# GRAPHICAL MODEL REPRESENTATIONS OF WORD LATTICES

*Gang Ji,* Jeff Bilmes,* Jeff Michels,† Katrin Kirchhoff,* Chris Manning†*

*Department of Electrical Engineering
University of Washington
Seattle, WA 98915

†Department of Computer Science
Stanford University
Stanford, CA 94305

## ABSTRACT

We introduce a method for expressing word lattices within a dynamic graphical model. We describe a variety of choices for doing this, including a technique to relax the time information associated with lattice nodes in a way that trades off hypothesis expansion with presumed segmentation boundary accuracy. Our approach uses a set of time-inhomogeneous and algorithmically expressed conditional probability tables to encode the lattice. The approach was implemented as part of the graphical model toolkit, and word error rate improvements on the Switchboard corpus indicate that our technique is a viable means to incorporate large state space speech recognition systems into a graphical model.

***Index Terms—*** word lattice, graphical model, DBN, dynamic Bayesian network, dynamic graphical network, GMTK

## 1. INTRODUCTION

Large vocabulary continuous speech recognition (LVCSR) systems often require multiple recognition passes in order to be computationally viable. Often, a first pass system will produce an $N$-best list or a word lattice, both of which represent a collection of top scoring sentence hypotheses produced using a relatively simple system. These hypotheses are then re-scored by a more accurate and more computationally complex second pass system. The hope is that somewhere within the first-pass lattice lie accurate hypotheses which can then be teased out using the more complex and precise second stage. Indeed, the use of word lattices in multi-pass systems has been crucial to the success of almost all modern LVCSR systems [1, 2, 3].

There is a variety of ways of producing a complex and accurate second pass system. Many systems use more advanced or better trained language models (LMs). For example, re-scoring enables the use of advanced parse-based language models with long time dependencies, something that could result in a computational explosion if done naively in a first pass. Other systems might still use a hidden Markov model (HMM) with more advanced acoustic models, where there are many additional underlying hidden states and corresponding Gaussian mixtures and components.

One type of second-pass system that is receiving increasing attention is dynamic graphical models (such as dynamic Bayesian networks (DBNs) and/or conditional random fields) [4]. This type of model often represents a variety of explicit and intricate aspects of the speech signal (such as articulatory features [5], or various cross-speaker or multi-stream information [6]). Algorithmic efforts to enable these complex models to produce first-pass hypotheses is a worthy research goal, but we may simultaneously evaluate certain aspects of these models using a lattice to limit the state space, something that does not require the more complex inference procedures.

This work was supported by an ONR MURI grant, No. N000140510388.

In this paper, we introduce a method for representing standard word lattices as encountered in LVCSR using graphical models, and, in particular, dynamic Bayesian networks. We will see that there are a number of structural choices when representing lattices in this way. We have implemented one such choice, and we show that it is a computationally viable and representationally rich direction in which to pursue novel forms of speech recognition systems. Such a representation allows lattice scores to be easily incorporated into the vast collection of statistical models that can be quickly expressed using a graphical model. This is an important achievement as it allows complex graphical model systems to be easily used in a multi-pass speech recognition system.

## 2. WORD LATTICES AND GRAPHICAL MODELS

Both Bayesian networks and word lattices are represented using directed graphs. In this section, we give a brief review and define terminology so as to avoid their confusion.

A word lattice consists of a directed graph $\mathcal{D} = (\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ is a set of graph nodes, and $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of directed links between two nodes, so that if $(n_1, n_2) \in \mathcal{L}$ then there is a link from $n_1$ to $n_2$. Nodes typically represent time points, and we will use the notation $\tau(n)$ to indicate the time point associated with node $n$. In this paper, we assume that time points within a lattice are quantized to frame granularity, meaning that both our time variable $t$, and also $\tau(n)$, are integers in units of "time frame." Links represent words along with a number of possible scores (acoustic, language, posterior, etc.) Figure 1 shows a simple example. Without loss of generality, we focus on lattices where there is a unique starting and ending node. A lattice represents a language consisting of all word strings that can be generated by the lattice. Treating the lattice as a standard Mealy-style finite automaton, link labels are taken from an output alphabet. More information may be found in [7].

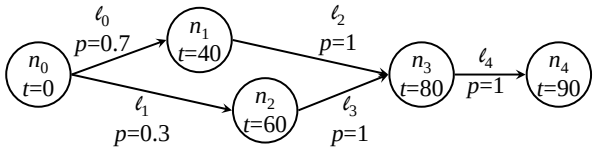


**Fig. 1**. Word lattice example

A graphical model (GM) is a graphical representation of factorization properties of families of probability distributions. When instantiated, a GM also includes local score functions such as conditional probability tables (CPTs). In particular, a DBN consists of a directed acyclic graph (DAG) $G = (V, E)$ where $V$ is a set of *vertices* (corresponding to random variables) and $E$ is a set of directed *edges*. We say that a given probability distribution $p$ factors with respect to $G$ if all factorization properties of the DBN $G$ are consistent in $p$. More details may be found in [8].

Using nodes/links for lattices and vertices/edges for graphical models allows us to refer to nodes and vertices unambiguously.

## 3. GRAPHICAL MODEL WORD LATTICES

There are several ways that the information in a lattice can be used to constrain the set of possible sentence hypotheses in a graphical-model-based decoder. In this section, we describe three possibilities.

### 3.1. Lattice sequencers without time constraints

In the simplest case, a lattice is merely a representation of a stochastic sequencer, where words may follow other words regardless of what time it is, as long as they are within one of the word sequences represented by the lattice. While a lattice node $n$ would ordinarily possess a specific time point $\tau(n)$, in this view we ignore this information and only utilize the fact that a node is a branching point in the lattice where a set of new words may stochastically follow.

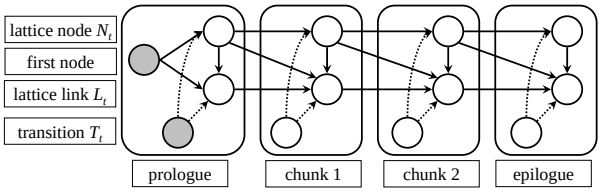


**Fig. 2**. The known-target representation

To represent this in a graphical model, we introduce two random variables (vertices) at each time frame, a vertex named "lattice node" and a vertex named "lattice link." Let $N_t$ be the lattice node vertex and $L_t$ be the lattice link vertex at time frame $t$. $N_t = i$ indicates that at time $t$ the lattice is currently at node $i$. Clearly, the two successive values $N_{t-1} = i, N_t = j$ determine a particular lattice link $L_t = \ell$ only if $(i, j) \in \mathcal{L}$ is link value $\ell$. The meaning of a time "frame" depends on the current use of the lattice. For example, when used as a representation of a constrained search space in a speech recognition system, a time frame takes on its normal meaning (e.g., 10ms acoustic frame steps). When the lattice is used for word re-scoring, however, the time frames might correspond to words (more on this below). In either case, we introduce a third binary random variable at each time point $T_t \in \{0, 1\}$ that indicates node transition. In other words, if $T_t = 1$ then the hypothesis is that at time $t$, we should move from the current lattice node to one of its next valid nodes. In this particular model, the transition nodes are redundant, since the probability of transitioning could be folded into the CPT for the lattice nodes, but we introduce binary control variables of this sort, since they are needed when using a DBN such as this as a sub-module within a more complex DBN.

There are still, however, two distinct ways of representing with these random variables the lattice node transition information. In the first case, the target (or destination) node is made available right at transition time (i.e., when $T_t = 1$). Since both the source and target are known at this point, the target node along with the newly determined link is selected and carried forward in time throughout the duration of this new link. We call this the *known-target* approach; it is depicted in Figure 2 (we use the standard GMTK-style DBN description as described in [9]). If there is a transition, $T_t = 1$, then $N_t$ is selected randomly based on a set of valid follow-on nodes starting from the node indicated by the value of $N_{t-1}$. Also, the value of $L_t$, which indicates a link, is determined based on $N_{t-1}$ and $N_t$ – thus, the new node is available right at transition time. If $T_t = 0$ then there is no transition, so we just copy previous values, i.e., $L_t = L_{t-1}$ and $N_t = N_{t-1}$ with probability one, carrying this link information along throughout the duration of the link. At the beginning of the graph, there are several extra beginning-of-sentence nodes. The *first node* vertex in the figure is always equal to the identity of the starting node in the lattice. The *transition* vertex is observed to be one so that $N_t$ is a random variable with values corresponding the second possible nodes in the lattice (e.g., $n_1$ and $n_2$ in Figure 1).

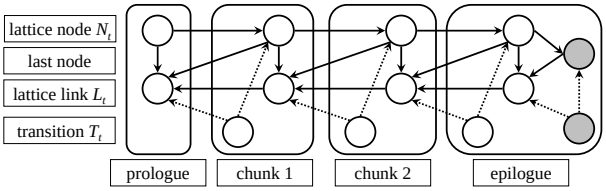


**Fig. 3**. The known-source representation

An alternative and perhaps more interesting strategy also exists. It may be that the target node is revealed only at the *next* transition time, so the current link is not known until then. This current link information made available at the next transition is carried backwards in time throughout the link duration. We call this the *known-source* approach, and it is depicted in Figure 3. Most of the vertices are the same as above. In this model, however, $N_0$ starts out with a value equal to the start node of the lattice and this is carried forward in time while there is no transition. At some time point $\bar{t} > 0$ there will be the first $T_{\bar{t}} = 1$ event at which point the next set of nodes are stochastically chosen. The previous node $N_{\bar{t}-1}$ and the new node $N_{\bar{t}}$ then determine the link $L_{\bar{t}-1}$ which is then carried back in time due to the fact that for all time less than $\bar{t}$ no transition occurs (and so link values were only copied backwards). This proceeds in an analogous way for future transitions. In this approach, additional observed vertices are required at the *end* of the graph (say time $T$). Here, a special *last node* vertex is observed with value equal to the terminal node. A special *last transition* vertex is observed to be unity which ensures that the combination of $N_{T-1}$ and *last node* are used to determine the set of valid possible final links.

We know of no theoretical advantage or disadvantage between the two approaches. It is perhaps the case, however, that the known-target approach is more intuitive since it requires only forward directed edges. Therefore, we continue our discussion only of this case.

### 3.2. Using time information

Clearly, not using time information as was done in the previous section implies a much larger set of sentence hypotheses that vary only in word segmentation times. A typical lattice, however, is not only a word sequencer, but it also indicates exact word segmentation boundaries via the node times. In this section, we describe how to incorporate such information into a graphical model.

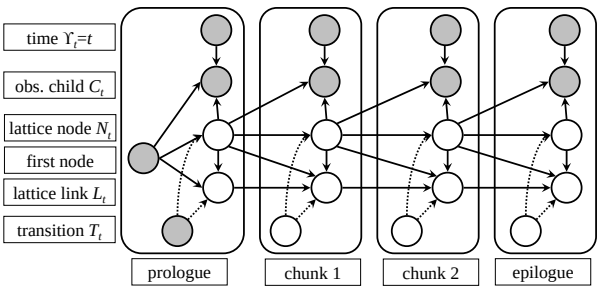


**Fig. 4**. Time-homogeneous CPTs

Our first representation utilizes only time-homogeneous CPTs in the DBN, that is, ones whose probability values do not change with

time. This DBN is shown in Figure 4 (known-target case). There are several differences from Figure 2. First, we see a new *time observation* vertex at each time frame $\Upsilon_t$, observed as $\Upsilon_t = t$ for all $t$. Second, there is an *observed child* variable $C_t = 1$ for all $t$ that is defined so as to provide a consistency constraint among its parents. The CPT $p(C_t = 1|\Upsilon_t = t, N_{t-1} = i, N_t = j)$ is created based on the lattice, defined so that the only way to explain $C_t = 1$ with non-zero probability is when it is possible in the lattice at time $t$ to move from node $i$ to node $j$. This way, the surviving hypotheses where $T_t = 1$ are the ones corresponding to valid lattice transitions, and if for any reason $T_t = 1$ at a point invalid with respect to the lattice, that hypothesis will get annihilated (i.e., is given zero probability).

We have so far seen two extreme cases in the use of the lattice's time information. Section 3.1 ignored $\tau(n)$ completely, and this section so far describes a model which forces transitions to occur at times prescribed by the model. An alternative hybrid approach allows "slack" at time transition points. Given two integers $\epsilon_1, \epsilon_2 \in \{0, 1, \ldots\}$, we may allow a word transition from node $i$ to a next possible node to occur no earlier than $\epsilon_1$ time frames earlier than $\tau(i)$, and no later than $\epsilon_2$ time frames latter than $\tau(i)$. If we set $\epsilon_1 = \epsilon_2 = 0$, we of course recover the earlier model in this section, and if $\epsilon_1 = \epsilon_2 = \infty$ we recover the model of Section 3.1. Allowing positive finite values, however, has two effects. First, it increases computation relative to $\epsilon_1 = \epsilon_2 = 0$ since additional word segmentations are allowed. Secondly, if word sequence information is reliable, but segmentation information is noisy, this allows a second pass system some leeway regarding when a word may begin and end thereby producing acoustic score variants, something that may improve accuracy (the standard evaluation tool, NIST's `sclite`, utilizes word boundary information e.g., for time-mediated scoring).

With this hybrid scheme in mind, we next describe how it is possible to avoid the extra vertices in Figure 4 if we allow the CPTs in the DBN to be time-inhomogeneous. Most CPTs in an HMM or DBN are such that they contain the same set of probabilities for all time. In other words, $p(N_t = j|N_{t-1} = i, T_t = b) = f(i, j, b)$ where the function $f(\cdot)$ does not change with time. A time-inhomogeneous CPT would allow a different set of CPT probabilities for each time point. Indeed, the approach simulates a time-inhomogeneous CPT in that the time observation $\Upsilon_t$ was used along with the CPT $p(C_t|\Upsilon_t, N_t, N_{t-1})$ that encodes everything for all time. Rather than doing this, however, we can revert back to the graph in Figure 2 if the CPT $p_t(N_t = j|N_{t-1} = i, T_t = b) = f(t, i, j, b)$ is time-inhomogeneous, i.e., is a function of time.

**Algorithm 1**: Inhomogeneous Algorithmic CPT $f(t, i, j, b)$

**if** $g(t) < \tau(i) - \epsilon_1$ **then**
  **if** $b = 1$ **then**
    return 0.0 ;
  **else**
    return $\mathbf{1}\{i = j\}$ ;
  **end**
**else if** $g(t) > \tau(i) + \epsilon_2$ **then**
  return 0.0 ;
**else**
  **if** $b = 1$ **then**
    return $p_{\mathcal{D}}(j|i)$
  **else**
    return $\mathbf{1}\{i = j\}$ ;
  **end**
**end**

This CPT can in fact be implemented algorithmically based on information contained within the lattice and the $\epsilon_1, \epsilon_2$ slack variables, as shown in Algorithm 1. In the algorithm, the *return* statement provides the final resulting probability value. The symbol $\mathbf{1}\{A\}$ is a binary $\{0, 1\}$-valued indicator variable that is 1 only if event $A$ is true. For now, $g(t) = t$ is the identity function (more on this in Section 3.3 below). We describe the algorithm via its three sections. In the first case, we have not yet reached the valid time range of the transition $t < \tau(i) - \epsilon_1$. Here, any hypothesized transition $b = 1$ is annihilated (equivalent to the observed child in Figure 4). A non-transition $b = 0$ on the other hand simply copies the node from time $t - 1$ to time $t$. In the second case, we are beyond the region of the transition $t > \tau(i) + \epsilon_2$, an event that is immediately killed off (why this occurs is described in the next paragraph).

In the third case, we are within the transition's time region. Here, if a transition is hypothesized $b = 1$ we then return $p_{\mathcal{D}}(j|i)$ which is the probability directly from the lattice of moving to node $j$ starting at node $i$. For example, in Figure 1, we have that $p_{\mathcal{D}}(n_1|n_0) = 0.7$ and $p_{\mathcal{D}}(n_2|n_0) = 0.3$. If $j$ is not a valid next node when starting at node $j$, then $p_{\mathcal{D}}(j|i) = 0$. The only tricky part of the algorithm is this third case where a non-transition $b = 0$ also induces a copy. The reason is that the lattice must force at least one transition to occur within its transition region. If a $b = 0$ event received a zero probability say at the beginning of a transition region, that hypothesis would be incorrectly annihilated since it might go on, in the following frame, to validly hypothesize a transition. The only hypotheses that should be annihilated are those that survive the entire transition region without making a transition. This may thus create events where $t > \tau(i) + \epsilon_2$, but these are then cleared up by case two above.

The logic for the matching CPT $p(L_t|L_{t-1}, N_t, N_{t-1}, T_t)$ follows similar reasoning and is omitted to save space. Obviously, the above CPTs do not normalize since we may have $\sum_j p_t(j|i, b) = 0$ for certain $i$ and $b$. Indeed, the DBN with the observed child (Figure 4) also does not globally normalize. We have, thus, effectively created an undirected graphical model with a global normalization constant that is not being computed. Of course, since this is a generative model, this constant is needed neither for maximum-likelihood training nor for decoding (a conditional model or discriminative training of this model would need at least an estimate of this constant though).

### 3.3. The utility of explicit time

The lattices above can be used for a number of purposes, including constrained automatic speech recognition (ASR) and simple lattice re-scoring, say with an improved LM. This latter application is how we have so far tested this approach (Section 4). In this case, however, there is no need to keep successive time frames at the granularity typical in ASR – in such cases, whenever there is no transition, the graph essentially would make successive identical copies of the DBN state at each time point, significantly slowing down performance. The only time "important" events occur are at the time points at which the lattice allows a transition.

We can exploit this observation by including an additional observed time parent vertex. These time parents indicate the succession of not-necessarily contiguous time frames at which the true lattice events occur. This can be represented by a function $g(t)$, where $g(t)$ is the time frame at which the $t^{th}$ real event in the lattice occurs. This information can significantly reduce the state space of the DBN. Therefore, we have a new variable $\Upsilon_t = g(t)$ for all $t$ as shown in Figure 5. Fortunately, in Algorithm 1, we have already expressed this part of the logic via function $g(t)$ with its new meaning here rather than the identity as described in Section 3.2. In practice, rather than incorporating the function $g(t)$ into Algorithm 1 as a mapping for each lattice, we instead pass only the ordinate values

of $g(t)$ directly into the algorithm — this approach has some advantages since values need only be computed once, yielding further speedups when the lattice is re-used many times. Indeed, we have found overall that our shrinking time approach improves decoding time by more than a factor of 100 for the lattices we tested.

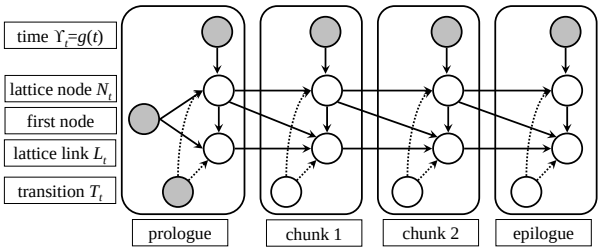


**Fig. 5**. Time observations indicate frames of real lattice events

## 4. EXPERIMENTS

To demonstrate our methodology's feasibility, we evaluate using a lattice re-scoring framework word lattices that had been generated, under a bi-gram LM, by the SRI International 5x decipher system [10] for use in the EARS continuous telephone speech (CTS) recognition system.[1] We used eval1998 + eval2000 as the development set (to adjust LM weights) and eval2001 as a test set.

We implemented graphical model lattices in the Graphical Model Toolkit (GMTK) [9] by creating new conditional probability tables objects "lattice node CPT" and "lattice link CPT". These CPTs get their information directly from HTK SLF files [11] (i.e., the CPTs themselves read the lattice information directly from the lattice files, thus allowing us to easily utilize lattices that have been produced by other speech recognition systems).

In the long run, we wish to use GMTK for a variety of novel LVCSR tasks, including re-scoring using multi-speaker LMs [6]. Therefore, our lattices are unique in that they were strung together to represent one side of an *entire conversation*. Thus, the lattices correspond on average to 5 minutes of speech! As far as we know, such long lattices have never been re-scored before – it is much more typical to re-score lattices corresponding to an individual utterance.

We must first, of course, augment the graphical models to include a LM. There are standard graphical representations of $n$-gram LMs [4]. The full graphical model we used for tri-gram re-scoring is shown in Figure 6 where the bottom portion implements the word tri-gram. Here, a transition causes a new word to be generated both based on the LM and based on the lattice. The LM is a standard DARPA back-off model, also supported in GMTK. A consistency variable is then used to make sure that the re-scored word is identical to the corresponding next lattice word (all other cases are annihilated). A bi-gram (or four-gram) lattice model can also easily be expressed, where the re-scored word depends on previous one (or three) words rather than only the previous two. In our experiments, the lattice CPT itself is set up to utilize the acoustic scores contained in the lattice, but to ignore the associated lattice LM scores since new LM scores are obtained directly from the new LM.

For our baseline result, we computed the best scoring lattice hypothesis using both the acoustic and LM scores within the lattice. Next we utilized our own bi- and tri-gram models (trained using much more data than that used for the bi-gram within the lattices). The model used was the initial model of Section 3.2, without slack variables. Results are shown in Table 1 for the test set, where all weights were optimized on the development set. As can be seen,

[1]Thanks to Arindam Mandal for generating the lattices.

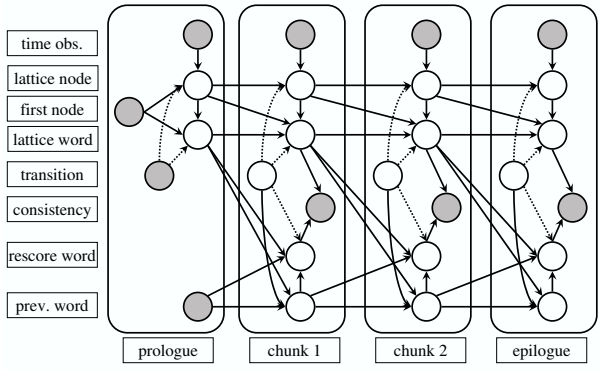


**Fig. 6**. Word lattices re-scoring with a tri-gram language model.

there is appreciable test-set improvement in word-error rate (WER) over the baseline. Decoding time, moreover, was comparable to standard lattice re-scoring systems which means that our technique is viable to add a lattice-based pruned hypothesis space to a variety of speech recognition systems expressed by a graphical model. In fact, when we attempted to re-score a 5-minute long lattice using the SRI `lattice-tool`, it ran out of memory, but GMTK with its algorithmic CPT was able to do a complete re-scoring in only 1.7 seconds, both on a 3.2 GHz 8GB Pentium-4 IA-32 class machine. Moreover, since we are utilizing a graphical model, our infrastructure allows much more than just re-scoring the lattice, or its utilization in a second-pass system. Rather, our representation allows lattice scores to be incorporated into the vast collection of statistical models expressible by a graphical model.

**Table 1**. Word lattice re-scoring word error rates

| model | WER | rel. impr. |
|---|---|---|
| baseline | 28.5% | - |
| bigram | 27.5% | 3.5% |
| trigram | 26.0% | 8.8% |

## 5. REFERENCES


[1] S. Matsoukas et. al., "The 2004 BBN 1xRT recognition systems for English broadcast news and conversational telephone speech," in *Proc. European Conf. on Speech Comm. and Tech.*, 2005, vol. 11.

[2] H. Soltau et. al., "The IBM 2004 conversational telephony system for rich transcription," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, March 2005.

[3] D.-Y. Kim et. al., "Development of the CU-HTK 2004 broadcast news transcription systems," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2005.

[4] J. Bilmes and C. Bartels, "Graphical model architectures for speech recognition," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 89–100, September 2005.

[5] K. Livescu, J. Glass, and J. Bilmes, "Hidden feature models for speech recognition using dynamic Bayesian networks," in *Proc. European Conf. on Speech Comm. and Tech.*, Geneva, Switzerland, 2003, vol. 7.

[6] G. Ji and J. Bilmes, "Multi-speaker language modeling," in *HLT/NAACL*, Boston, MA, May 2004.

[7] S. Young, "A review of large-vocabulary continuous-speech recognition," *IEEE Signal Processing Magazine*, pp. 45–57, 1996.

[8] S. L. Lauritzen, *Graphical Models*, Oxford Science Publications, 1996.

[9] J. Bilmes and C. Bartels, "On triangulating dynamic graphical models," in *Uncertainty in Artificial Intelligence: Proc. of the 19$^{th}$ Conf. (UAI-2003)*. 2003, pp. 47–56, Morgan Kaufmann Publishers.

[10] A. Stolcke et. al, "Development of the 2004 SRI/ICSI/UW speech-to-text system," in *Proc. DARPA 2004 Rich Transcr. Workshop*, 2004.

[11] S. Young et. al, *The HTK Book*, 2002.