Software Documentation : Java Custom Component Competition Predictor

This page last changed on Oct 02, 2008 by ivern.

1. Scope

1.1 Overview

This component provides a concrete implementation of the Predictor Framework API, specialized to predict the outcome of a component competition.

1.1.1 Version

1.0

1.2 Logic Requirements

The component provides specialized implementations of the Situation and Prediction interfaces, as well as a simple Predictor. It also includes SituationGenerators and a PredictionComparator for analysis purposes.

1.2.1 Component Competition Situation

The component provides a Situation bean for accessing the data associated with a component competition.

This Situation implementation contains the following data:

Field	Type
Component Id	Integer
Component Version Id	Integer
Project Id	Integer
Catalog	String
Component	String
Version	String
Project Category	String
Project Status	String
Posting Time	Date and Time
End Time	Date and Time
Scorecard Id	Integer
Number of Final Fixes	Integer
Prize	Double
Is Rated	Boolean
Is DR	Boolean
DR Points	Integer

Description	String
Keywords	Zero or more Strings
Technologies	Zero or more Technologies
Participants	Zero or more Participants

A Technology contains the following data:

Field	Type
Name	String
Description	String

A Participant contains the following data:

Field	Type
User Id	Integer
Rating	Integer
Reliability	Double
Auto Screening Result	String
Screening Score	Double
Passed Screening	Boolean
Score Before Appeals	Double
Score After Appeals	Double
Passed Review	Boolean
Number of Appeals	Integer
Number of Successful Appeals	Integer

1.2.2 Component Competition Fulfillment Prediction

This Prediction implementation contains a single double precision value, which is the expected number of submissions that pass review.

1.2.3 Component Competition Simple Predictor

This class provides a simple Predictor<ComponentCompetitionSituation, ComponentCompetitionFulfillmentPrediction> implementation whose predict() method returns a ComponentCompetitionFulfillmentPrediction whose value is the sum of the reliabilities of all registered competitors.

Due to the stateless nature of this algorithm, no training is needed.

1.2.4 Component Competition Simple Average Predictor

This class provides an implementation of Predictor<ComponentCompetitionSituation, ComponentCompetitionFulfillmentPrediction> that holds instances of zero or more predictors of the same type.

Special care needs to be taken to ensure that this class works properly with PredictorManager, by allowing it to be constructed from configuration (including its contained predictors).

1.2.4.1 Prediction

The predict method gets a prediction from every contained predictor. It then returns the mean of the predictions. If there are three or more predictions, the highest and lowest one get discarded before the mean is calculated.

1.2.4.2 Training

Training methods for this predictor simply call the equivalent training methods for all contained predictors.

1.2.4.3 Readiness

This predictor is ready if and only if it contains one or more predictors and all of its predictors are ready.

1.2.5 Component Competition Situation Generators

The component will provide three situation generators.

1.2.5.1 Component Competition Situation Timeline Generator

This generator can be initialized with a ComponentCompetitionSituation and a range of acceptable durations for its timeline, as well as a stepping parameter. It generates all of the Situations whose timeline falls within the accepted range and is of the form (minimum timeline + N * step), where N is a non-negative integer number.

Timeline manipulation should be performed by adjusting the end date and time.

1.2.5.2 Component Competition Situation Prize Generator

This generator can be initialized with a ComponentCompetitionSituation and a range of acceptable first place prizes, as well as a stepping parameter. It generates all of the Situations whose first place prize falls within the accepted range and is of the form (minimum prize + N * step), where N is a non-negative integer number.

1.2.5.3 Component Competition Situation Timeline and Prize Generator

This generator can be initialized with a ComponentCompetitionSituation, a range of acceptable durations for its timeline, and a range of acceptable first place prizes, as well as stepping parameters for each range. It generates all of the Situations whose timeline falls within the accepted range and is of the form (minimum timeline + M * step), and whose first place prize falls within the accepted range and is of the form (minimum prize + N * step), where M and N are non-negative integer numbers.

Timeline manipulation should be performed by adjusting the end date and time.

1.2.6 Component Competition Fulfillment Prediction Comparators

The component provides two comparator implementations to sort predictions with. Each implementation should have a constructor that allows a user to create comparator objects that take into account certain minimum requirements for Situations to be used when comparing them.

1.2.6.1 Component Competition Fulfillment Prediction Timeline Comparator

Given a range of acceptable predictions (defined by a minimum and maximum desired prediction) for the number of submissions that pass review and a target prize, this comparator operates as follows (in order of importance):

1. Predictions in range < predictions above the range < predictions below the range
2. Prizes lower than or equal to the target < prizes higher than the target
3. Within the categories determined by rules 1 and 2, predictions are sorted by their situation timeline duration (lowest to highest).

1.2.6.2 Component Competition Fulfillment Prediction Prize Comparator

Given a range of acceptable predictions (defined by a minimum and maximum desired prediction) for the number of submissions that pass review and a target timeline duration, this comparator operates as follows (in order of importance):

1. Predictions in range < predictions above the range < predictions below the range
2. Timelines shorter than or equal to the target < timelines longer than the target
3. Within the categories determined by rules 1 and 2, predictions are sorted by their prize (lowest to highest).

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used to predict the outcome of TopCoder component competitions.

1.5 Future Component Direction

More complex Predictor implementations have been developed separately, and they will be integrated with this component in a separate competition or set of competitions.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

This component is an implementation of the interface provided by the Predictor Framework component. Read its design documentation for more information.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5

2.1.4 Package Structure

com.topcoder.predictor.impl.componentcompetition

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

The simple Predictor implementations need to be usable with PredictorManager, which will require configuration.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

Frameworks: Predictor Framework 1.0

3.2.2 TopCoder Software Component Dependencies:

- Predictor Framework 1.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- RedHat Enterprise Linux 4
- Sun JDK 1.5

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.