Data Mining Challenges in Automated Prompting Systems

Barnan Das

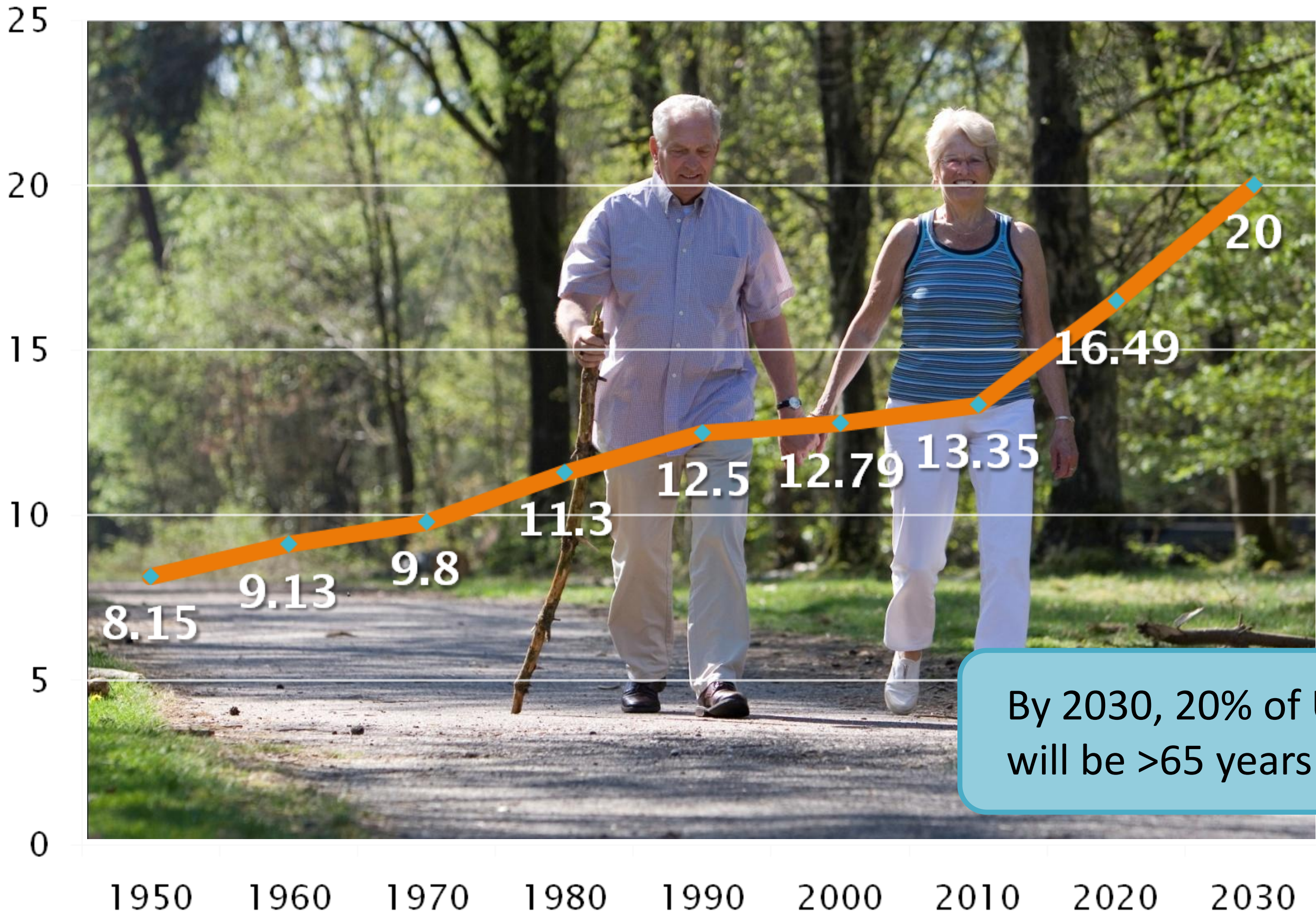
Center for Advanced Studies in Adaptive Systems (CASAS)

Washington State University

February 13, 2011

Americans want
to age in place

Avg. cost of skilled nursing home care: ~$70,000/person/year.



By 2030, 20% of US population will be >65 years of age

Innovative health care technology can help sustain independent lifestyle.

“Prompting Systems”

The Problem

Please turn off the burner.

Sugar is in the cupboard.

Its time to take medicine.

Sam is trying to get in touch with you.

You look tired, why don't you take a nap.

Automatic delivery of verbal or non-verbal interventions that would help a smart home inhabitant in successful completion of daily tasks.

Its John's birthday, you wanna write a card?

Please take a look at the Wattage of the light bulb.

You just picked up the wrong vessel.

Its time to take medicine.

Sam is trying to get in touch with you.

Sugar is in the cupboard.

It would be a good idea to take a walk.

Our Solution

Prompting Users and Control Kiosk

Under development at CASAS, WSU

PUCK

Automated Prompting System

Based on Supervised Learning

System Architecture

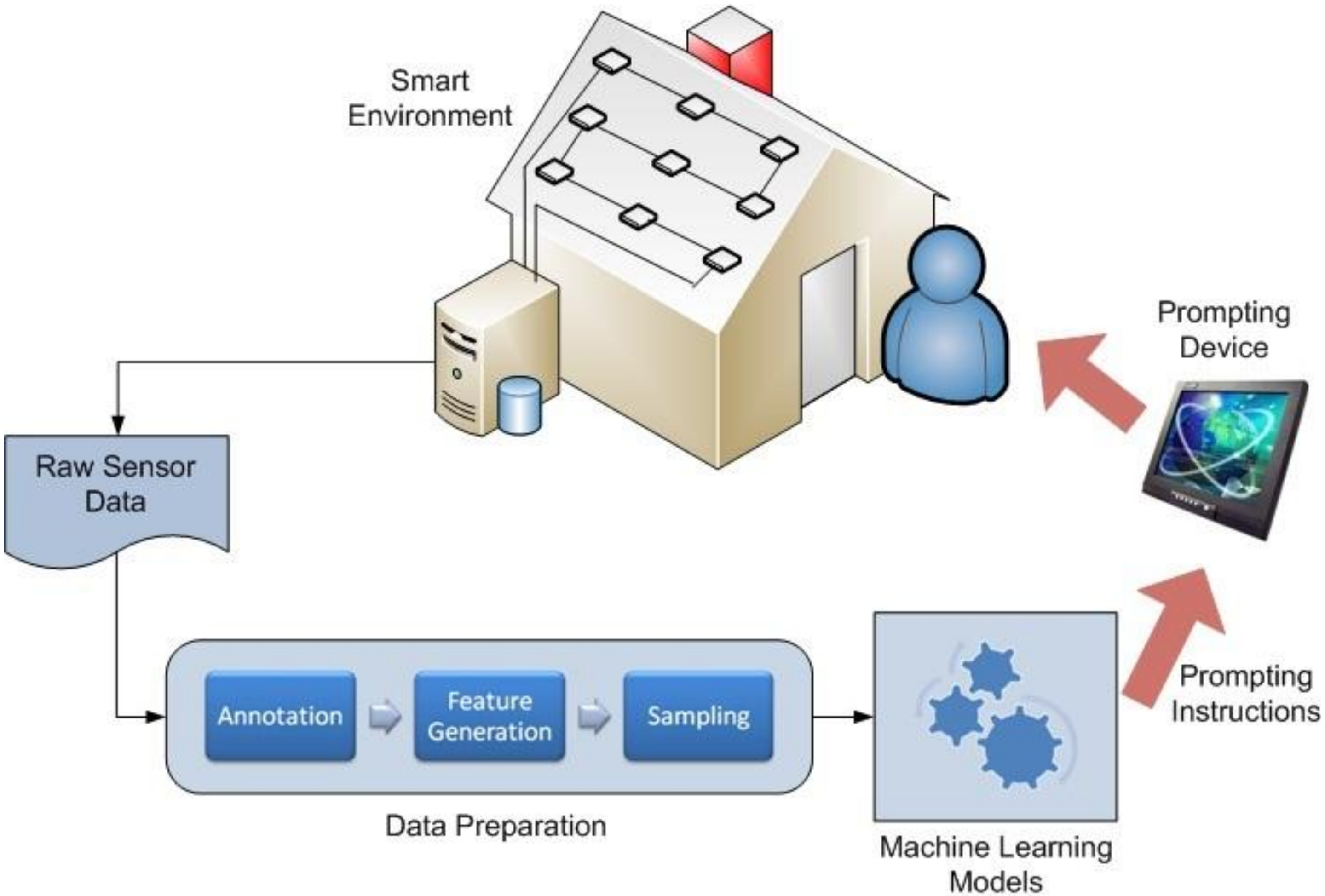


Figure 1: System Architecture of PUCK

Experimental Setup

- Testbed: 2 story apartment in WSU campus
- Sensors: Motion, door, object, temperature, power
- Participants: 128 older adults with mild cognitive disorder
- Activities: Sweeping, Medication, Writing birthday card, Watching DVD, Water plants, Phone call, Cooking, Selecting outfit
- Activities are subdivided into steps.
- Activities monitored via web cam. Experimenter remotely plays (in)direct audio/video cues when an error is detected.
- Human annotators annotate datasets for activities and activity steps.

Feature Generation

Feature #	Feature Name	Description
1	stepLength	Length of the step in time (seconds)
2	numSensors	Number of unique sensors involved with the step
3	numEvents	Number of sensor events associated with the step
4	prevStep	Previous step
5	nextStep	Next step
6	timeActBegin	Time (seconds) elapsed since the beginning of the activity
7	timePrevStep	Time (seconds) difference between the last event of the previous step and the first event of the current step
8	stepsActBegin	Number of steps visited since the beginning of the activity
9	activityID	Activity ID
10	stepID	Step ID
11	M01 … M51	All of M01 to M51 are individual features denoting the frequency of firing of these sensors associated with the step
12	Class	Binary class. 1-"Prompt", 0-"No-Prompt"

Experimentation

- 3 fold cross validation with:
  - Decision Tree (J48)
  - Support Vector Machines (SMO)
  - Ensemble Boosting (LogitBoost)

Performance of Baseline Classifiers

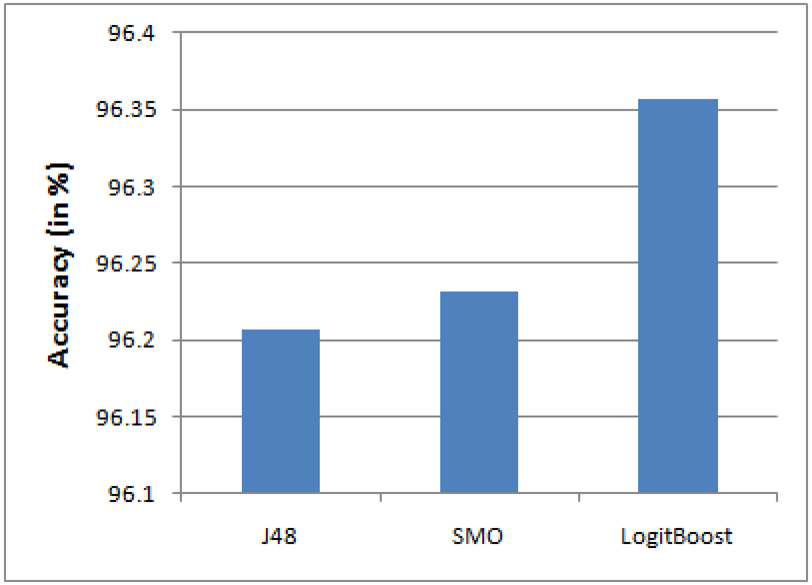


Figure 3: Accuracy Performance for Baseline Classifiers

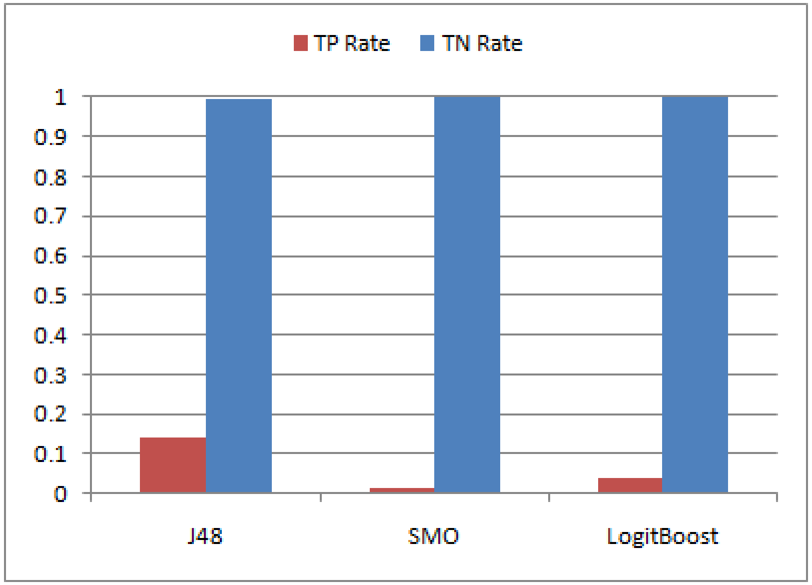


Figure 4: TP and TN Rates for Baseline Classifiers

Failure of Baseline Classifiers

Problem: Highly imbalanced class distribution.

Cause: Vast majority of training situations do not require prompts.

Total # unique steps: 53
steps recognizable by annotators: 38
prompt instances: 149 (3.74% of total # of instances)

Handling Imbalanced Class Distribution

- **Sampling**
- **Cost Sensitive Learning**

Handling Imbalanced Class Distribution

- Sampling
- Cost Sensitive Learning

Sampling

Solution: Boosting prompt situations in the training set without under/over representation.

Technique: Synthetic Minority Over-sampling Technique or SMOTE.

Over-sampling

i. Compute the difference between the feature vector (sample) under consideration and its nearest neighbor.

ii. Multiply this difference by a random number between 0 and 1.

iii. Add the product to the feature vector under consideration.

Under-sampling

Random under-sampling

SMOTE-Variant

Why can't we use SMOTE directly?

- Minority class instances small in absolute number (149 in our case).
- No nearest neighbor with same step of an activity in some cases.

SMOTE-Variant:

i. Randomly pick a minority class instance.
ii. Consider activityID and stepID to find nearest neighbor.
iii. Randomly choose any one nearest neighbor.
iv. Synthesize new data point in the same way as SMOTE.

Sampling

What is the ideal class distribution?

Vary the percentage of minority class from 5-95% and test its performance using J48 Decision Tree.

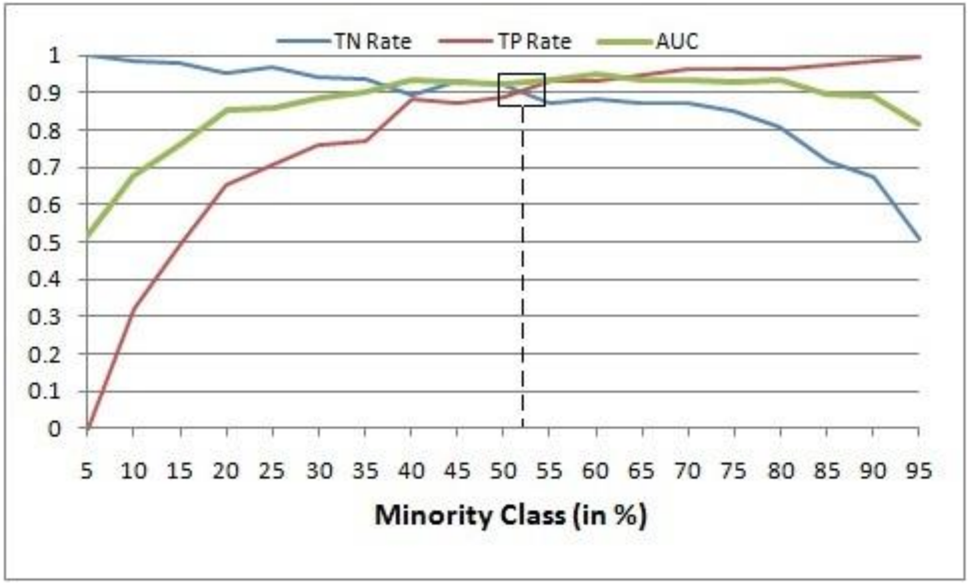


Figure 5: Effect of Class Distribution

Handling Imbalanced Class Distribution

- Sampling
- Cost Sensitive Learning

Cost Sensitive Learning

		Actual	
		Negative	Positive
Predicted	Negative	True Negative or C_{TN}	False Negative or C_{FN}
	Positive	False Positive or C_{FP}	True Positive or C_{TP}

- Assumption of classical machine learning techniques:
  "Different misclassification costs in the confusion matrix are equal."

- A CSL approach weighs the different categories of misclassification differently.

- In our domain, misclassifying a "prompt" situation as "no-prompt" is much costlier than the reverse.

Cost Sensitive Learning

Determination of different misclassification costs:

i. No cost for correct prediction, i.e. C_{TN} and C_{TP} are 0.

ii. Number of false positives is low. $\therefore C_{FP} = 1$.

iii. False negatives are critically important. We need to fine the near ideal C_{FN}.

Cost Sensitive Learning

Finding near ideal cost matrix empirically: Repeated experiments with different values of C_{FN} .

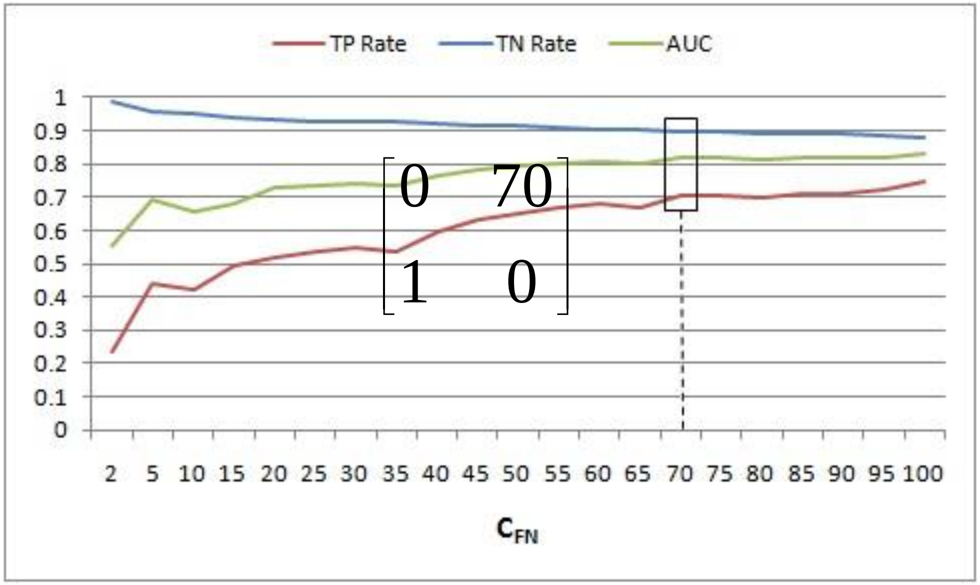


Figure 8: Effect of C_{FN} distribution

Comparative Analysis

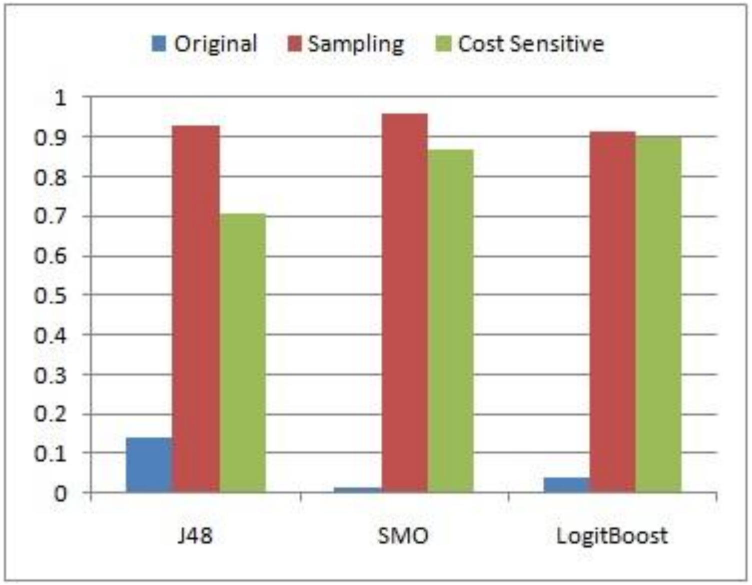


Figure 9: Comparison of TP Rate

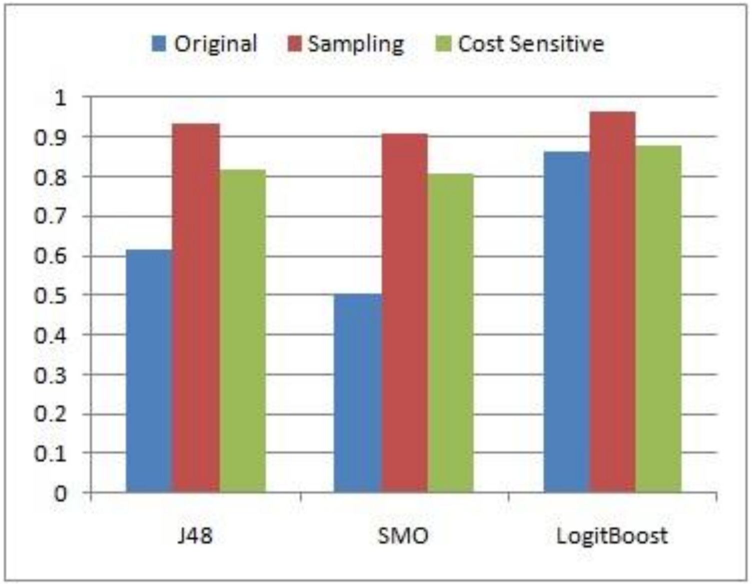


Figure 10: Comparison of AUC

Conclusion

- Description of PUCK.
- Proposed SMOTE-Variant.
- Comparative analysis with CSL.