Toolkit to Support Intelligibility in Context-Aware Applications

Presented by Mary Salinas

WORCESTER POLYTECHNIC INSTITUTE 1865 WPI

What problem are they trying to solve?

- Context-aware applications are great for helping users to automatically serve users better but the complexity of models can make it difficult for users to understand them.
- Users can become frustrated and lose trust in the applications. Applications must be intelligible and provide explanations of context-awareness

What did they do?

- Created an architecture for generating a wide range of explanations including Why, Why Not, How To, What, What If, Inputs, Outputs and Certainty.
- Create a library of reference implementations of explanation algorithms for 4 different models.
- Provide automated support for the common explanations to promote good design practices.

Surveyed existing use of model types

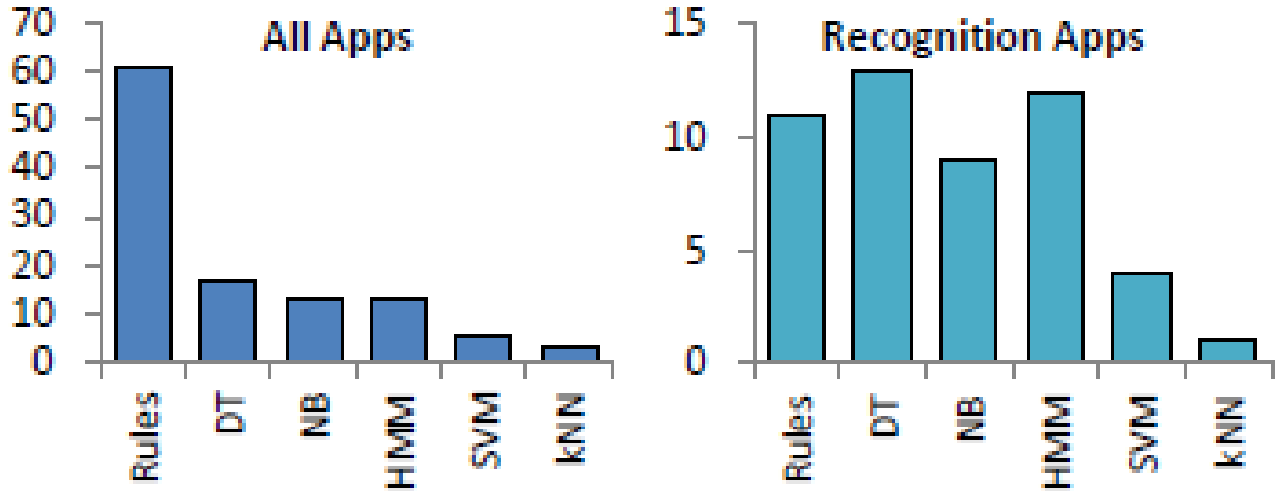

Figure 1: (Left) Counts of model types used in 109 of 114 reviewed context-aware applications. (Right) Counts for 50 recognition applications; classifiers are used most often for applications that do recognition. Key: decision tree (DT), naïve Bayes (NB), hidden Markov models (HMM), support vector machines (SVM), k-Nearest Neighbor (kNN).

Rule-based Decision Model

- Rules are usually if/else logic or simple mapping
- Most popular decision model.
- Used in personalization, activity recognition, monitoring, location guides

Input Conditionals	Output Values
a: Activity = Sitting *b*: Noise = Quiet *c*: Latitude *near* Office's *d*: Longitude *near* Office's *e*: Schedule = In Meeting	☺: Availability = Yes 😐: Availability = Somewhat Not ☹: Availability = Not

Table 1. Pedagogical example of input conditionals and output values for rule and decision tree. This describes an application to infer a user's availability based on his activity, the noise level around him, his proximity to his office (by latitude, longitude), and his schedule.

Input state $(a, \neg b, \neg c, d, \neg e)$**:** user is sitting in a noisy place at latitude not near the office, longitude near the office, and is not in a meeting.

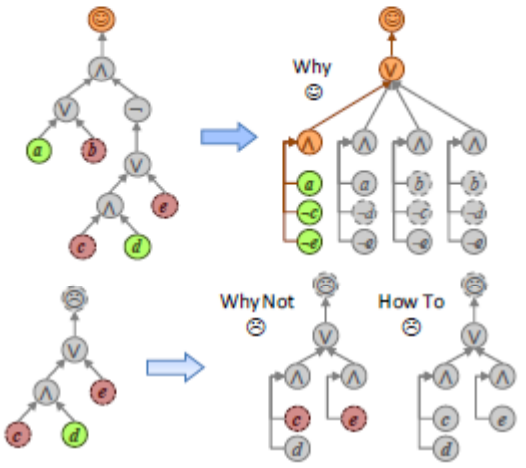

Figure 4: Generating explanations from Rules in DNF.

Decision Tree Model

- Similar to rules, but decisions are made from top down rather than bottom up.
- Decision trees are built from statistical data so they can model certainty from probability of leaves.

Figure 7: IM Autostatus. Demonstration of various explanations from an IM responsiveness prediction plugin that uses a decision tree to predict when a buddy would respond.

Bayes Decision Model

- Naïve Bayes is sum of evidence assumes features are independent of each other.
- Includes prior probabilities of selected class value and from each feature value

Figure 6: Why Not (Left), and How To + What If (Right) explanations for a mobile phone physical activity recognition application using accelerometer data trained with a naïve Bayes classifier. The application has inferred that the user is *Sitting*.

Hidden Markov Model

- Apply weights of evidence as similar to Bayes and include temporal factors as well

Figure 8: Demonstration of Why explanation visualization from an application using a HMM to model domestic activity.

This explains why the application inferred a sequence of Sleeping → Toilet → Toilet → Breakfast → Breakfast in the last 5 min. Evidence due to features (summed across the last 5 min) are indicated by the area of bubbles around the corresponding sensors in the floorplan. Evidence for each sensor across time is revealed in a tooltip.

We can see that the Hall Bedroom Door being open is a strong indicator of inferring the sequence. The door being open is a stronger indicator than it being closed 4 min ago. The microwave is another strong indicator (biggest bubble in top right corner).

Implementation

- Implemented in Java based on Enactor and Context Toolkit created by Anind.
- Explainer to generate explanations for model-independent types and one Explainer for each of the 4 decision model types.
- Reducer to remove explanations that include too many reasons or each reason is too long.
- Presenter renders the explanation in form suitable for users. Developers can build several Presenters if needed.

Architecture for Rules and Classifiers

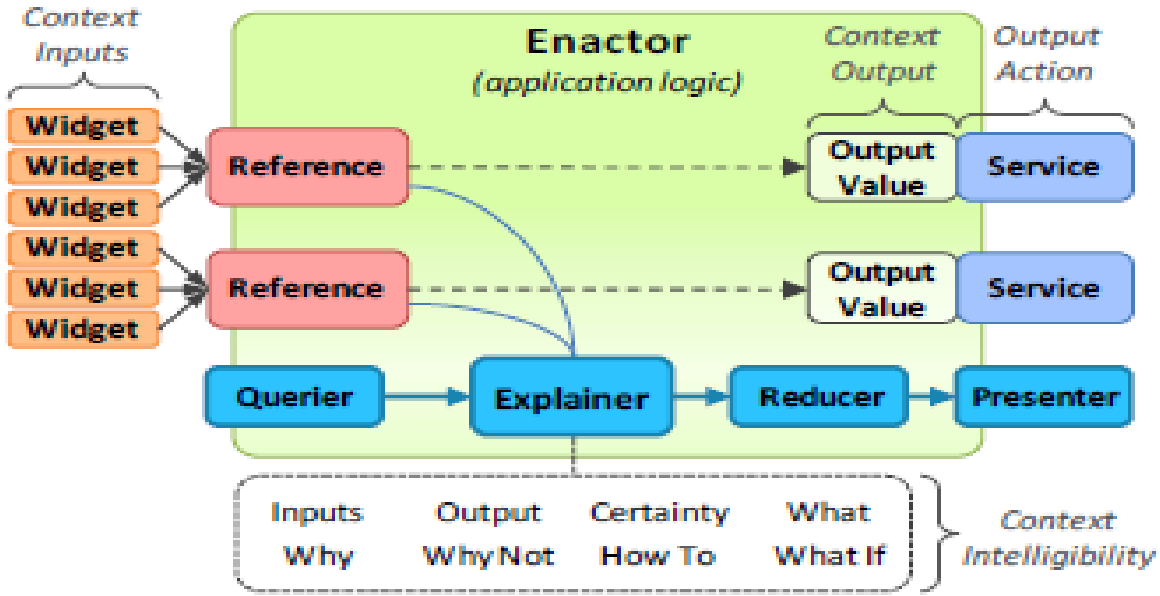

Figure 2: Architecture for handling rules and classifiers. The Intelligibility Toolkit adds four components to the Enactor framework of the Context Toolkit. Users ask for explanations with Querier, and invoke Explainer to generate explanations. The explanations may be simplified with a Reducer, and rendered through a Presenter.

Claimed Results

- Allows for fast prototyping of context-aware applications
- Provides lower barrier to providing explanations
- Provides flexibility of using explanations
- Facilitate appropriate explanations automatically.

What has happened since then?

- Active development is continuing by Lim at http://www.contexttoolkit.org/
- Lim's home page: http://www.brianlim.net/

Any Questions?