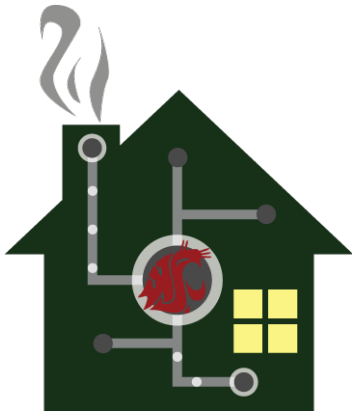
CASAS Prompter Manual

Version 1.1
August 3, 2012



1 Introduction

The Prompter system has four main modes. The *learn* mode learns patterns about when an activity occurs. The *monitor* mode uses the learned activity patterns to predict the occurrence of an activity and issue appropriate prompts. The *generate* mode is used for testing; it generates synthetic data based on a given set of activity patterns. The *test* mode evaluates learned patterns against given sensor data. These modes are described in more detail below, along with installation instructions, the input data format, and the activity pattern format.

2 Installation

The Prompter system is available at the CASAS website (http://ailab.wsu.edu/casas) under the Tools section. Download the latest version and unzip. To compile the system, go to the `src` directory and type "make" and then "make install". This will put the "prompter" executable in the `bin` directory. Note that the prompter requires the gloox library (available at http://camaya.net/gloox).

The `sample` directory contains sample configuration, data and pattern files. The `config` file contains a sample CASAS configuration consisting of the sensors, activities, promptable activities, data file names and pattern file name. The `sampledata` file contains a sample of real sensor data, and the `patterns.out` file contains the patterns learned from the `sampledata` file for each activity in the `config` file. A set of test patterns are provided in the `testpatterns1` file. Data generated by the prompter (in generate mode) from these test patterns are provided in the `sampledata1.out` file. The patterns learning by the prompter (in learn mode) using the `sampledata1.out` file as input are provided in the `testpatterns1.out` file. All of the above use the same `config` file, but with different settings for the "data" file name and "pattern" file name configuration parameters.

The `prompts` directory contains two scripts for generating audio prompts as well as MP3 versions of the prompts for all of the activities listed in the `sample/config` file. These prompts were generated using the "say" program available on Mac OS X. The `generate.sh` script also requires the "Samantha" voice (see General Settings → Speech on Mac OS X) and the

"lame" program (available at http://lame.sourceforge.net). Three prompts are generated for each activity, and the audio files should be named in the format <activity>_#.mp3, where # is 1, 2 or 3. These prompts should be placed in an appropriate directory so that the program listening for prompt events can play the appropriate prompt audio file.

The prompter assumes it will be operating in an environment with three other agents: activity recognition (AR), chime, and prompt. In monitor mode, the prompter waits for a message from the AR agent, which indicates a certain activity has begun. The prompter also monitors the chime agent for a message marking the passing of a minute, at which point the prompter generates appropriate time-based activities (e.g., beginning of an hour, day, month, year, etc.). The activities from AR and the chime agent can initiate prompts for other activities, which are sent out on the prompt channel, to be read by the prompt agent. The prompter also listens on the prompt channel for user responses from the prompt agent (e.g., "cancel", "okay"). These other agents are provided as separate tools from CASAS.

3 Input Data

In *learn* mode input to the prompter consists of a configuration file and one or more sensor data files. The output of *learn* mode is a set of activity patterns, whose format is described in section 3.3. In *monitor* mode the prompter inputs the configuration file and the activity patterns file. In *monitor* mode the prompter receives sensor events and their associated activity from appropriate channels within the CASAS middleware, and issues prompts on a designated prompting channel for execution within the environment. In *generate* mode the prompter inputs the configuration file and a set of activity patterns and outputs sensor events consistent with the patterns. In *test* mode the prompter inputs the configuration file, a set of activity patterns, and one or more sensor data files, and outputs statistics about the accuracy of the activity patterns for predicting (and thus prompting for) activities.

3.1 Configuration File

The configuration file specifies several parameters used by the prompter. Each line of the file is in the format: *ParameterName ParameterValue(s)*. The following sections describe the different parameters and their values. See the `sample/config` file for an example.

3.1.1 data

The "data" parameter is followed by a space-separated list of sensor data files. For *learn* mode, the data from these files are used to learn patterns for each activity. For *generate* mode, the generated sensor events are written to the first file name in the list. For *test* mode, the data from these files are used to evaluate given patterns.

3.1.2 sensor

The "sensor" parameter is followed by a space-separated list of sensors available in the environment and that can appear in the sensor data files.

3.1.3 activity

The "activity" parameter is followed by a space-delimited list of activities that can appear in the sensor data files. The prompter will learn a pattern for each of these activities and output these patterns to the filename specified for the "patterns" parameter.

3.1.4 prompt
The "prompt" parameter has the same format as the "activity" parameter, but specifies just those activities for which a prompt will be generated if it does not occur within the time specified by its learned pattern.

3.1.5 pattern
The "pattern" parameter is followed by the name of the file into which the activity patterns are written (in *learn* mode) and read from (in *monitor*, *generate* and *test* modes).

3.2 Sensor Data
The sensor data files are generated from the smart environment and read into the prompter during *learn* mode. In this section we describe the format of this data. Typically these datasets are both written and read automatically, so the user need not understand the sensor data format to use the prompter. A sample sensor data file is in `sample/sampledata`. Additional datasets are available at the CASAS website (http://ailab.wsu.edu/casas).

The prompter reads in a textual representation of a sequence of sensor events. The prompter assumes that the events are given in temporal order. If multiple sensor data files are input, then they must not overlap temporally, and their order in the "data" parameter of the configuration file must be in temporal order. Each line in a sensor data file contains a single sensor event in the format:

Date Time HighLevelSensorID LowLevelSensorID SensorValue Label

3.2.1 Date
The date refers to the specific day on which the sensor event occurred. The date is in the format *yyyy-mm-dd*.

3.2.2 Time
The time refers to the time of day at which the sensor event occurred. The time is in the format *hh*<:*mm*:*ss*.*x**>, where the minutes, seconds, and milliseconds are optional and an arbitrary precision is allowed.

3.2.3 HighLevelSensorID
Each sensor that is used in the dataset has a specific ID which is represented by a string. The high level ID is an abstract representation of the sensor, often a room name combined with sensor type.

3.2.4 LowLevelSensorID
The low level sensor ID is a second sensor type that describes the sensor generating the event. The low level ID is a more specific functional description of the sensor, perhaps a specific object or area in a room of the building.

3.2.5 SensorValue
The sensor which generated the current event has a value for this event. Currently the sensor values are represented and processed as string values.

3.2.6 Label

Each sensor event is labeled with the corresponding activity. The interpretation is that this sensor event corresponds to an activity with the label "label". There cannot be a sensor event without an activity label. If there is no specific activity associated with the sensor event, then "Other_Activity" can be used to label this event.

3.3 Activity Patterns

The prompter learns activity patterns for each activity listed in the configuration file. A pattern for activity A consists of a set of relative activities RA, each with an associated mean and standard deviation for the delay in seconds between when the relative activity RA begins and the activity A begins. The specific format for an activity pattern is as follows.

<activity> [<relative_activity> <mean> <standard_deviation>]+

where the "+" means one or more. The possible relative activities for an activity A include all the activities given in the configuration file (other than A), plus some additional periodic activities. These periodic activities include the start of each year, month, day, week and hour, as well as the start of each month of the year (January, February, …, December) and the start of each day of the week (Sunday, Monday, …, Saturday). So, for example, if the activity "Phone_Daughter" were to take place every Monday around 6pm (+/- 10 minutes), the associated pattern would look like:

Phone_Daughter Prompter_Activity_Monday 64800 600

If the user phoned their daughter every Monday and Friday around 6pm, then the pattern would look like:

Phone_Daughter Prompter_Activity_Monday 64800 600 Prompter_Activity_Friday 64800 900

If the user phoned their daughter every evening around 6pm, the pattern would look like:

Phone_Daughter Prompter_Activity_Day 64800 600

And finally, if the user phoned their daughter about thirty minutes after dinner begins each night, then the pattern would look like:

Phone_Daughter Eat_Dinner 1800 300

The method for choosing an appropriate relative activity is described in the next section.

4 Learning Activity Patterns (—learn)

To run the prompter in learn mode, execute the following.

prompter –learn <configuration_file>

The Prompter will learn a pattern for each activity given in the configuration file. For an activity *A*, the learning algorithm chooses a relative activity other than *A* from among the activities given in the configuration file, as well as the periodic relative activities described in section 3.3. For a pattern for activity *A*, a relative activity *RA* is evaluated according to three properties: the likelihood that activity *A* occurs after each activity *RA*, the confidence in the distribution of the occurrence times of *A* relative to *RA*, and the mean delay between *RA* and *A*. Given *m* instances of relative activity *RA* in the sensor data, and *n* instances of activity *A* occurring between two consecutive *RA*s, we estimate the occurrence likelihood as *n/m*. Over all such occurrences of *A* after *RA*, we compute the mean μ and standard deviation σ in seconds of the delay between the beginning of *RA* and the beginning of *A*. We use the standard error $\sigma/\sqrt{n}$ as an estimate of the confidence (smaller the better) that *A* follows μ seconds after *RA*. We use the factor $\sqrt{m-n}$ (smaller the better) to measure the error of predicting *A* after *RA* when, in fact, *A* does not occur after *RA*. We use the factor of $1/\sqrt{\mu}$ to prefer smaller mean delays between *RA* and *A*. Finally, we combine the occurrence likelihood, confidence and mean delay to arrive at a promptability measure $P = (n/m)/\sqrt{\mu}(\sigma/\sqrt{n})\sqrt{m-n} = n^{1.5}/(m\sigma\sqrt{\mu}\sqrt{m-n})$ of how well the relative activity *RA* prompts for activity *A*. If *m=0* or *n=0*, we set *P=0*. If σ*=0*, we set σ*=1*. If μ=0, we set μ=1. If *(m-n)=0*, we set $\sqrt{m-n} = 1$. The relative activity with the highest *P* value, along with its associated mean and standard deviation, are output as the activity's pattern. If two relative activities have the same *P* value, we prefer the one with the smaller mean.

While activity patterns allow for multiple relative activities per pattern, considering all subsets of activities as potential patterns is not feasible. However, we do accommodate such patterns involving subsets of the months of the year (January, February, …, December), the days of the week (Sunday, Monday, …, Saturday), and the hours of the day, since many activities are scheduled relative to specific sets of months, days or hours (e.g., leaving for work at 7am Monday through Friday). To accomplish this, we consider three additional relative activities, month-of-year, day-of-week and hour-of-day, where their *P* values are computed as the sum of the *P* values of each individual month, day or hour within the set that has a significant frequency. If one of these relative activities wins out over all the others, then the output pattern consists of all the individual month, day or hour relative activities whose frequency is in the upper-half of the range of normalized frequencies. So, in our example of leaving for work at 7am Monday thru Friday, we would consider the day-of-week relative activity by summing the high *P* values for Sunday, Monday, …, Saturday. We would assume that the frequencies for Sunday and Saturday are low, and thus not included. Therefore, the final pattern would look as follows (assuming 7am +/- 5 minutes):

> *Leave_for_Work Prompter_Activity_Monday 25200 300 Prompter_Activity_Tuesday 25200 300 Prompter_Activity_Wednesday 25200 300 Prompter_Activity_Thursday 25200 300 Prompter_Activity_Friday 25200 300*

5 Monitoring Activities and Prompting (—monitor)

To run the prompter in monitor mode, execute the following.

> *prompter –monitor <configuration_file>*

In addition to reading the configuration file, the prompter will also read activity patterns from the file specified in the configuration. Therefore, the prompter should have already been run in learning mode so that the activity patterns are available. The prompter then observes the real-time activity recognition *ar* channel. The prompter also observes the *chime* channel for events every minute and uses this to initiate appropriate time-based relative activities (e.g., start of hour, start of day, etc.). If the prompter detects the occurrence of a relative activity *RA* in the pattern of one of the prompt activities *PA*, the prompter then watches for the beginning of *PA*. If *PA* does not occur within $(\mu - \sigma)$ seconds, where μ and σ are the mean and standard deviation from the pattern, then the prompter issues a first prompt on the prompt channel for the user to execute this activity *PA*. If another σ seconds go by without activity *PA* being observed, then the prompter issues a second prompt for *PA*. Finally, if yet another σ seconds go by without observing activity *PA*, the prompter issues a third and final prompt for *PA*, and then returns to watching for the next relative activity.

The prompts are sent on the prompt channel of the CASAS middleware and consist of the activity name and the prompt level (1, 2 or 3). The system assumes that these prompt messages will be appropriately handled by another process, which will most likely play the audio file named <activity>_<level>.mp3, and then provide a pop-up alert to the user with "Cancel" and "Okay" option buttons. The prompter will look for a response from this process, where the prompt-level value of the prompt message has been changed to one of "cancel" or "okay. The "okay" value is ignored, but the "cancel" value will deactivate this instance of the activity prompt and return to watching for the next relative activity.

6 Data Generation (—generate)

To run the prompter in generate mode, execute the following.

prompter –generate <configuration_file> <start_date> <start_time> <end_date> <end_time>

In generate mode, the prompter generates sensor data in the format specified earlier over the given time span and consistent with the activity patterns read in from the "pattern" filename specified in the configuration file. The sensor data is written to the first filename as specified in the "data" parameter in the configuration file. Actual sensor identifiers and values are not generated; "SENSOR SENSOR VALUE" are used as placeholders in the output data. The date and time arguments for the generate mode are in the same format as used for the sensor data: YYYY-MM-DD and HH:MM:SS.0.

7 Testing (—test)

To run the prompter in test mode, execute the following.

prompter –test <configuration_file>

In test mode, the prompter evaluates the given activity patterns on the given data. The prompter reads the configuration file, the sensor data files listed in the "data" parameter, and the activity patterns from the file name in the "pattern" parameter. The prompter then simulates the occurrence of the activities in the data and collects statistics on how well the activity patterns

predict the occurrence of the activity. The test mode outputs statistics for each activity and for all activities combined. The statistics include the activity count, true positives (number of times activity occurred after its pattern's relative activity), false positives (number of times activity did not occur after its pattern's relative activity), and false negatives (number of times activity occurred while not having observed its pattern's relative activity). There are no true negatives in the prompting scenario. For true positives, the test mode also outputs the mean error in seconds between the activity pattern's predicted time of occurrence and the activity's actual time of occurrence.

8 More Information

For more information visit the CASAS website at http://ailab.wsu.edu/casas.