Toolkit to Support Intelligibility in Context-Aware Applications

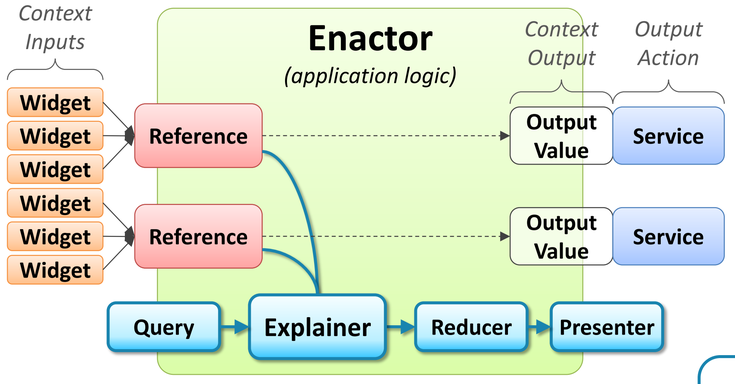
Brian Y. Lim, Anind K. Dey

{ byl, anind } @cs.cmu.edu

• Human-Computer Interaction Institute

UBICOMP2010

To help users better understand and trust context-aware applications, these applications should be *intelligible*; they should provide explanations about what they know, and their behavior. We have developed an Intelligibility Toolkit to support the implementation of 8 types of explanations for the 4 most popular decision models used in context-aware applications. For this demonstration, we present 4 applications showing how the Intelligibility Toolkit can be used to generate and provide explanations across decision models and application domains.



Toolkit Requirements

R1) **Lower barrier** to providing explanations

R2) **Flexibility** of using explanations

R3) Facilitate **appropriate** explanations automatically

R4) Support **combining** explanations

R5) **Extensible** across
- **Explanation** types
- Application (decision) **models**
- **Provision** styles

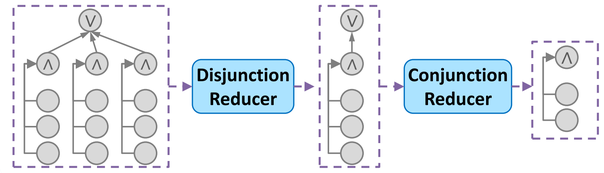
(2) Explainers

Explanation Types

Model-Independent	**Input**	*What does the application use to sense events?*
	Output	*What predictions and actions can it make?*
	What	*What is the current value of the output?*
	What If	*If A changes, what would it do?*
Model-Dependent	**Why**	*Why did it do X?*
	Why Not	*Why didn't it do Y?*
	How To	*How does it distinguish among output values?*
	Certainty	*How certain (confident) is it of this output value?*

Explainers support the automatic generation of 8 explanation types from supported decision models. 4 explanation types are model-dependent, while 4 are not. Explainers take Queries and produce Explanation Structs.

(3) Reducers



Reducers simplify explanations so that they are easier for users to assimilate. There are two types: disjunction and conjunction, to reduce the number of reasons, and length of reasons, respectively.

(4) Presenters

Text Presenter **Diagram Presenter**

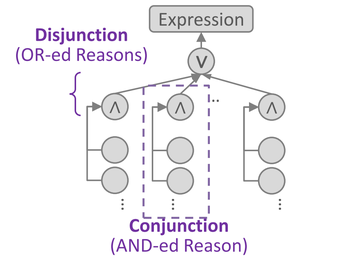
Presenters allow explanations to be rendered in different styles.

(1) Queries

Query	**Inputs, Outputs, What, Why, Certainty**
AltQuery + Alt Output value	**Why Not, How To**
WhatIfQuery + Alt Input values	**What If**

Queries encapsulate different question types to put into Explainers. Some queries can take parameters to specify questions (*e.g.*, asking Why Not about a target value).

Explanation Struct in Disjunctive Normal Form

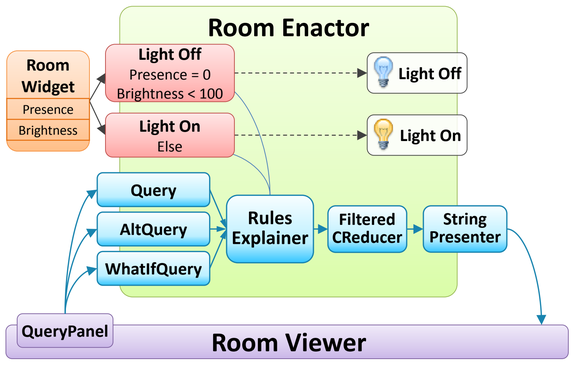


Auto-Light Living Room

Rules

This takes two factors (Presence and Brightness) to determine whether to turn the light on in a living room. The light would be off if Presence = 0 (i.e. no one in the room) or the detected brightness measure is less than 100 (out of 255).

- **QueryPanel** to receive user interactive queries
- **RulesExplainer** to generate explanations from rules
- **StringPresenter** to generate text output for the UI

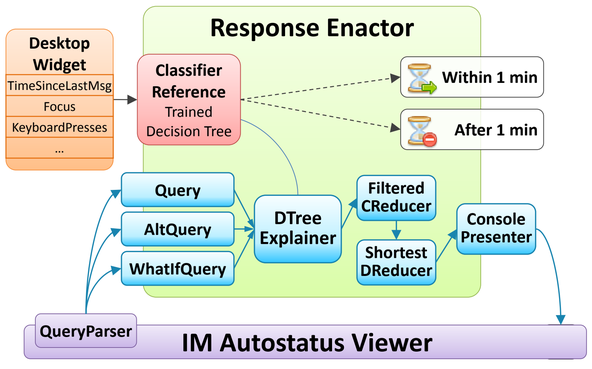


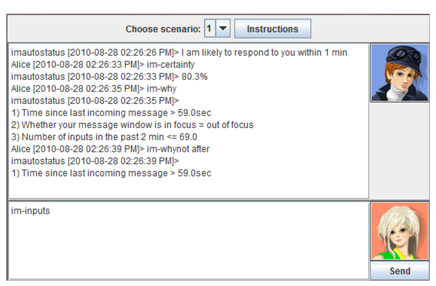
IM Response Prediction

Decision Tree

This predicts when a buddy will respond to a message. It is trained on an existing dataset from [Avrahami et al. 2006] to build a decision tree. It takes desktop-based sensor inputs and makes response predictions (within/after 1 min) .

- **QueryParser** to parse user text input as queries
- **DTreeExplainer** to generate explanations from the decision tree
- **ShortestDReducer** returns the shortest reason when multiple traces are computed (*e.g.*, for Why Not and How To),
- **FilteredCReducer** to simplify the explanations by showing only more easily understood features
- **ConsolePresenter** to generate text output for the UI



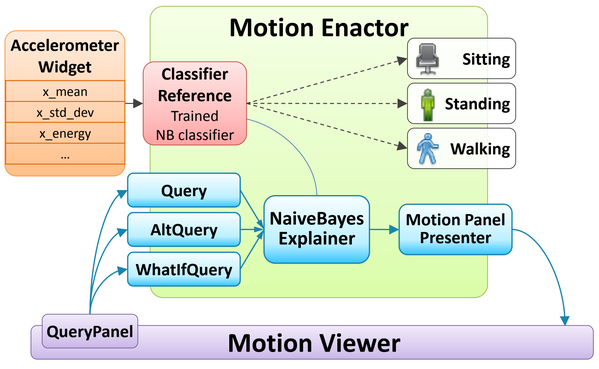


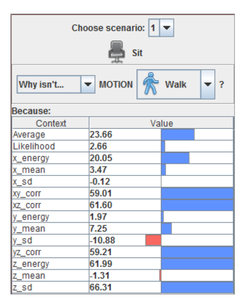
Mobile Phone Accelerometry

Naïve Bayes

This is a physical activity recognizer that uses the accelerometer on a Google Android mobile phone to infer whether the user is sitting, standing, or walking.

- **QueryPanel** to receive user interactive queries
- **NaiveBayesExplainer** to generate explanations from the naïve Bayes classifier
- *No Reducer*
- **MotionPanelPresenter** to represent the phone UI.





Home Activity Recognition

Hidden Markov Model

This uses the dataset from [Kasteren et al. 2008] about domestic activity, and train a HMM. The application takes 14 binary input sensors and infers which activity (out of 7) the user is performing. Explanations are presented by sensors and by time.

- **QueryPanel** to receive user interactive queries
- **HMMExplainer** to generate explanation from the HMM
- **TimeCReducer** to aggregate evidences of sensors across time
  SensorCReducer to aggregate time-step evidences of each sensor
- **FloorplanPresenter** to render evidences as bubbles in a floorplan
  TimeBarPresenter to render time-step evidences of each sensor