Using Intelligent Environments for Language Model Selection

Daniel Wilson DWILSON@CS.CMU.EDU
Assistive Intelligent Environments Group, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

About 1 in 5 Americans have some kind of disability, and 1 in 10 have a severe disability (McNiel, 1997). Intelligent environments can augment lost abilities, but they depend upon speech recognition. Unfortunately, current speech recognition systems are not accurate enough for common use. A home environment was instrumented with home automation devices, motion detectors, and a speech recognition system, which logged data for approximately half a month. Three language models were generated, based upon three command categories. We compare a reinforcement learning approach and a probabilistic approach towards using information from the motion detectors to automatically select the most relevant language model. As expected, using initial data the probabilistic approach returned better results, with an error rate of 47.24%. The reinforcement learner had similar results, with an error rate of 53.14%. We look forward to reporting future results using more reasonable amounts of data.

1. Introduction

An intelligent environment uses information from sensors embedded in the environment to augment human physical and cognitive abilities. Such environments perform physical tasks such as manipulating lights, operating a telephone, or reading out loud, and cognitive tasks such as reminding or answering questions. The success of such an environment depends upon the user interface. Although the most efficient form of interpersonal communication is often speech, it is usually impractical for use in human-computer situations. Limitations in speech technology hurt the capabilities of intelligent environments (Abowd & Mynatt, 2000). Improving speech recognition accuracy is vital for the widespread use of intelligent environments.

Our speech recognition software uses a language model to improve recognition accuracy. A language model is a statistical description of which words are said, and which words are said together. Recognition accuracy increases as the language model becomes more domain specific, i.e. as probabilities increase for specific phrases. The inherent multi-modal capability of intelligent environments provides data that can be used to infer the most appropriate language model for any situation. In this paper we use information from a network of motion detectors to train our learners to associate situations with corresponding spoken commands or queries.

We introduce two learners. We have a reinforcement learner that uses an instance-based technique for state generalization. Information is collected from generalizing over past experiences and from our confidence that each language model was correct. Beliefs are updated in proportion to this feedback confidence, as are the beliefs of the k-nearest neighbors. We also have a Bayes learner. This network is implemented in the Netica Application, a Bayesian Belief Network package, developed by the Norsys company (www.norsys.com). Netica can build, learn, modify, transform and store networks. Cross-validation can be done quickly by feeding in test and training sets and retrieving detailed error analysis. The probabilistic learner outperforms the RL learner, though both show improvement over the baseline results.

2. Prior Work

Over the last several years much effort has been put into improving the construction of language models over simple tri-gram models, including higher-order n-grams, skipping (Rosenfeld, 1994; Huang et al., 1993; Ney, Essen, & Kneser, 1994], modified Kneser-Ney smoothing (Ney, Essen, & Kneser, 1994; Chen & Goodman, 1999), clustering (Ney, Essen, & Kneser, 1994; Brown et al., 1992), and combinations of these techniques (Goodman, 2000). Intelligent environments have been applied to several domains, including automatic control of utilities such as heating (Mozer, 1998), automatic control of mixed media such as DVD movies based upon user presence (Brumitt et al., 2000), facilitation of group meetings (Cohen, 1999), and accident detection (Abowd et al., 2000). However, none of these environments have been used to improve language model selection.

Though intelligent environments remain an unexplored resource for automatic language model selection, there have been past efforts to use multi-modal information to adaptively resolve language model selection. For instance, at the CMU Interactive Systems laboratories the information gained from gaze tracking has been used to

help a Janus III spoken language system decide when to swap language models (Stiefelhagen & Yang, 1997). Unfortunately, this approach is expensive in terms of equipment and computational resources.

The problem has also been approached from a theoretical standpoint. Popovici and Baggia have attempted to instantiate automatic language model selection based upon dialogue predictions (1997). Also, IBM researchers have attempted to adapt tri-gram language models to different situations based upon the observed topic of conversation (Rao, Monkowski, & Roukos, 1995). We are unable to exploit these techniques, because in our environment there is no extended dialogue.

Many researchers have noted the significance of this problem and grappled with it. In the past intelligent environments have been criticized as solutions looking for problems (Huang et al., 1999; Araya, 1995), but they offer concrete solutions to the problems faced by those with physical or cognitive disabilities. By applying the information inherent in an intelligent environment to solving the problem of automatic language model selection, we can improve speech recognition accuracy.

3. Background

For our environment we use a two bedroom, two story house with two student occupants. The environment uses speech recognition to recognize and execute certain commands received through a wireless microphone.

3.1 Intelligent Environment Setup

The house has been outfitted with an X10 home automation system (www.smarthome.com). X10 is a communications language that allows devices to communicate via existing 110V electrical wiring. Our environment uses automated wall switches and lamp switches to control lighting levels. There are also six wireless motion detectors located in six rooms: the kitchen, living room, study, hallway, bathroom, and master bedroom. The detectors are attached by velcro to the ceilings in order to maximize room coverage. The detectors require both heat and light to trigger. Once triggered the detectors pause for ten seconds before taking new readings. Motion detector readings are logged continuously. Sequentiality is not preserved, as multiple detectors may be triggered simultaneously by different people in different rooms.

For speech recognition we use the Sphinx II speech recognition system (Ward & Issar, 1994). Sphinx is a speaker independent continuous speech recognition system that manipulates our X10 system in response to verbal commands through a Shure wireless microphone (www.shure.com). The system is also connected to the Internet, and can answer questions regarding the weather, time, and date. In the living room we have an entertainment center with television, DVD player, stereo, and VCR. The environment will soon use an S-LINKE serial to infrared device to manipulate the television channels and stereo volume (www.nirvis.com).

3.2 Sphinx Speech Recognizer

Statistical language models (LM) improve the performance of speech recognition systems by providing estimates of a priori probabilities of word sequences. Sphinx uses tri-gram language models that are built automatically using LMTool (Clarkson & Rosenfeld, 1997) from a sentence corpus that contains relevant examples of what might be said. For instance, the CMU Communicator airline reservation system language model was trained using a corpus of dialogue between travel agents and customers (Rudnicky et al., 1999).

Our system also uses a modified dialogue management module originally developed by the CMU Speech group (Rudnicky et al., 1999). This dialogue manager sorts each utterance into different command categories based upon a pre-written grammar. The grammar is a collection of possible word combinations and their classifications. The following might be a partial grammar for a "turn on the lights" command/classification:

```
[Lights_On]
*please TURN on *the LIGHT *please
TURN: turn, flip, switch
LIGHT: light, lights, lamp
```

The asterix denotes an optional word. A capital word has synonyms which are later enumerated. This small grammar could understand, "Turn on light", and "Please switch on the lights," and so on. In this manner utterances are categorized and the commands or queries may be answered appropriately. These grammars must be hand-written, but the notation is flexible and powerful, and our domain limited enough to make this feasible.

Table 1. The three command groups and respective abilities.

COMMAND GROUP	EFFECT
LIGHT MANIPULATION	Turn on/off lights, dim/brighten lights
QUERY/SCHEDULE COMMAND	Set reminders for medicine, wake up times, etc., Find time, date, weather, temperature, etc.,
INFRARED COMMAND	Turn on/off television/vcr/dvd, Change channels, volume

Our grammar categorizes spoken commands pertaining to three main categories: light manipulation, infra-red commands (remote control), and query/scheduling commands. The commands are structured as natural English sentences, based upon observed interactions between our target audience and their assistants and care

givers (Wilson, 2000). The command groups are enumerated in Table 1.

A sentence corpus is needed to generate language models. There is very little data available to create such a corpus. Therefore, language models are created using corpora generated directly from the structure of our grammar, with slight modifications (Kellner, 1998). The drawback is that although there are several ways to give the same command, a user may usually only use one of them. Unfortunately every variation is equally represented in the grammar, and subsequently the language model. For this reason we chose to use our limited initial data to find probabilities for each command, and then placed certain sayings more often into the corpora. Favorite ways of giving commands therefore have a higher probability within the language model. In the future this process will be automated, so that language models are periodically re-trained online, based on the most recent user utterances.

We generated three corpora corresponding to the three command categories (i.e., lights, IR remote, & query/scheduling). By combining these three corpora in differing amounts we generate language models that expect commands from the different groups with differing probability. We created three language models, each of which has a focus on one command group, with less but equal emphasis on the two other groups. This corresponds to an expectation that the next user utterance will come from a specific command group.

4. Approach

4.1 Introduction to Learners

When the project began our feedback was the confidence that the active language model was correct. This formulates as a reinforcement learning problem, where the learner receives feedback about current behavior, but is never given the correct behavior. We recently implemented an improvement that allows offline decoding of each utterance by *every* language model. Therefore, our feedback now includes three confidences, one for each LM. In essence our task became a supervised learning problem. The dilemma is that the teacher may not be sure of the true answer. To compensate, our learners quantify trust in the feedback, and apply the feedback proportionally.

4.1.1 FEEDBACK CONFIDENCE

Each time an utterance is received from the user it is run through the active language model for a quick response. Afterwards the utterance is run off-line through *every* language model. Each language model returns a parse and a model confidence w_i. These confidences are generated independently of one another. After normalization the confidences range from 0 to 1, with 1 representing full confidence that the utterance belongs to the LM.

4.1.2 STATE CONFIGURATION

Features extracted from motion detector data and utterance information are used to define a state space $s \in S$. In this preliminary work we only extract features from motion detector data and dialog information provided by the speech recognizer. In future work we will add information from more sensors. For now the state space is represented by the following vector:

$$s = \{a_1,\ldots,a_6,b_1,\ldots,b_6,c_1,\ldots,c_6,d,e>$$

Specifically, we would like to capture transition information from motion detectors over the last minute, five minutes, and ten minutes. Sequentiality cannot be reconstructed due to simultaneous readings caused by multiple inhabitants. We improvise by keeping a count of the number of detections from each detector (in the three time intervals). These three groups of counts correspond to *a*, *b*, & *c*. The time of day (*d*) is also important, because behaviors often follow patterns that depend upon current time (i.e., mealtimes, bedtimes, playtimes, etc.,). We also include the last utterance within the last five minutes (*e*), because what the user has just said often affects what will be said next (i.e., a "tv on" command is often followed by commands to change the channels). We now describe the process in detail.

4.2 Reinforcement Learner

Given a state the learner must accurately choose which language model the next utterance will be from. The learner's choice has no impact upon the next state. Instead, the learner must choose actions that maximize immediate reward as a function of the current state. This class of problems is often called *associative* reinforcement learning (Kaelbling, Littman, & Moore, 1996). Though simpler than a full reinforcement learning problem, it is still harder than standard supervised learning. The learner only receives partial feedback; it is told whether an action was good or bad, but not which action was correct.

4.2.1 FORMULATION

The learner interacts with the environment according to a protocol where the world provides a state $s \in S$, the learner responds with an action $a \in A$, and the world returns a reward $r \in \{-1,0,1\}$. The goal of the learner is to identify a good behavior strategy $c : S \rightarrow A$.

4.2.2 TRAINING INSTANCES

A traditional definition of a training instance $< s,a,r >$ contains information about a *subset* of possible actions – specifically, those actions which have been tried before from that state. In order to partially sidestep this limitation, our training instances are formulated as:

$$< s,\{\alpha_1,\alpha_2,\alpha_3\} >$$

where,

$$\alpha_i \in \{0..1\}$$

and,

$$\sum_{i=1}^{3} \alpha_i = 1$$

In this formulation the values of α_i, which we call *action likelihoods*, represent the probability that a certain action will be selected. They correspond to light, query, and IR command groups respectively. With no information the probabilities are initialized to a prior distribution of {0.35, 0.323, 0.327}, which was generated by observing the distribution of utterances in our dataset. This formulation incites exploratory behavior because actions are chosen randomly, weighted by each α_i. When an action is chosen the world returns either a positive or negative reward, or no reward. If the reward is positive then the corresponding action likelihood is increased, or decreased if punished, otherwise it is not modified. The method of reward and punishment is described in the next section.

4.2.3 REWARD & PUNISHMENT

The learner is trained every time a user gives a command to the environment. This is very intermittent feedback, and therefore it is important to make the most of every opportunity. We reward a behavior by allocating more probability to the rewarded action likelihood. In a sense we are punishing the other behaviors in order to reward the first. We do not wish to punish behaviors that have been rewarded in the past. For this reason we use an exponential function to decide how much probability to remove from each action likelihood. This function punishes more probable action likelihoods less and smaller likelihoods more. This way our action likelihoods will 'accelerate' towards 1 or 0. The amount of change can be described by an exponential function:

$$\alpha_t = \frac{\alpha_t \cdot e^{\alpha_t}}{e^1}$$

It is also important to take into account our confidence in the feedback, w_i. Each new utterance is run through all three language models, and each returns a confidence. These three values are used to compute the overall confidence for our reward or punishment. This value is used to weight our reward or punishment function.

Reward function for α_t:

$$\theta = \alpha_t$$

Allocate,

$$\alpha_i = \frac{\alpha_i \cdot e^{\alpha_i}}{e^1} \cdot w_i \qquad \forall_{i \neq t}$$

Reward,

$$\alpha_t = \alpha_t + w_t \cdot \sum_{i \neq j} \theta - \alpha_i$$

Punishment function:

$$\theta = \alpha_t$$

Punish,

$$\alpha_t = \frac{\alpha_t \cdot e^{\alpha_t}}{e^1} \cdot w_t$$

Redistribute,

$$\alpha_i = \alpha_i + \frac{\alpha_i \cdot (\theta - \alpha_t)}{(1 - \theta)} \cdot w_i \qquad \forall_{i \neq t}$$

4.2.4 STATE GENERALIZATION

The learner needs to select the correct action even when the world provides a new, never before seen state. There are many dimensions in our state space, so this happens quite often. We generalize from previous experience using a distance weighted k-nearest neighbor algorithm. When a new state is encountered the action likelihoods are calculated as weighted combinations of the k-nearest instances. When this new state is rewarded/punished, the results are then used to update nearby instances. Note that K was chosen through cross-validation and is set to 5 for our experiments. We use a modified Euclidean distance metric to determine the distance between states. Because each dimension represents different information, we weight dimensions differently. These weights were chosen using a cross-validation approach. Future work will include a more systematic selection of these values. The values in our dimensions may take on any real value, but they are not distributed randomly. Most values are either 0 or 1, though they may range up to 60 readings. The distance function was chosen so that differences near 0 have more impact than differences around 60 readings. For instance, $d(0,1)$ is greater than $d(59,60)$. The equation is:

$$d_i = \frac{(p_i - q_i)^2}{|p_i + q_i + 1|}$$

We use the Euclidean distance for N dimensions,

$$dist = \sqrt{\sum_{i=1}^{N} d_i^2}$$

4.3 Bayesian Learner

Bayesian networks provide a probabilistic approach to inference. Bayesian reasoning assumes that our variables are describable by probability distributions, and that optimal decisions can be made by reasoning about these probabilities combined with observed data. This technique fits well with our problem because it takes advantage of all given information and provides a quantitative approach to judging the evidence supporting several hypotheses.

We used a very basic network structure in which each feature related directly to the classification. We are not interested in how the features relate to one another, only in their effect on the classification. We used cross-validation to choose a subset of the available features. Using every feature would have been computationally

intractable. Some features share mutual information, so the problem was to experiment with several combinations. Ultimately, the best features were the motion detector reading counts at five minutes and the time of day. From our earlier notation this can be described as:

$$s = \{b_1, \ldots, b_6, d\}$$

4. Results

The nature of working with and within the intelligent home environment has one unfortunate drawback, data collection. For the majority of the past few months the system has been undergoing improvements and modifications. During this time learning algorithms were tested using synthetic data. Consequently, only fifteen days of full data are available for testing and training. This corresponds to approximately 300 utterances, around twenty per day. Approximately 35% of these utterances were for light manipulation, giving us our baseline. Testing and training for both learners was done with the same jacknifed dataset. We collect more data each day, and expect to have more plausible results after at least one month of data has been collected. Results are summarized below in Table 2.

Table 2. Performance of different learners

CLASSIFIER	MEAN ERROR RATE	SAMPLE VARIANCE
RL	53.14%	7.06
BAYESIAN NET	47.24%	5.79

4.1 Reinforcement Learner

The mean error rate of the reinforcement learner was 54.14%. This is equivalent with a 19.80% relative reduction of error from our baseline.

4.2 Probabilistic Learner

Mean error rate of the Bayesian network was 47.24%, equivalent with a 27.32% relative reduction in error rate from the baseline.

5. Discussion

With such a small dataset our observations are very limited. As expected, however, the results from the reinforcement learner were worse than the Bayesian network. This is because the unsupervised learner uses the information less efficiently than the supervised learner. In the future, however, the situation may change. For instance, with the addition of more sensors to the environment (e.g., people tracking, footstep recognition, & object usage information), we will add (many) more features to the problem. Also, as the capabilities of the system grow, there will necessarily be new command groups and hence new language models. The addition of classifiers and states will be dealt with differently within each algorithm. The RL should be able to handle more features than the Bayesian network. On the other hand, as the number of classifiers grows the network will learn more about every class during each training example. As we collect more data and extend the system we intend examine these issues in an effort to create the best performing algorithm.

6. Conclusion

A significant number of people with disabilities could be helped by intelligent environments. This will require continuous speech recognition at a more accurate level than is currently available. We evaluate two learning methods that attempt to improve language model selection and therefore improve speech recognition accuracy. Our learners use information gathered from motion detectors to determine which language model to activate. We show that over time the learners select more appropriate language models based upon user location. Our results show significant improvement over the baseline, though with more data we hope to show even greater improvement.

Acknowledgements

I would like to thank Chris Atkeson for many discussions and Paul Carpenter for putting up with motion detectors and talking computers. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

Rosenfeld, R. (1996) A maximum entropy approach to adaptive statistical language modeling. Computer Speech and Language, 10:187-228, 1996.

Y. Gotoh and S. Renals. (2000) Variable word rate N-grams. IEEE, 1591-1594, 04/2000.

Ward, W. and Issar, S. (1994) Recent improvements in the CMU spoken language understanding system. In Proceedings of the ARPA Human Language Technology Workshop, March 1994, 213-216.

Rudnicky, A.I., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., Oh, A. (1999) Creating natural dialogs in the Carnegie Mellon Communicator system. Proceedings of Eurospeech, 4, 1531-1534.

Goodman, J. (2000) Putting it all together: language model combination. IEEE, 1647-1650, 04/2000.

R. Rosenfeld. (1994) Adaptive statistical language modeling: a maximum entropy approach. PhD thesis, Carnegie Mellon University, April 1994.

X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld. (1993) The SPHINX-II speech recognition system: An overview. Computer, Speech, and Language, 2:137-148.

H. Ney, U.Essen, and R. Kneser. (1994) On structuring probabilistic dependences in stochastic language modeling. Computer, Speech, and Language, 8:1-38.

P. F. Brown, V. J. DellaPietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. (1992) Class-based n-gram models of natural language. Computational Linguistics, 18(4):467-479, December 1992.

S. F. Chen and J. Goodman. (1999) An empirical study of smoothing techniques for language modeling. Computer Speech and Language, 13:359-394, October 1999.

Mozer, M. C. (1998) The neural network house: An environment that adapts to its inhabitants. In M. Coen (Ed.), Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments (pp. 110-114). Menlo, Park, CA: AAAI Press.

McNeil, J. (1997) Disabilities affect one-fifth of all Americans: Proportion could increase in coming decades. U.S. Census Brief.

Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. (2000) EasyLiving: Technologies for Intelligent Environments. Handheld and Ubiquitous Computing,

Cohen, M. (1999) The future of human-computer interaction or how I learned to stop worrying and love my intelligent room. IEEE Intelligent Systems. March/April 1999.

Stiefelhagen, R. and Yang, J. (1997) Gaze tracking for multimodal human-computer interaction. IEEE. 2617-2620.

Popovici, C. and Baggia, P. (1997) Specialized Language Models Using Dialogue Predictions. IEEE. 815-818.

Huang, A., Ling, B., Ponekanti, S., and Fox, A. (1999) Pervasive Computing: What Is It Good For? Communications of the ACM. 84-91. 5/1999.

Araya, A. (1995) Questioning Ubiquitous Computing. ACM Transactions on Computer-Human Interaction. 230-237. 5/1995.

Rao, P., Monkowski, M., and Roukos, S. (1995) Language Model Adaptation Via Minimum Discrimination Information. IEEE. 161-164. 5/1995.

Clarkson, P. and Rosenfeld, R. (1997) Statistical Language Modeling Using the CMU-Cambridge Toolkit. From Proceedings ESCA Eurospeech.

Song, F. and Croft, B. (1999) A General Language Model for Information Retreival. ACM.

Abowd, G., Atkeson, C., Bobick, A., Essa, I., MacIntyre, B., Mynatt, E., and Starner, T. (2000) Living Laboratories: The Future Computing Environments Group at the Georgia Institute of Technology. Proceedings of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, Netherlands, April 1-6, 2000.

Kellner, A. (1998) Initial Language Models for Spoken Dialogue Systems. IEEE. 6/1998

Sutton, R. (1992) Reinforcement Learning. Kluwer Academic Publishers, Boston/Dordrecht/London.

Abowd, G. and Mynatt, E. (2000) Charting Past, Present, and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, Pages 29-58, March 2000.

Wilson, D. (2000) Preliminary Survey of Murdoch Villa. Carnegie Mellon University Assistive Intelligent Environments Project.

Kaelbling, L. P. and Littman M. L. and Moore A.W. (1996) Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4:237—285.

SmartHome. Networked home products using X10 technology. http://www.smarthome.com/.