingo --version
clingo version 5.4.0
...
Configuration:  with Python 2.7.17, with Lua 5.2.4
```

If you have Clingo installed on your machine prior to building TRAIL and you wish to use the existing version of Clingo with TRAIL (assuming it is version v5.x.x and it is properly configured),

[3] `https://www.scala-sbt.org/`
[4] `https://potassco.org/clingo/`
[5] `https://github.com/anskarl/LoMRF`
[6] `https://github.com/potassco/clingo`

it is possible at runtime to override the default Clingo build that is generated by the installation script (see Section 10).

2.2 Generating a Jar File

Once installed, the TRAIL tools may be used via a java jar file. To generate the jar file do:

```
> cd $TRAIL_HOME
> sbt assembly
```

The jar file will then be located in the `$TRAIL_HOME/target/scala-2.12` directory (named as `trail-x.x-SNAPSHOT.jar` – for instance, `trail-0.1-SNAPSHOT.jar`).

To run the application:

```
> java -cp $TRAIL_HOME/target/scala-2.12/trail-x.x-SNAPSHOT.jar \
         <Runner> \
         <Mandatory args> \
         <Optional args>
```

where *Runner* is the name of a "main" class in TRAIL, which varies according to the tasks that the user wishes to run. Such main classes, in addition to mandatory & optional runtime arguments will be discussed in detail in the following sections.

Remark 2.2. To simplify the presentation in what follows we use **"`trailrun`"** as a **name alias** for `java -cp path-to-jar/trail-x.x-SNAPSHOT.jar`. For example, with this convention the command template above could had been written as

```
> trailrun <Runner> <Mandatory args> <Optional args>
```

■

2.3 Using TRAIL as a Library

TRAIL may be used as a library (external dependency) available to other projects that may use its functionality. To allow for that type:
`> sbt assembly`

in a terminal from within the root directory on the TRAIL source code.

3 Quick Start

Let us attempt a test run with a very simple concept learning task to make sure that everything has been installed properly.

3.1 LearnRevise

3.2 OLED

3.3 WOLED

4 Formulating Learning Tasks

4.1 Preliminaries

4.2 Language Bias & the Hypothesis Space

4.3 Background Knowledge

4.4 Input Data

4.5 Inference

4.6 A Classical Set-Cover Rule Learning Algorithm for Horn Logic

5 One-Shot Theory Learning

5.1 Learning Theories from Scratch

5.2 Revising Theories

6 Incremental Theory Learning

6.1 ILED-HillClimbing

6.2 ILED-CrossOver

7 Online Learning

7.1 OLED

7.2 Probabilistic Inference & Weight Learning

7.3 Weighted Theories & their Probabilistic Semantics

7.4 Weight Learning

7.5 Structure + Weight Learning

7.5.1 WOLED-MLN

7.5.2 WOLED-ASP

7.6 Online Learning with Expert Advice

7.6.1 The Experts Setting

7.6.2 The Multi-Armed Bandits Setting

8 Data & Applications

9 Command Line Arguments

10 Extending TRAIL

We are now ready to perform learning with OLED. cd to your folder and run the following in a terminal:

```
java -cp oled-0.1.jar app.runners.OLEDDefaultRunner \
--inpath=<PATH TO caviar-bk FOLDER> --delta=0.00001 \
--prune=0.8 --target=meeting --train=caviar-train \
--saveto=<PATH TO SOME LOCATION>/theory.lp
```

To see all available command line arguments do `java -cp oled-0.1.jar -help`. The learnt hypothesis will be saved in `/oledhome/theory.lp`. You may evaluate this theory on the test set as follows:

```
java -cp oled-0.1.jar app.runners.OLEDDefaultRunner \
--inpath=<PATH TO caviar-bk FOLDER> --target=meeting \
--test=caviar-test --evalth=<PATH TO theory.lp>
```

References

[1] Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. Probabilistic complex event recognition: A survey. *ACM Comput. Surv.*, 50(5):71:1–71:31, 2017. 3

[2] Alexander Artikis, Marek Sergot, and Georgios Paliouras. An event calculus for event recognition. *Knowledge and Data Engineering, IEEE Transactions on*, 27(4):895–908, 2015. 3

[3] Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. Logic-based event recognition. *The Knowledge Engineering Review*, 27(04):469–506, 2012. 3

[4] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012. 3

[5] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016. 3, 4

[6] Luc De Raedt. *Logical and relational learning*. Springer Science & Business Media, 2008. 3, 4

[7] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Marius Lindauer, Max Ostrowski, Javier Romero, Torsten Schaub, and Sven Thiele. Potassco user guide, version 2.0. *URL: https://sourceforge. net/projects/potassco/files/guide/2.0/guide-2.0. pdf/download*, 2015. 4

[8] Nikos Katzouris and Alexander Artikis. WOLED: A tool for online learning weighted answer set rules for temporal reasoning under uncertainty. In *KR 2020*, 2020. 4

[9] Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Incremental learning of event definitions with inductive logic programming. *Machine Learning*, 100(2-3):555–585, 2015. 3

[10] Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Online learning of event definitions. *TPLP*, 16(5-6):817–833, 2016. 3

[11] Vladimir Lifschitz. *Answer set programming*. Springer, 2019. 4

[12] Erik T Mueller. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann, 2014. 3

[13] Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009. 3