On Designing Context Sensitive Language Models for Spoken Dialog Systems

Vaibhava Goel, Ramesh Gopinath

IBM T. J. Watson Research Center, Yorktown Heights, NY.

{vgoel,rameshg}@us.ibm.com

Abstract

In this paper we describe our approach to building dialog context sensitive language models for improving the recognition performance in spoken dialog systems. These methods were developed and successfully tested in the context of a large-scale commercially deployed system that takes in over ten million calls each month. Dialog sensitive language models are typically built by clustering dialog histories into groups that elicit similar responses. A key innovation in this paper is to use an EM clustering procedure in lieu of a k-means clustering procedure that is typically used. The EM procedure results in clusters with higher log-probability which we argue leads to better recognition performance. Additionally, we empirically observe that the EM approach has much better worst case behavior than the k-means approach as it pertains to local optima.

1. Introduction

In spoken dialog systems, dialog states often use n-gram models or finite state grammars (FSG) as prior models of user input expected at those states. These language models are typically specific to the dialog state for which they are used, and are often derived based on the user interaction with that state. It has also been shown [1, 2, 3] that making the language model for a dialog state dependent on the user interaction (dialog context) leading up to that state results in improvement in recognition accuracy.

In this paper we describe our approach to building dialog context sensitive language models. These methods were developed and successfully tested in the context of a large-scale (over ten million calls each month) commericallly deployed call-routing system shown in Figure 1 and described later.

When building such dialog context dependent language models for a state, ideally we would like to build a separate one for each distinct history leading up to that state. However, except in trivial dialog systems, or with trivial definitions of dialog history, we would neither have data nor computational resources to do that. Consequently, the dialog histories are clustered into groups that elicit similar user responses [1, 3]. One language model is then built for each cluster and used for recognition.

One approach for partitioning dialog histories, tried by Bechet et.al. [1], is to build a decision tree by asking questions about the history constituents and selecting the optimal question under some objective function. The advantage of using this approach is that the resulting trees can immediately handle unseen histories, and if the questions are well designed they have very good generalization properties. However, the requirement of a question set is a limitation of this approach.

An alternative partitioning approach, proposed by Chou [4, 5] builds a tree by directly optimizing the objective function at each split. A k-means clustering like algorithm is used to derive the optimal split. This approach has previously been applied for building dialog history dependent language models [1, 3]. The main advantage of using this approach is that it is not restricted by a pre-specified question set. However, one of its disadvantages is that it does not immediately handle unseen histories, and another disadvantage is that the k-means procedure is susceptible to local optima.

To try to alleviate the local optima problem of k-means clustering, in this paper we propose use of EM clustering [6, 7] to derive the optimal dialog history partitioning tree. It still does not address the problem of handling unseen histories, but that was not a concern for us.

The rest of the paper is organized as follows. We first describe the history partitioning problem and describe the k-means and EM clustering procedures that attempt to achieve optimal partitioning. We then describe our experimental setup, followed by a description of our results and discussion.

2. K-Means and EM Clustering for Dialog History Partitioning

In this section we formally state the dialog history partitioning problem and describe the k-means and EM procedures that attempt to achieve optimal partitioning.

Let Q denote the dialog state for which we wish to build history cluster dependent n-gram language model or stochastic finite state grammar. Let $H = H_{-1}H_{-2}H_{-3}...H_{-n}$ denote dialog history of Q, as a collection of categorical variables $H_i, i = -1, .., -n$. For example, H_{-1} could be the identity of the dialog state encountered just before Q, H_{-2} could be the recognition output at state H_{-1}, etc. We shall call H_i *component* or *constituent* variables of dialog history. An instance h of H denotes a particular history for Q.

Let $\mathcal{H} = \{h_1, h_2, ..., h_m\}$ be the collection of all distinct histories of state Q that are observed in the training data set. Associated with each history $h \in \mathcal{H}$ is a set of sentences (word strings), contained in the training set, that were spoken in context of that dialog history. In practice, this sentence set could be obtained either by manually transcribing the spoken utterances or from the output of an ASR system.

Given a subset $l \subset \mathcal{H}$ of histories, the bigram model obtained from the counts of the bigrams in l is

$$f(w_2|w_1, l) = \frac{\sum_{h \in l} c(w_1 w_2|h)}{\sum_{h \in l} \sum_{w_2} c(w_1 w_2|h)} \quad (1)$$

where $c(w_1 w_2|h)$ is the count of bigram $w_1 w_2$ obtained from sentences associated with h. We shall also use $f(|l)$ to denote the bigram of Equation 1.

The log-probability of data belonging to h, under the bigram for l (Eqaution 1) is

$$L(h|f(|l)) = \sum_{w_1 w_2 \in h} c(w_1 w_2|h) log f(w_2|w_1, l), \quad (2)$$

and the total log-probability of data belonging to l is

$$L(l) = \sum_{h \in l} L(h|f(|l)). \quad (3)$$

The objective of partitioning algorithms is to find a partition P of $\mathcal{H}$ so as to maximize total log-probability

$$\sum_{l \in P} L(l) \quad (4)$$

We note that instead of using bigram counts and probabilities, we could have used unigram, trigram, sentence counts, or some other suitable model. Our choice of using bigrams is largely arbitrary, it is motivated by our intuition that this would provide a good model and alleviate data sparsity problem.

We follow greedy top-down partitioning procedures where a binary tree is built by iteratively finding optimal binary splits (with largest log-probability gain) of tree nodes.

The tree growing process is stopped when the log-probability gain from a split falls below a threshold, or when the count of sentences (associated with histories) after split falls below a threshold.

2.1. K-means Clustering

We now briefly describe our k-means procedure (similar to the ones used in [1, 3] that attempts to maximize the log-probability of a tree node split. Given a node (set of histories) l to be split in two, the procedure is

1. randomly split l into two sets lA and lB
2. build bigram models $f(|lA)$ and $f(|lB)$ (Equation 1)
3. create two new empty sets lA_{new} and lB_{new}
4. for each $h \in l$, place h in lA_{new} if $L(h|f(|lA)) \geq L(h|f(|lB))$, otherwise place h in lB_{new}.
5. if lA_{new} is same as lA stop. Otherwise assign $lA = lA_{new}$ and $lB = lB_{new}$ and go to step 2.

Smoothing

Due to the possiblity of $c(w_1 w_2|h) = 0$, it is important to smooth the history bigram counts. We consider a simple form of smoothing whereby bigram counts for $h \in l$ are augmented as

$$c_{new}(w_1 w_2|h) = c(w_1 w_2|h) + \frac{\alpha}{|l|} \sum_{h' \in l} c(w_1 w_2|h') \quad (5)$$

where α is a smoothing parameter, and $|l|$ is the size of set l.

An alternative to smoothing the history counts is to use the Gini index, as discussed in [5, 4]. We experimented with both these methods of avoiding the 0-count problem.

Initialization

To derive the initial random partition of l into lA and lB, we consider two methods. In the first method, each history belonging to l is placed in lA if the outcome of a uniform (0,1) random variable is less than 0.5, in lB otherwise.

In the second initialization method, we first obtain two bigram models, fA and fB, by randomly perturbing the bigram model $f(|l)$. Each history in l is then placed in lA if $L(h|fA) > L(h|fB)$, otherwise it is placed in lB.

We experimented with both these initialization procedures; results from these experiments are presented in Section 4.

2.2. EM Clustering

The EM clustering of dialog histories that we propose works as follows.

Given a set of histories (tree node) l to be split in two

1. randomly perturb $f(|l)$ to create two initial bigram distributions fA and fB. Initialize $LL = -Inf$.
2. create two new empty sets lA_{new} and lB_{new}. Set $LL_{new} = 0.0$
3. foreach $h \in l$, find

$$p(A|h) = \frac{e^{L(h|fA)}}{e^{L(h|fA)} + e^{L(h|fB)}} \quad (6)$$

$$p(B|h) = 1.0 - p(A|h) \quad (7)$$

$$LL_{new} = LL_{new} + log(e^{L(h|fA)} + e^{L(h|fB)}) \quad (8)$$

4. If $LL_{new} - LL <$ threshold, stop.
5. multiply bigram counts of h by $p(A|h)$ and assign to lA_{new} and multiply bigram counts of h by $p(B|h)$ and assign to lB_{new}
6. assign $fA = f(|lA_{new})$, $fB = f(|lB_{new})$, and $LL = LL_{new}$. go to Step 2.

For EM clustering algorithm also we used the smoothing procedure of Equation 5.

3. System Description

The dialog system we consider is shown in Figure 1. This is a call routing application that attempts to direct customer calls to IVR applications or customer service agents. The application starts in an "open-ended" state, labeled S1 that prompts users to state their request. This state uses an N-gram based statistical language model for recognition. User utterance is processed by an ASR engine and the recognition output is assigned a semantic label that determines the next dialog action. In Figure 1, list-choices is a semantic interpretation.

If the users have trouble using the system at S1, or if they ask the system to list their choices, it takes them to a "directed-dialog" state, labeled S2, which lists some of the choices available to the user. This state uses a stochastic finite state grammar for recognition. As with state S1, each recognition output at S2 is assigned a semantic interpretation that determines the next dialog action. There is also a "Confirmation" state that uses a finite state grammar containing various ways of saying "yes" and "no."

Some example user utterances at S2 and corresponding dialog histories (H_{-1} H_{-2} H_{-3} H_{-4}) are

more options	list-choices S1 ϕ ϕ
operator	<no-speech> S1 <no-speech> S1
my account balance	more-options S2 list-choices S1
information	<mumble> S1 <noise> S1
<no-speech>	list-choices S1 ϕ ϕ

The histories are read right-to-left; for example, in the third line above the user started at S1, asked for "list-choices," went to S2, asked for "more-options," reached S2 again, and said "my account balance." ϕ indicates a null value for H_i.

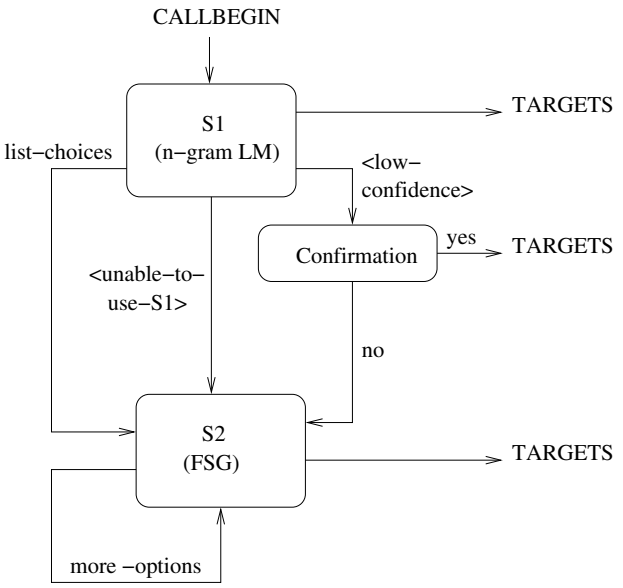

Figure 1: The call-routing dialog system considered in this paper.

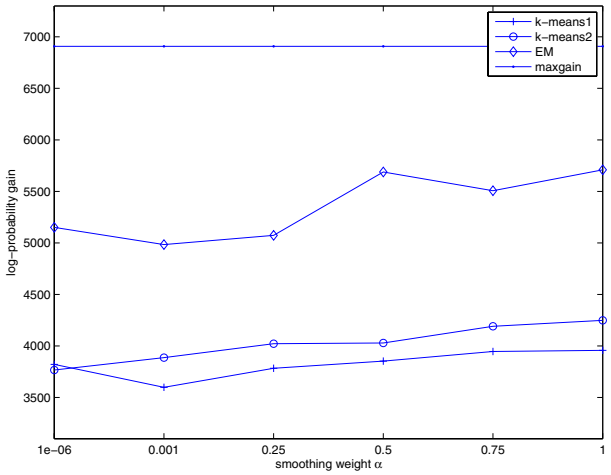

Figure 2: Log-probability gain of k-means and EM clustering, as a function of smoothing parameter α.

	mean gain	max gain	num max gain	min gain
k-means 1	3852.42	6907.76	16	1260.50
k-means 2	4028.27	6907.76	7	1688.30
EM	5688.91	6907.76	45	2271.90

Table 1: Comparion of k-means and EM clustering.

4. Experiments & Results

4.1. Data Sets

From the dialog system of Figure 1, we collected 19972 instances of user interaction with state S2. For each of these instances, we also kept a detailed record of user interaction with the dialog system leading up to state S2. The utterances that the user spoke at state S2 were manually transcribed.

The 19972 instances were divided into two sets - a training set containing 16943 instances and a test set containing the remaining 3029 instences. All our experiments are results reported here are based on these two data sets.

4.2. Selecting Dialog History Components

Our first step was to decide what components to include in the dialog history. Keeping dialog state label and semantic interpretations for all dialog states in the history, as shown in example in Section 3, resulted in 666 unique histories for our training set; an average of about 25 instances for each history. We decided this to be too sparse and chose to use only the previous state label and semantic interpretation at that state as components of the dialog history. This choice resulted in 40 unique histories with an average of about 423 interaction instances for each history.

4.3. Comparing K-means and EM Clustering

To compare effectiveness of k-means and EM in clustering dialog histories, we first measured the log-probability gain obtained in splitting the training set (root node of the tree) into two clusters.

Figure 2 shows the log-probability gain of EM clustering procedure and of two different initializations of k-means - random split based initialization labeled “k-means 1” in the figure, and model perturbation based initialization labeled “k-means 2” in the figure. Each point on this plot is obtained by taking the average of log-probabilities gains over 100 random runs. The flat line on the top marks the maximum log-probability gain achieved among all clustering runs for all smoothing parameters and initializations. This represents our approximation for the global maxima as getting the true global maxima would be computationally prohibitive.

From Figure 2 we note that on average EM clustering results in significantly higher log-probability gain as compared to either initializations of k-means clustering.

The mean value of log-probability gain for Gini index based k-means was 3231.27 which is significantly worse than the count based smoothing.

To gain further insight into how often k-means and EM achieve high log-probability gains, we looked at their 100 runs for smoothing weight of 0.5 and computed the maximum and minimum gain achieved, and number of times maximum gain was achieved. These results are presented in Table 1.

From the results in Table 1 we note that the maximum gains achieved by the k-means and EM clustering procedures are identical - therefore, if either procedure is run enough number of times with random initializations, similar results can be obtained. However, in any given run, the EM clustering appears to have a much higher chance of attaining the maximum value. Furthermore, the worst local optima that k-means procedure got caught in was significantly worse than the worst local optima that the EM clustering was caught in.

4.4. Building Trees

In the next set of experiments we built binary trees using k-means and EM clustering. To prevent the trees from getting very deep,

	mean gain	max gain	min gain
k-means 1	11935.1	12569.3	10124.8
k-means 2	11927.4	12555.8	10361.0
EM	12213.0	12586.4	11178.9

Table 2: Results of tree building experiments

a minimum log-probability gain of 400.0 and a minimum instance count of 1000 for resulting nodes was required from each split.

Table 2 presents the results of the tree building procedure using k-means and EM clustering. As with Table 1, these numbers are based on 100 tree building runs with random initializations.

From these results, the two procedures appear closer than they did based on their performance at splitting the root node (Table 1); however, the overall trends of EM having better average gain and better local optima property (the min gain is higher for EM as compared to other two procedures) are still clearly there.

4.5. Speech Recognition with Context Sensitive Grammars

As our final set of experiments we evaluated the impact of log-probability gain on recognition accuracy on the held out test set.

Ideally, to compare EM and k-means clustering trees for their recognition performance, we would like to carry out recognition with all the trees built using these procedures. However, that was not computationally feasible at the moment, hence we selected 5 trees built in Section 4.4 that had log-probability gains ranging from minimum gain of 10124.8 to maximum gain of 12586.4 (cf. Table 2). For each of these trees, context sensitive grammars were built using data from leaf nodes of these trees and at test time each utterance was recognized using the grammar appropriate for the context specified by the dialog history of that utterance.

Table 3 shows the word and sentence error rates resulting from use of these five different trees. It also shows the baseline word and sentence error rate with the non-context-sensitive grammar; i.e. grammar built with data at root node of these trees.

baseline wer/ser : 25.89/32.72	
log-prob gain	wer/ser
10124.8	24.68/30.73
10699.3	24.64/30.67
11179.0	24.45/30.60
12123.5	24.44/30.57
12495.2	24.37/30.57

Table 3: Recognition word and sentence error rates using trees with different log-probability gains

From Table 3 we first note that the use of context sensitive grammars results in a significant gain in word and sentence error rates over the baseline. Furthermore, the log-probability gain obtained in tree building procedure appears to be fairly directly correlated with the recognition accuracy, but the improvement in accuracy with increased tree log-probability is relatively small.

The observation that trees with higher log-probability result in higher recognition accuracy, combined with the observation that EM clustering on-average results in higher log-probability gains as compared to k-means and also has a much higher chance of achieving optimal log-probability, argues for use of EM as the clustering procedure of choice. This is especially true in cases where one doesn't have the liberty of trying many random initializations.

5. Conclusions & Future Work

We evaluated use of EM clustering procedure for building dialog context sensitive stochastic FSGs for a large scale commercially deployed system. In particular, we contrasted it with a previously used k-means like clustering procedure. Our experiments suggest that using EM for clustering results in partitions with higher log-probability on average, has better local optima properties, and would be expected to result in trees that have a higher recognition accuracy as compared to the k-means based clustering procedure.

All of the clustering experiments reported in this paper are carried out with manually transcribed data. One of the future directions that we are pursuing is unsupervised clustering and context sensitive model building, as that will allow us to use orders of magnitude more data and will hopefully result in more robust and more accurate models.

6. Acknowledgments

We would like to thank Jeff Kuo and Stanley Chen for help with experiments this paper.

7. References

[1] F. Bechet, G. Riccardi, and D. Hakkani-Tur, "Mining spoken dialog corpora for system evaluation and modeling," in *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain, 2004.

[2] C. Popovici, P. Baggia, P. Laface, and L. Moisa, "Automatic classification of dialogue contexts for dialogue predictions," in *Proc. ICSLP*, Barcelona, Spain, 1998.

[3] F. Wessel, A. Baader, and H. Ney, "A comparison of dialogue-state dependent language models," in *Proc. ESCA Workshop on Interactive Dialogue in Multi-Modal Systems*, Kloster Irsee, Germany, 1999.

[4] P. Chou, "Optimal partitioning for classification and regression trees," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 340–354, 1991.

[5] F. Jelinek, *Statistical methods for speech recognition*, The MIT Press, Cambridge, MA, 1999.

[6] R. Xu and D. Wunsch II, "Survey of clustering algorithms," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[7] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Computational Statistics and Data Analysis*, vol. 14, no. 3, pp. 315–332, 1992.