Probabilistic Modeling with Bayesian Networks for Automatic Speech Recognition

Geoffrey Zweig† and Stuart Russell‡
†IBM T. J. Watson Research Center
‡University of California, Berkeley
gzweig@watson.ibm.com, russell@cs.berkeley.edu

Abstract

Bayesian networks are an extremely general probabilistic modeling framework, and are increasingly being applied to complex real-world problems. In this paper, we describe the use of a Bayesian network system in large vocabulary isolated word recognition. We briefly review the algorithms and network structures used, and present results showing that significant improvements in word error rate result from modeling acoustic and articulatory context with a multivalued context variable. The network parameters are highly correlated with simply defined acoustic characteristics, and utterance clustering results in a partitioning according to both speaker gender and the presence of liquid consonants.

1 Introduction

The problem of automatic speech recognition has been extensively studied over the past several decades, culminating in recent years with the appearance of commerical products for dictation, data-entry, and automated help. Surprisingly, however, the basic machinery of speech recognition has not changed much during this period, and is facing some severe limitations.

Most current speech recognition systems are based on probabilistic modeling with hidden Markov models (HMMs). In this paradigm, speech generation is modeled as a stochastic process, and an HMM is defined in terms of two basic concepts: phonetic states and acoustic emissions. The atomic sounds of a language are categorized into a finite set of up to several thousand linguistic units or phones, and a probability distribution over sounds is associated with each state. To generate an utterance, a speaker is assumed to translate from the sequence of words he wants to say to a sequence of phones (this mapping is often deterministic), and then to proceed from state to state emitting sounds according to the associated output distributions. More specifically, at each instant in time, the speaker makes a stochastic decision whether to stay in the state he is in or to proceed to another state, and then makes a second stochastic decision about what sound to emit. Denoting the observation sequence associated with an utterance by $\mathbf{o}$, and the state sequence by $\mathbf{q}$, the probability of an utterance is given by $P(q_1)P(o_1|q_1)\prod_{t=2}^{N} P(q_t|q_{t-1})P(o_t|q_t)$. Although the meaning assigned to states varies from system to system, as does the exact method for representing $P(o_t|q_t)$, the set of concepts and the basic factorization is identical in most commercial systems.

The reason that one might want to explore probabilistic models with a richer vocabulary is that speech is in fact generated by a process that involves significantly more that a finite set of phonetic states. In particular, it is generated by an articulatory process in which the lips, tongue, jaw, and other speech articulators move in a coordinated (but not entirely predictable) way to generate sound.

Bayesian networks [9] are a powerful modeling framework that facilitates the expression and computational testing of detailed probabilistic models. Whereas an HMM associates exactly two variables with each time frame, a Bayesian network allows arbitrary sets of variables to be associated with each frame. Additionally, arbitrary factorizations of the joint probability distribution can be specified. Bayesian network algorithms exist with the same functionality as HMM algorithms, but are general enough to handle larger sets of variables with relaxed constraints on the factorization. These qualities are desireable in building more detailed probabilistic models in which not every variable (e.g. the tongue) depends on every other (e.g. glottal state).

We have implemented a system for isolated word recognition with Bayesian networks, and in previous work [14], we reported results for the PhoneBook database [11] showing relative improvements in the word error rate of between 12 and 29% with a binary auxiliary variable representing acoustic/articulatory context. This paper reviews Bayesian network algorithms and structures for speech recognition, and presents new results showing that the use of multivalued and multiple context variables results in a further improvement. Additionally, we present results in which the network is structured for unsupervised utterance clustering.

2 Bayesian Networks

2.1 Definition

A Bayesian network expresses a joint probability distribution over a set of random variables and consists of:

1. A set of random variables $X_1, \ldots X_n$.
2. A directed acyclic graph in which each variable appears once. The immediate predecessors of a variable X_i are referred to as its parents, with values $Parents(X_i)$. The joint probability distribution is factored as:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i | Parents(X_i)).$$

3. A representation of the required conditional probabilities. When the variables are discrete, a tabular representation is convenient. For real-valued acoustic observations, Gaussian mixtures can be used.

Temporal processes are modeled with a variant referred to as dynamic Bayesian networks (DBNs) [1] . In a DBN, a set of variables is associated with each frame, and the complete set of variables consists of the union of all these subsets. The graph structure is repeating, and the conditional probabilities associated with analogous variables in different frames are tied.

2.2 Algorithms

Bayesian networks are useful in ASR because there are algorithms for performing the same tasks that can be solved for HMMs, while at the same time working with more general models of probability distributions. The computations are simplest when the Bayesian network is tree-structured and contains only discrete variables. In the next section, we show how to do inference with such a network; we then sketch the extensions necessary for general network structures. Our approach is based on that of [10], and more detail can be found in [13]. Other inference algorithms can be found in, e.g. [9, 7, 5].

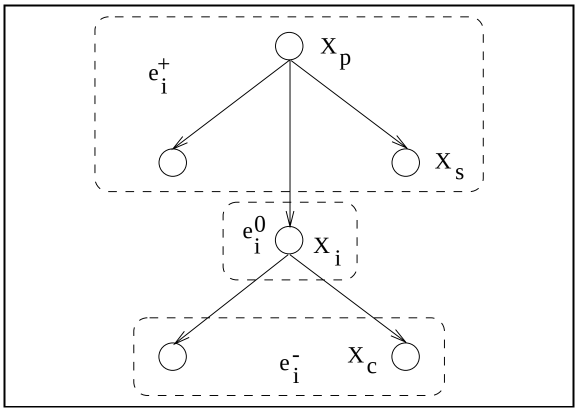

Figure 1: A tree of variables. The partitioning of the evidence is shown for X_i. Parent (X_p), child (X_c), and sibling (X_s) variables are also shown.

There is a distinction between variables with known values (observation variables) and variables whose values are unknown (hidden variables).

2.2.1 Tree Structured Networks

The observed variable values, or evidence, are partitioned into three sets for each variable X_i. These sets are: $\mathbf{e}_i^0, \mathbf{e}_i^-$, and $\mathbf{e}_i^+$. $\mathbf{e}_i^0$ is the observed value for X_i (if known; otherwise $\mathbf{e}_i^0 = \emptyset$). $\mathbf{e}_i^-$ is the set of known values for the variables which occur in subtrees rooted in X_i, and $\mathbf{e}_i^+$ is the remainder (see Figure 1). We denote the values of X_i that are consistent with e_i^0. In the case at hand, where X_i is a single variable, this means that when X_i is hidden, $CON(e_i^0)$ contains all its possible values, and when X_i is observed, $CON(e_i^0)$ contains only its observed value. In section 2.2.2, we will consider the more general case of composite variables whose values range over the Cartesian product of a set of constituent variables. In this case, $CON(e_i^0)$ refers to the set of cross-product values that are consistent with all known constituent values.

Note that the union of the evidence includes all the observations, and

$$P(e_i^+, e_i^-, e_i^0, X_i = j) = \\ P(e_i^+, X_i = j) P(e_i^-, e_i^0 | X_i = j, e_i^+) = \\ P(e_i^+, X_i = j) P(e_i^-, e_i^0 | X_i = j).$$

In the case that $X_i = j$ contradicts e_i^0, the second factor is zero.

In the inference procedure, the following two key quantities will be calculated for each variable X_i:

- $\lambda_j^i = P(\mathbf{e}_i^-, \mathbf{e}_i^0 | X_i = j)$

Algorithm **Inference**()
for each variable X_i in postorder
  if X_i is a leaf
    $\lambda^i_j = 1,\ j \in CON(e^0_i);$
    $\lambda^i_j = 0,\ otherwise.$
  else
    $\lambda^i_j =$
$\prod_{c \in children(X_i)} \sum_f \lambda^c_f * P(X_c = f|X_i = j),$
  $j \in CON(e^0_i);$
    $\lambda^i_j = 0,\ otherwise.$

for each variable X_i in preorder
  if X_i is the root
    $\pi^i_j = P(X_i = j)$
  else
    let X_p be the parent of X_i
    $\pi^i_j = \sum_{v \in CON(e^0_p)} P(X_i = j|X_p = v) * \pi^p_v *$
    $\prod_{s \in siblings(X_i)} \sum_f \lambda^s_f * P(X_s = f|X_p = v)$

Figure 2: Inference in a tree.

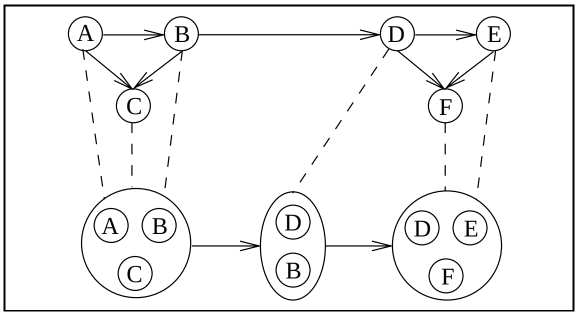

Figure 3: A non chain-structured Bayesian network (top), and a representation with composite variables (below). The new composite variables are connected in a chain structure, and conditional probabilities can be assigned so that the two Bayesian networks represent the same probability distribution (see text). The dashed lines show the composite variables to which the original variables are assigned. The members of each composite variable are indicated by the labeled interior circles.

- $\pi^i_j = P(\mathbf{e}^+_i, X_i = j)$.

It follows from these definitions that for every variable X_i,

- $P(Observations) = \sum_j \lambda^i_j * \pi^i_j$.
- $P(X_i = j|Observations) = \frac{\lambda^i_j * \pi^i_j}{\sum_j \lambda^i_j * \pi^i_j}$.

Figure 2 presents an algorithm for computing these values. Finding the likeliest assignment of values can be done by replacing the sums by maximizations, analogous to Viterbi decoding with an HMM.

2.2.2 General Graphs

Inference in non-tree-structured graphs such as that in Figure 5 is done by using a change-of-variables to convert the graph into an equivalent tree-structured one. Each new variable represents a subset of the old variables, and ranges over the Cartesian-product of its constituents. The new network will work if the following requirements are met [13]:

1. Each original variable must be found along with its parents in at least one of the new variables. Each original variable is "assigned" to one such new variable.
2. The new variables must be connected in a tree-structure such that if an original variable is found in two of the new variables, it is also present in every new variable along the path connecting the two.

Figure 3 shows an example of a non-tree structured network and a new network representation that satisfies these two conditions.

Conditional probabilities in the new representation are defined as follows. Suppose Y_i is the parent of Y_j in the new tree. Let $\mathcal{V}$ be the set of original variables assigned to Y_j. The conditional probability $P(Y_j = m|Y_i = n)$ (with m and n being cross-product values) is defined as:

- 0, if $Y_j = m$ and $Y_i = n$ imply inconsistent values for a shared original variable.
- $\prod_{V \in \mathcal{V}} P(V|Parents(V))$, with the values for V and $Parents(V)$ implied by m, if $\mathcal{V} \neq \emptyset$. (In this case, n is not used.)
- 1, otherwise.

If Y_j is the root, there is no conditioning on Y_i, and only the last two items are relevant. The evidence sets are defined to reflect what is known about the new variables' possible values, based on the observed values of their constituents.

It can be shown [?] that there is a one-to-one mapping between variable assignments that have non-zero probability in these two representations. The probability of the observations can be computed as in section 2.2.1, using Ys in place of Xs. Further, let Y_i be the new variable to which X_i is assigned. Let V^i_j be the set of Y's cross-product values corresponding to underlying variable assignments that include $X_i = j$. Then $P(X_i = j|Observations) = \sum_{w \in V^i_j} P(Y_i = w|Observations)$. Hence inferences about the original variables can be computed in terms of the composite variables.

In an efficient implementation, the running time of the inference procedures is proportional to the total number of cross-product values in the resulting tree; often, this is a relatively small number, but in the worst case it is exponential in the number of variables in the original network. With DBNs, a trivial method of constructing trees that satisfy the above requirements is to proceed as follows: create a variable $Y_{i,i+1}$ that represents all the original variables in the two time frames i and $i+1$. Also create a variable Y_i that represents the original variables from just frame i. Then connect the new variables in a simple chain: $Y_{1,2}, Y_2, Y_{2,3}, Y_3, \ldots$. This procedure is guaranteed to work for graphs with the first-order Markov property (such as Figure 5), but in general it creates many more cross-product values than are necessary. Procedures for producing more efficient trees can be found in [9, 7, 13].

2.2.3 EM

The sufficient statistics required for EM can be computed from the marginal posterior probabilities computed for the new variables. Let N_{ijk} be the number of times that variable X_i has value k and its parents are found in the jth possible configuration. In EM, θ_{ijk}, the probability that $X_i = k$ given that its parents have instantiation j, is estimated as $\frac{N_{ijk}}{\sum_k N_{ijk}}$ [6, 4]. Hence, all that is necessary is to estimate N_{ijk}. Let Y_i be the new variable to which X_i is assigned. Let $\mathbf{V}^i_{jk}$ be the set of Y_i's cross-product values corresponding to underlying variable assignments that include $X_i = j$ and $Parents(X_i) = k$. N_{ijk} can be found by summing over the appropriate values:

$$N_{ijk} = \sum_{w \in \mathbf{V}^i_{jk}} P(Y_i = w | Observations)$$

Estimating N_{ijk} from a collection of examples requires summing the individual estimates for each example.

2.3 Isolated Word ASR Networks

We begin the discussion of DBN word models by relating DBNs to HMMs. Figure 4 shows an HMM word model, and a schematic DBN representation. There are several things to note. First, the DBN is explicit about time: there is a separate set of variables for each frame. Secondly, the two diagrams must be read in very different ways: the HMM diagram represents a stochastic finite state automaton, whereas the DBN diagram represents conditional independence relations between variables. In the HMM, the nodes represent states and the arcs transitions; in the DBN, the nodes represent variables, and the arcs represent conditioning.

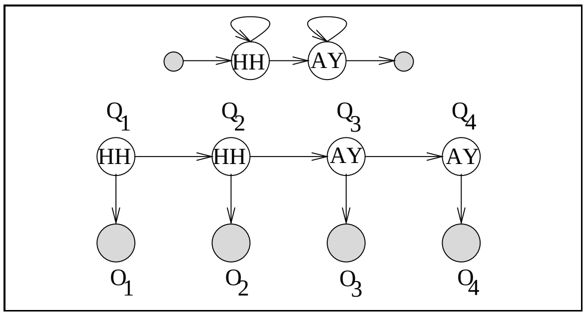

Figure 4: An HMM model of the word "hi" (top), and a conceptual DBN representation (bottom) for a four-frame utterance. Nodes represent states in the HMM, and variables in the DBN. Shaded nodes represent initial and final states in the HMM, and observed (acoustic) variables in the DBN. Arcs represent transitions in the HMM, and conditioning relationships in the DBN. The values assigned to the DBN state variables correspond to one particular path through the HMM: two time steps in /HH/, and two in /AY/. This DBN model is inadequate because it will assign nonzero probability to assignments that do not correspond to paths in the HMM, and cannot represent parameter tying (see text).

The basic idea behind the DBN representation is to create a one-to-one correspondence between assignments of values to the hidden variables, and paths through the HMM. The two representations should assign equal probabilities to analogous paths/assignments. Unfortunately, the rudimentary DBN of Figure 4 will associate nonzero probability with variable assignments that do not correspond to valid paths through the HMM (for example, when all the state variables are simply assigned the value /HH/).

The DBN of Figure 4 also does not accurately represent parameter tying. To see this, consider a left-to-right word model of the word "digit": /D IH JH IH T/. The occurrence of the /IH/-/JH/ transition requires that $P(Q_t = /JH/ \mid Q_{t-1} = /IH/) \neq 0$, whereas the occurrence of the /IH/-/T/ transition requires $P(Q_t = /JH/ \mid Q_{t-1} = /IH/) = 0$. (Otherwise, the second /IH/ could be followed by another /JH/ rather than /T/. Therefore, the two occurrences of /IH/ must be treated as different states, precluding parameter tying.

Figure 5 shows a DBN that solves the various problems associated with the simpler representation. The position variables represent the state in

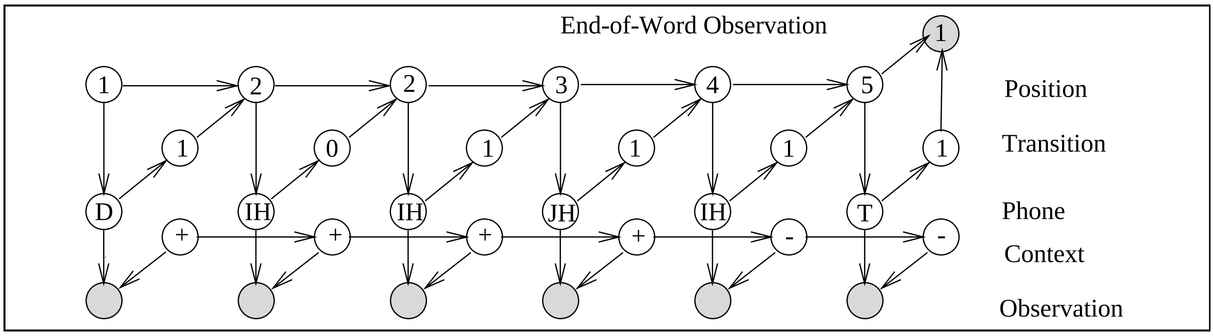

Figure 5: An improved DBN representation of an HMM, and an illustrative assignment of values to the variables. This DBN will associate a probability of 0 with hidden variable assignments that do not correspond to paths through the HMM. It also directly represents parameter tying, so for example, positions 2 and 4 will have identical behavior with respect to transition and emission probabilities because they both correspond to /IH/. The context variables are not needed to emulate an HMM, but improve performance. In this picture, they are are assigned values representative of voicing. The last /IH/ is unvoiced due to feature spreading.

an HMM word model at each time frame. The word model is assumed to be a simple left-to-right model so position i is always followed by $i+1$. (In general, arbitrary finite-state word models can be represented [13].) The phone variables represent the corresponding phone labels, and the transition variables explicitly represent when there are transitions between phones.

Figure 5 shows a representative assignment of values for the word "digit." Thus position 1 maps into /D/, position 2 into /IH/, and so forth. The probability of a transition is conditioned on the phone, thus encoding a distribution over phone durations. Depending on the value of the preceding transition variable, the position variable in a frame either retains its previous value or increases by 1. The "end-of-word" variable is assigned the arbitrary value of 1, and the conditional probabilities are defined as $P(EOW = 1|Position \neq 5 \ or \ Transition \neq 1) = 0$. This ensures that all assignments end with a transition out of the last emitting state in the word. The explicit representation of phone labels and transitions allows for parameter tying. The context variables are not required to emulate an HMM, but improve performance. With the context variable as shown, the network is similar to factorial HMMs [3].

With this representation, it is possible to assign the conditional probabilities so that there is a one-to-one correspondence between assignments of values to the DBN variables, and paths through an HMM [13]. The transition and emission probabilities are encoded in the conditional probabilities associated with the transition and observation variables. All the other conditional probabilities are either 0 or 1, and reflect deterministic relationships between the variables.

With the basic machinery required to emulate an HMM established, a variety of more interesting network structures can be tested. In particular, variables can be introduced to represent acoustic context, articulator positions, noise sources, speech rate, and other factors [13].

3 Experimental Results

This section presents results for the PhoneBook database, which is a large collection of telephone-quality isolated-word utterances chosen to exhibit coarticulatory effects [11]. The data was processed in 25ms frames overlapped by 2/3, to generate MFCCs and their deltas. Following cepstral mean subtraction, the MFCCs and deltas were vector quantized separately into two eight-bit data streams. C_0 and delta-C_0 were each quantized to four bits and concatenated to form a third eight-bit data stream.

Training, tuning and test sets are as in [2]. There were 19,421 training utterances, 7,291 tuning utterances, and 6,598 test utterances. There was no overlap between the training and testing vocabularies or speakers. The database has a vocabulary of about 8,000 words, divided into subsets of about 75 words each; the test task consisted of selecting among the word models in a single subset. Our word models were based on the context-independent phone transcriptions provided with the database.

Previous work [14] established that the use of a single binary context variable (as in Figure 5) can significantly improve performance, and Table 1 indicates that the use of multivalued context variables and multiple context chains (two per frame) further improves performance. Using two binary

States per Phone	Number of Context Variables	Ctxt Var. Arity	Total System Params	Word Error Rate
3	0 (HMM)	-	96k	5.4%
3	1	2	191k	4.1%
3	1	3	287k	4.0%
3	1	4	383k	3.8%
3	2	2	383k	3.6%
4	0 (HMM)	-	127k	4.8%
4	1	2	254k	3.6%
4	1	3	381k	3.5%
4	1	4	508k	3.2%
4	2	2	508k	3.2%

Table 1: Results for networks with one and two context variables per frame; $\sigma \approx 0.25\%$.

context chains was as good or better than using a single 4-valued context chain. The factored representation is preferable because it has only 2 independent context-transition parameters as opposed to 15.

It is not surprising that the ability to model context improved recognition performance. However, the use of a context variable differs significantly from the use of context-dependent phones: context-dependent phones encode a-priori knowledge about expected acoustics, based on the surrounding phone labels, and are insensitive to the acoustics of individual utterances. A context variable as in Figure 5 captures information about the surrounding acoustics as observed on an utterance-by-utterance basis. For example, consider simple left-to-right word models with context-dependent phones. The sequence of phones will be the same for all utterances. In contrast, a context variable can switch unpredictably between values.

Table 2 shows results using a context-dependent phonetic alphabet based on diphones (see [14]). Doubling the number of parameters by using a context-dependent alphabet produced a greater improvement than using a context variable with context-independent phones. However, the use of both kinds of context did the best (2.6% word error rate). The combination was better than a system with about the same number of parameters that simply used twice as many context-dependent phones, reducing the error rate by 16% relative to this system.

To interpret our results, we examined the correlations between the context variable and various acoustic features. The value of the context variable was most strongly related to C_0 and delta-C_0; the relationship is illustrated in Figure 6 for a single binary context variable and 4-state phone

Network	Parameters	Error Rate
CDA-HMM	257k	3.2%
CDA-Chain-BN	515k	2.6%
CDA-HMM	510k	3.1%

Table 2: Test results with a context-dependent alphabet (CDA). The first two CDA results used 336 phones; the last CDA result used less frequently occurring phones and had a size of 666. The CDA-Chain-BN has the topology of Figure 5, with a binary context variable. $\sigma \approx 0.20\%$.

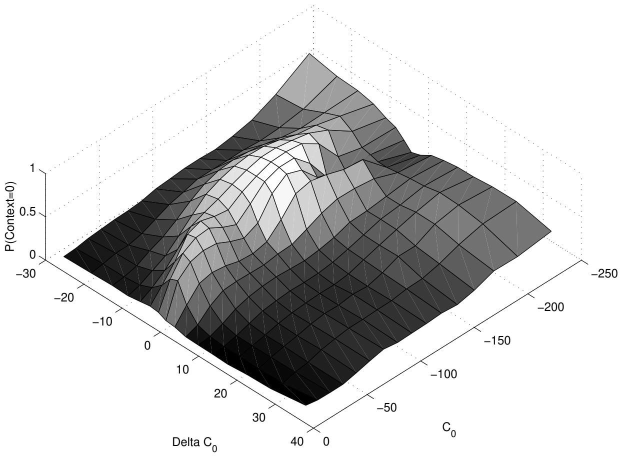

Figure 6: Association between the learned context variable and acoustic features for the network of Figure 5. Assuming that each mel-frequency filter bank contributes equally, C_0 ranges between its maximum value and about 50 decibels below maximum.

models. The context variable tends to have a value of 0 when delta-C_0 is near 0, or slightly negative. The pattern is the same for 3-state phone models, and with the context-dependent alphabet, but different for other network topologies.

3.1 Clustering Results

To illustrate the ease with which Bayesian networks can be used to perform different tasks, we configured the network of Figure 5 to do unsupervised utterance clustering. This is done by constraining the auxiliary variables to "copy" the previous value, which can be done with appropriate conditional probabilities: $P(C_t = 0/1 \mid C_{t-1} = 0/1) = 1.0$. In testing, the likeliest value of the context variable, determined by a Viterbi decoding, identifies the cluster value. A clustering network produced a word-error-rate of 4.5%, and the resulting clusters show interesting patterns with respect to both speaker and word characteristics.

The degree to which such clustering occurs can

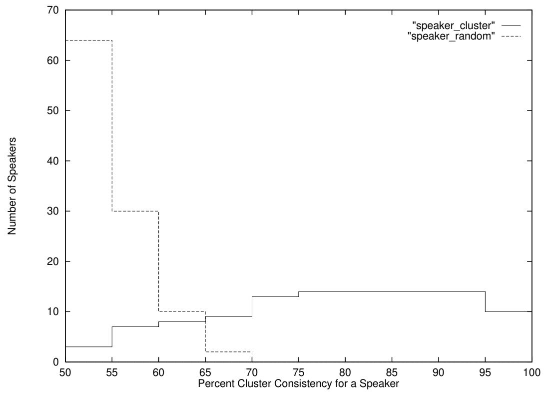

Figure 7: The frequency with which utterances from a single speaker were assigned to the same cluster. For example, about 15 speakers has their utterances clustered together with 85% consistency. On average, there are 68 utterances per speaker. The dashed line shows what would be expected at random.

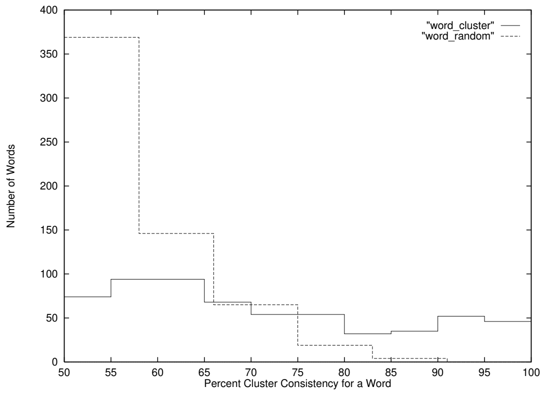

Figure 8: The frequency with which utterances of a single word were assigned to the same cluster, and what would be expected at random. On average, there are 12 occurrences of each word.

be measured by looking at the degree to which utterances with a particular characteristic are classified together in a single cluster. Figures 7 and 8 show that both speaker and word clustering are observed. Since there are more cross-validation utterances than test utterances, the histograms are based on that subset (no tuning was involved).

Figure 7 shows the consistency with which utterances from a single speaker were classified together, and what would be expected at random. Clearly, utterances from individual speakers are being grouped together with high frequency. It is to be expected that gender would be highly correlated with speaker-clustering, and in fact 75% of the female utterances were placed in cluster 0, and 82% of the male utterances in cluster 1.

Figure 8 shows the same information for particular words. The fact that the occurrences of a single word tend to be clustered together indicates that word characteristics, as well as speaker characteristics, are being modeled by the auxiliary variable.

To determine the word-characteristics associated with the two clusters, we examined the words that were very consistently assigned to a particular cluster. These are shown in Figure 3. The "female" cluster is characterized by words beginning in liquid consonants (e.g. laundromat, livelihood), while the "male" cluster is characterized by words ending in liquid consonants (e.g. pathological, unethical).

4 Conclusion

Bayesian networks are a well-principled and flexible way of representing and reasoning with probability distributions. This paper applies Bayesian networks to isolated word ASR, and presents experimental results that show that the use of an auxiliary context variable can improve recognition performance. Learned utterance clusters reflect both word and speaker dependent factors. We are currently extending the methodology to continuous speech recognition and more complicated network structures.

5 Acknowledgements

This work benefited from discussions with Jeff Bilmes, Brian Kingsbury, Su-Lin Wu, Nikki Mirghafori, Eric Fosler-Lussier, and Nir Friedman. It was funded by NSF grant IRI-9634215, and ARO grant DAAH04-96-1-0342. We are grateful to the *International Computer Science Institute* for making available the parallel computing facilities that made this work possible.

References

[1] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 524–528, St. Paul, Minnesota, 1988. American Association for Artificial Intelligence.

[2] S. Dupont, H. Bourlard, O. Deroo, V. Fontaine, and J.-M. Boite. Hybrid HMM/ANN systems for training independent tasks: Experiments on PhoneBook and related improvements. In *ICASSP-97*, pages 1767–1770. IEEE Computer Society Press, 1997.

Cluster 1					Cluster 0	
aboveboard	elsewhere	incapable	reels	squirreled	bathing	irving
mainville	melrose	oval	isabell	foghorn	landberg	laundromat
store	bale	pebbles	dimsdale	ungovernable	lifeboat	livelihood
visual	whipples	arrivals	baffled	westworld	citizend	floodgates
gospels	salesroom	scarsdale	seldom	tranquil	increasingly	motown
forced	starched	summerall	spoilt	unapproachable	negligently	plaintiff
seafowl	bakeware	bridgeforth	heartfelt	astronomical	redness	spacelink
fairchilds	gulps	geographical	dolphin	silverstone	implicitly	mcbee
mistrustful	pinwheel	unforgivable	unethical	unquestionable	engagingly	heaves
torso	quails	unusual	unawares	sparkled	honda	nape
waffles	carlson	pathological	bulls	squabble	eighths	included
unborn	untraveled	westwall			lancelet	nat
strolls	totals	allies			peanut	lindsey
beagle	cashdrawer	dialed			cupcakes	woodlawn

Table 3: The words that occurred in a particular cluster more than 90% of the time. About half the words in the first cluster end in liquid consonants (/l/ or /r/), even more if terminal /s/ is allowed. For example, "unapproachable" and "astronomical." None of the words in the second cluster end in liquid consonants. Instead, about a quarter of them *begin* with liquid consonants, e.g. "lifeboat" and "laundromat." Only one of the words in the first cluster, "reels," begins with a liquid consonant.

[3] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2/3), 1997.

[4] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995. Revised June 1996.

[5] Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 5(4):269–282, 1990.

[6] Steffen L. Lauritzen. The EM algorithm for graphical association models with missing data. Technical Report TR-91-05, Department of Statistics, Aalborg University, 1991.

[7] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B 50(2):157–224, 1988.

[8] H. Lucke. Which stochastic models allow Baum-Welch training? *IEEE Trans. on Signal Processing*, 44(11):2746–2755, 1996.

[9] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, California, 1988.

[10] Mark Peot and Ross Shachter. Fusion and propagation with multiple observations. *Artificial Intelligence*, 48(3):299–318, 1991.

[11] J. Pitrelli, C. Fong, S. Wong, J. Spitz, and H. Leung. Phonebook: A phonetically-rich isolated-word telephone-speech database. In *ICASSP-95*, pages 101–104. IEEE Computer Society Press, 1995.

[12] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. Technical Report MSR-TR-96-03, Microsoft Research, Redmond, Washington, 1996.

[13] Geoffrey Zweig. *Speech Recognition with Dynamic Bayesian Networks.* PhD thesis, University of California, Berkeley, Berkeley, California, 1998.

[14] Geoffrey Zweig and Stuart Russell. Speech recognition with dynamic Bayesian networks. In *AAAI-98*, 1998.