Castor Manual

Jose Picado

September 16, 2019

1 Command Line Arguments

- *dataModel* <*data_model_file*> (required): JSON file containing mode declarations (language bias).
- *parameters* <*parameters_file*> (required): JSON file containing parameters (explained below).
- *inds* <*inds_file*>: JSON file containing inclusion dependencies.
- *trainPosSuffix* <*train_pos_suffix*>: Suffix of table containing positive training examples.
- *trainNegSuffix* <*train_neg_suffix*>: Suffix of table containing negative training examples.
- *testPosSuffix* <*test_pos_suffix*>: Suffix of table containing positive testing examples.
- *testNegSuffix* <*test_neg_suffix*>: Suffix of table containing negative testing examples.
- *posTrainExamplesFile* <*pos_train_examples_file*>: CSV file containing positive training examples.
- *negTrainExamplesFile* <*neg_train_examples_file*>: CSV file containing negative training examples.
- *posTestExamplesFile* <*pos_test_examples_file*>: CSV file containing positive testing examples.
- *negTestExamplesFile* <*neg_test_examples_file*>: CSV file containing negative testing examples.
- *test*: Test learned definition using testing examples.
- *outputSQL*: Output learned definition in SQL format.
- *sat*: Only build bottom-clause for example specified by argument *e*.

- *groundSat*: Only build ground bottom-clause for example specified by argument *e*.
- *e*: Index of example to build (ground) bottom-clause when *sat* or *groundSat* arguments are specified (default: 0).

2 Parameters

Configuration parameters:

- *dbURL*: VoltDB server URL. (default: 'localhost')
- *port*: VoltDB client port. (default: 21212)
- *threads*: Number of threads; used to parallelize coverage operations. (default: 1)
- *randomSeed*: Random seed. (default: 1)
- *createStoredProcedure*: Create stored procedures that run bottom-clause construction algorithm. (default: true)
- *useStoredProcedure*: Use stored procedures to run bottom-clause construction algorithm. (default: true)

Parameters to restrict the hypothesis space:

- *iterations*: Number of iterations in bottom-clause construction algorithm. Equivalent to maximum depth of variables in a bottom-clause.
- *beam*: Number of candidate clauses to keep. (default: 1)
- *sample*: Number of examples to use when generalizing a clause using ARMG. (default: 1)
- *recall*: Maximum number of literals added to a bottom-clause for each application of a mode declaration. (default: 10)
- *groundRecall*: Maximum number of literals added to a ground bottom-clause for each application of a mode declaration. Ground bottom-clauses are used to evaluate coverage using theta-subsumption. If this parameter is restricted, the result of coverage testing is approximate. (default: Integer.Max_VALUE)
- *queryLimit*: Limit the number of tuples returned by each query made to the database. If this parameter is restricted, the result of coverage testing is approximate because a subset of all the available data may be used. Useful to avoid OverflowException (when the result of a query is bigger than the allowed max). (default: Integer.Max_VALUE)

Parameters to control the search strategy:

- *evalfn*: Scoring function used to evaluate clauses. Possible values are: 'coverage', 'precision', 'recall', and 'f1'. (default: 'coverage')

- *reductionMethod*: Function used in negative reduction. Possible values are: 'consistency', 'precision', and 'none'. With 'consistency', Castor looks for the first literal in the clause such that the negative coverage of the clause up to this literal is equal to negative coverage of the original clause. With 'precision', Castor looks for the reduced clause that delivers the best precision. With 'none', Castor does not reduce the clauses. (default: 'consistency')

Parameters to set the minimum criteria that clauses must satisfy before being included in the learned definition:

- *minprec*: Minimum precision that a clause must satisfy to be included in the learned definition (computed based on uncovered positive examples). In other words, how precise each clause should be. (default: 0.5)

- *minrec*: Minimum recall that a clause must satisfy to be included in the learned definition (computed based on all positive examples). In other words, the minimum percentage of positive examples that a clause should cover. (default: 0).

- *minPos*: Minimum number of positive examples that a clause must cover to be included in the learned definition. (default: 2)

Parameters to control sampling:

- *samplingMethod*: Sampling method used to sample the tuples that are used to create literals in bottom-clause construction. The number of tuples sampled for each application of a mode declaration is controlled by the *recall* parameter. Possible values are: 'naive', 'olken', 'stream', 'stratified'. (default: 'naive')

- *sampleGroundBottomClauses*: Apply sampling when building ground bottom-clauses. The sampling method is specified by the *samplingMethod* parameter. The number of tuples sampled for each application of a mode declaration is controlled by the *groundRecall* parameter. If this parameter is restricted, the result of coverage testing is approximate. We recommend to set this and the *sampleInCoveringApproach* parameter to the same value. (default: false)

- *sampleInCoveringApproach*: Apply sampling when building ground bottom-clauses that are used to evaluate coverage of examples in the covering approach (where a positive example is discarded if it is covered by the current definition). The sampling method is specified by the *samplingMethod* parameter. We recommend to set this and the *sampleGroundBottomClauses* parameter to the same value. (default: false)

- *sampleInTesting*: Apply sampling when building ground bottom-clauses of testing examples. The sampling method is specified by the *samplingMethod* parameter. The number of tuples sampled for each application of a mode declaration is controlled by the *groundRecall* parameter. If this parameter is restricted, the result of coverage testing of test examples is approximate. (default: false)

Parameters to learn over heterogeneous databases (CastorX):

- *matchingLiteralPrefix*: Prefix of relation names for matching relations. (default: '_m').
- *allStableCoverageInTraining*: Use modified coverage semantics during training. (default: false)
- *allStableCoverageInTesting*: Use modified coverage semantics during testing. (default: false)

Other parameters:

- *useInds*: Use inclusion dependencies in bottom-clause construction and generalization. See the *inds* command line argument. (default: true)
- *shuffleExamples*: Shuffle the order of positive and negative examples. (default: false)
- *randomizeRecall*: Shuffle the order of returned tuples of every query made during (ground) bottom-clause construction when using naive sampling. (default: false)

3 Castor Assumptions

Castor makes the following assumptions.

- The schema contains unique relation names.
- All attributes in schema are strings.
- Only one attribute is input (+) in mode declarations (language bias).