SCARF: A Segmental CRF Speech Recognition System

Geoffrey Zweig and Patrick Nguyen
{gzweig,panguyen}@microsoft.com

April 2009

Technical Report
MSR-TR-2009-54

We propose a theoretical framework for doing speech recognition with segmental conditional random fields, and describe the implemenation of a toolkit for experimenting with these models. This framework allows users to easily incorporate multiple detector streams into a discriminatively trained direct model for large vocabulary continuous speech recognition. The detector streams can operate at multiple scales (frame, phone, multi-phone, syllable or word) and are combined at the word level in the CRF training and decoding processes. A key aspect of our approach is that features are defined *at the word level*, and can thus identify long span phenomena such as the edit distance between an observed and expected sequence of detection events. Further, a wide variety of features are automatically constructed from atomic detector streams, allowing the user to focus on the creation of informative detectors. Generalization to unseen words is possible through the use of decomposable consistency features [1, 2], and our framework allows for the joint or separate training of the acoustic and language models.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

1 Introduction

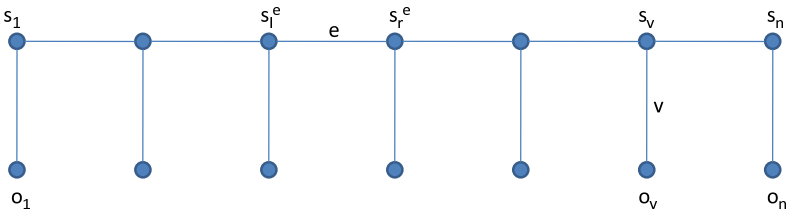

Figure 1: Graphical representation of a CRF.

The SCARF system uses Segmental Conditional Random Fields - also known as Semi-Markov Random Fields [3] or SCRFs - as a theoretical underpinning. To explain these, we begin with the standard Conditional Random Field model [4], as illustrated in Figure 1. Associated with each vertical edge v are one or more feature functions $f_k(s_v, o_v)$ relating the state variable to the associated observation. Associated with each horizontal edge e are one or more feature functions $g_d(s_l^e, s_r^e)$ defined on adjacent left and right states. (We use s_l^e and s_r^e to denote the left and right states associated with an edge e.) The set of functions (indexed by k and d) is fixed across segments. A set of trainable parameters λ_k and ρ_d are also present in the model. The conditional probability of the state sequence $\mathbf{s}$ given the observations $\mathbf{o}$ is given by

$$P(\mathbf{s}|\mathbf{o}) = \frac{\exp(\sum_{v,k} \lambda_k f_k(s_v, o_v) + \sum_{d,e} \rho_d g_d(s_l^e, s_r^e))}{\sum_{\mathbf{s}'} \exp(\sum_{v,k} \lambda_k f_k(s'_v, o_v) + \sum_{d,e} \rho_d g_d(s_l'^e, s_r'^e))}$$

In speech recognition applications, the labels of interest - words - span multiple observation vectors, and the exact labeling of each observation is unknown. Hidden CRFs (HCRFs) [5] address this issue by summing over all labelings consistent with a known or hypothesized word sequence. However, in the recursions presented in [5], the Markov property is applied at the individual state level, with the result that segmental properties are not modeled. This has some disadvantages, in that there is an inherent mismatch between the scale of the labels of interest (words) and the scale of the observations (100 per second). More generally, graphical models such as Dynamic Bayesian Networks, and CRFs, that assign a word label to every frame [6, 7] suffer a number of problems:

- The conceptual linkage between a symbol (word) and observation (100ms cepstral vector) is weak, and in fact the structure is just an undesired side-effect of the model formalism

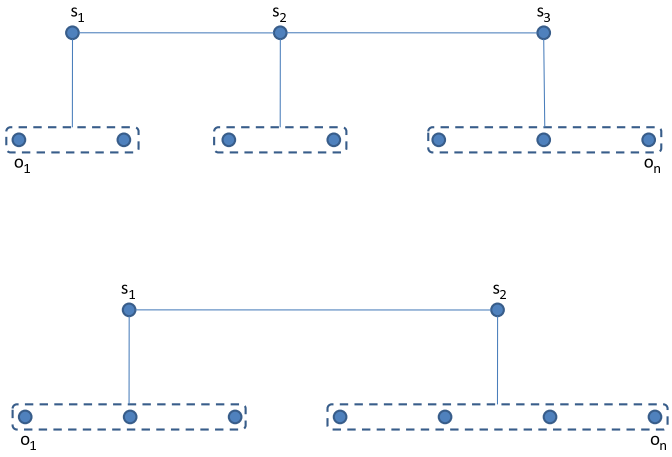

Figure 2: A Segmental CRF and two different segmentations.

- The transition functions or probabilities defined at the word level are out-of-sync at the observation level (word values only change after tens or hundreds of observations)
- The mechanisms [6, 7] which one can set up to compensate for the aforementioned transition problems are elaborate and complex

Segmental CRFs avoid this scale-mismatch. In contrast to a CRF, the structure of the model is not fixed a-priori. Instead, with N observations, all possible state chains of length $l < N$ are considered, with the observations segmented into l chunks in all possible ways. Figure 2 illustrates this. The top part of this figure shows seven observations broken into three segments, while the bottom part shows the same observations partitioned into two segments. For a given segmentation, feature functions f_k and g_d are defined as with standard CRFs. Because of the segmental nature of the model, transitions only occur at logical points (when the state changes), and it is clear what span of observations to use to model a given symbol. To denote a block of original observations, we will use o_i^j to refer to observations i through j inclusive.

Since the g functions already involve pairs of states, it is no more computationally expensive to expand the f functions to include pairs of states as well, as illustrated in Figure 3. This can be useful, for example, in speech recognition where the state labels represent words, to model coarticulatory effects where the relationship between a word and its acoustic realization may be dependent on the preceding word. Effects involving *both* left and right state context are, however, inherently more computational complex to model, and not supported.

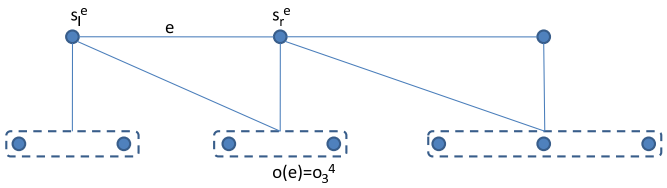

Figure 3: Incorporating last-state information in a SCRF.

(Even so, right-context can be implicitly modeled by allowing the f features to examine the *observations* in the following segment.) This structure has the further benefit of allowing us to drop the distinction between g and f functions.

In the semi-CRF work of [3], the segmentation of the training data is known. However, in speech recognition applications, we cannot assume this is so: to train, we are given a word sequence and an audio file, but no segmentation of the audio. Therefore, in computing sequence likelihood, we must consider all segmentations consistent with the state (word) sequence $\mathbf{s}$, i.e. for which the number of segments equals the length of the state sequence.

Denote by $\mathbf{q}$ a segmentation of the observation sequences, for example that of Fig. 3 where $|\mathbf{q}| = 3$. The segmentation induces a set of edges between the states, referred to below as $e \in \mathbf{q}$. One such edge is labeled e in Fig. 3. Further, for any given edge e, let $o(e)$ be the segment associated with the right-hand state s_r^e, as illustrated in Fig. 3. The segment $o(e)$ will span a block of observations from some start time to some endtime, o_{st}^{et}; in Fig, 3, $o(e)$ is identical to the block o_3^4. With this notation, we represent all functions as $f_k(s_l^e, s_r^e, o(e))$ where $o(e)$ are the observations associated with the segment of the right-hand state of the edge. The conditional probability of a state (word) sequence $\mathbf{s}$ given an observation sequence $\mathbf{o}$ for a SCRF is then given by

$$P(\mathbf{s}|\mathbf{o}) = \frac{\sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}|} \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l^e, s_r^e, o(e)))}{\sum_{\mathbf{s}'} \sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}'|} \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l'^e, s_r'^e, o(e)))}$$

1.1 Gradient Computation

In the SCARF system, SCRFs are trained with gradient descent. Taking the derivative of $\mathcal{L} = \log P(\mathbf{s}|\mathbf{o})$ with respect to λ_k we obtain:

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \frac{\sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}|} (\sum_{e\in\mathbf{q}} f_k(s_l^e, s_r^e, o(e))) \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l^e, s_r^e, o(e)))}{\sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}|} \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l^e, s_r^e, o(e)))}$$
$$- \frac{\sum_{\mathbf{s}'} \sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}'|} (\sum_{e\in\mathbf{q}} f_k(s_l'^e, s_r'^e, o(e))) \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l'^e, s_r'^e, o(e)))}{\sum_{\mathbf{s}'} \sum_{\mathbf{q} \text{ s.t. } |\mathbf{q}|=|\mathbf{s}'|} \exp(\sum_{e\in\mathbf{q},k} \lambda_k f_k(s_l'^e, s_r'^e, o(e)))}$$

This derivative can be computed efficiently with dynamic programming, using the recursions described in Section 3.

2 Adaptations to the Speech Recognition Task

Specific to the speech recognition task, we represent the allowable state transitions with an ARPA language model. There is a state for each $1 \dots n-1$ gram word sequence in the language model, and transitions are added between states as allowed by the language model. Thus, from the state corresponding to "the dog," a transition to "dog barked" would be present in a trigram language model containing the trigram "the dog barked." A transition to the lower-order state "dog" would also be present to allow for bigram sequences such as "dog nipped" that may not be present as suffixes of trigrams. Note that any word sequence is possible, due to the presence of backoff arcs, ultimately to the null-history state, in the model. In SCARF, one set of transition functions simply returns the appropriate transition probability (possibly including backoff) from the language model:

$$f^e_{LM}(s^e_l, s^e_r, \cdot) = LM(s^e_l, s^e_r),$$

independent of observations.

While one of the advantages of the SCRF method is the natural ability to jointly train the language and acoustic models in a discriminative way, it is often convenient to keep them separate. Thus, once an acoustic model is trained (the observation feature function λs), one is able to swap in different language models as necessary for particular tasks. To support this operation, we provide the ability to train a single λ to apply to all language model features. When convenient, we will refer to this distinguished parameter as ρ. The training process then learns a weight (ρ) generally appropriate to the language model, and the acoustic λs are learned in this context. To swap in a different language model, one simply needs to specify a new ARPA file, and possibly fine-tune ρ on a development set.

In the segmental framework, it is in general necessary to consider the possible existence of a segment between any pair of observations. Further, in the computations, one must consider labeling each possible segment with each possible label. Thus, the runtime is quadratic in the number of detection events, and linear in the vocabulary. Since vocabulary sizes can easily exceed $100,000$ words and event sequences in the 100s are common, the computation is excessive unless constrained in some way. To implement this constraint, we provide a function $start(t)$ which returns the set of words likely to begin at event t. The words are returned along with hypothesized end times. A default implementation of $start(t)$ is built in, which reads a set of possible word spans from a file, e.g. generated by a standard speech recognizer.

3 Computation with SCRFs

3.1 Forward Backward Recursions

The recursions make use of the following data structures and functions.

1. An ARPA n-gram backoff language model. This has a null history state (from which unigrams emanate) as well as states signifying up to $n-1$ word histories. Note that after consuming a word, the new language model state implies the word. We consider the language model to have a start state - that associated with the ngram $< s >$ - and a set of final states $\mathcal{F}$ - consisting of the ngram states ending in $< /s >$. Note that "being in" a state s implies the last word that was decoded, which can be recovered through the application of a function $w(s)$.

2. $start(t)$, which is a function that returns a set of words likely to start at observation t, along with their endtimes.

3. $succ(s, w)$ delivers the language model state that results from seeing word w in state s.

4. $features(s, s', st, et)$ returns a set of feature indices $\mathcal{K}$ and the corresponding feature values $f_k(s, s', o_{st}^{et})$. Only features with non-zero values are returned, resulting in a sparse representation. The return values are automatically cached so that calls in the backward computation do not incur the cost of recomputation.

Let Q_i^j represent the set of possible segmentations of the observations from time i to j. Let S_a^b represent the set of state sequences starting with a *successor* to state a and ending in state b. We define $\alpha(i, s)$ as

$$\alpha(i,s) = \sum_{\mathbf{s} \in S^s_{startstate}} \sum_{\mathbf{q} \in Q_1^i \mathrm{s.t.} |\mathbf{q}|=|\mathbf{s}|} \exp(\sum_{e \in \mathbf{q}, k} \lambda_k f_k(s_l^e, s_r^e, o(e)))$$

We define $\beta(i, s)$ as

$$\beta(i,s) = \sum_{\mathbf{s} \in S^{stopstate}_s} \sum_{\mathbf{q} \in Q_{i+1}^N \mathrm{s.t.} |\mathbf{q}|=|\mathbf{s}|} \exp(\sum_{e \in \mathbf{q}, k} \lambda_k f_k(s_l^e, s_r^e, o(e)))$$

The following pseudocode outlines the efficient computation of the α and β quantities. For efficiency and convenience, the implementation of the recursions can be organized around the existence of the $start(t)$ function. All α and β quantities are set to 0 when first referenced.

Alpha Recursion:
$pred(s, x) = \emptyset \ \forall s, x$
$\alpha(0, startstate) = 1$
$\alpha(0, s) = 0, \ s \neq startstate$
for $i = 0 \ldots N-1$
  foreach s s.t. $\alpha(i, s) \neq 0$
    foreach $(w, et) \in start(i+1)$
      $ns = succ(s, w)$
      $\mathcal{K} = features(s, ns, i+1, et)$
      $\alpha(et, ns) += \alpha(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(s, ns, o^{et}_{i+1}))$
      $pred(ns, et) = pred(ns, et) \cup (s, i)$

Beta Recursion:
$\beta(N, s) = 1, \ s \in \mathcal{F}$
$\beta(N, s) = 0, \ s \notin \mathcal{F}$
for $i = N \ldots 1$
  foreach s s.t. $\beta(i, s) \neq 0$
    foreach $(ps, st) \in pred(s, i)$
      $\mathcal{K} = features(ps, s, st+1, i)$
      $beta(st, ps) += beta(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(ps, s, o^{i}_{st+1}))$

3.2 Gradient Computation

Let L be the constraints encoded in the $start()$ function with which the recursions are executed. For each utterance u we compute:

$Z^L(u) = \sum_{s \in \mathcal{F}} \alpha(N, s) = \beta(0, startstate)$
for $i = N \ldots 1$
  foreach s s.t. $\beta(i, s) \neq 0$
    foreach $(ps, st) \in pred(s, i)$
      $\mathcal{K} = features(ps, s, st+1, i)$
      $F^L_{k \in \mathcal{K}}(u) += \frac{f_k(ps, s, o^i_{st+1}) \alpha(st, ps) \beta(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(ps, s, o^i_{st+1}))}{Z^L(u)}$

We compute this once with constraints corresponding to the correct words to obtain $F^{cw}_k(u)$. This is implemented by constraining the words returned by $start(t)$ to those starting at time t in a forced alignment of the transcription. We then compute this without constraints, i.e. with $start(t)$ allowed to return any word, to obtain $F^{aw}_k(u)$. The gradient is given by:

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_u (F^{cw}_k(u) - F^{aw}_k(u))$$

3.3 Decoding

Decoding proceeds exactly as with the alpha recursion, with sums replaced by maxs:

$pred(s, x) = \emptyset \ \forall s, x$
$\alpha(0, startstate) = 1$
$\alpha(0, s) = 0, \ s \neq startstate$
for $i = 0 \ldots N - 1$
  foreach s s.t. $\alpha(i, s) \neq 0$
    foreach $(w, et) \in start(i + 1)$
      $ns = succ(s, w)$
      $\mathcal{K} = features(s, ns, i + 1, et)$
      if $\alpha(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(s, ns, o_{i+1}^{et})) > \alpha(et, ns)$ then
        $\alpha(et, ns) = \alpha(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(s, ns, o_{i+1}^{et}))$
        $pred(ns, et) = (s, i)$

Once the forward recursion is complete, the predecessor array contains the "backpointers" necessary to recover the optimal segmentation and its labeling.

4 Feature Construction

SCARF is designed to make it easy to test the effectiveness of detector-type features. To enable this, it allows the user to specify multiple streams of detector outputs, from which features are automatically derived. Additional prior information may be injected into the system by providing dictionaries that specify what units are to be expected in words. This process is now described in detail; first we describe the inputs which are available in the feature generation process and then we describe how the features are automatically generated.

4.1 Inputs

4.1.1 Atomic Feature Streams

An atomic detector stream provides a raw sequence of detector events. For example, a phoneme detector might form the basis of a detector stream. Multiple steams are supported, for example, a "fricative detection" stream could complement a phone detection stream. Each stream defines its own unique unit set, and these are not shared across streams.

The format of an atomic detector stream is:
stream-name stream
(unit time)+
The first column specifies the unit name. The second specifies the time at which the unit is detected. It is used to synchronize between multiple feature streams, and to provide candidate word boundaries.

4.1.2 Unit Dictionaries

A dictionary providing canonical word pronunciations is provided for each feature stream. For example, phonetic and syllabic dictionaries could be provided. As discussed below, the existence of a dictionary enables the automatic construction of certain consistency features that indicate (in)consistency between a sequence of detected units and those expected given a word hypothesis.

The format of a dictionary is:
stream-name dictionary
(word unit+)+

4.1.3 Language Model

An ARPA format language model must be provided as an input to both the training and decoding processes.

4.2 Feature Creation

SCARF has the ability to automatically create a number of different features, controlled by the command line. Each type can be automatically generated for every atomic feature stream.

4.2.1 Ngram Existence Features

Recall that a language model state s implies the identity of the last word that was decoded: $w(s)$. Existence features are of the form:

$$f_u(s, s', o_{st}^{et}) = \delta(w(s') = r)\delta(u \in span(st, et))$$

They simply indicate whether a unit exists with a word's span. No dictionary is necessary for these, however, no generalization is possible across words. Higher order existence features, defined on the existence of *ngrams* of detector units, can be automatically constructed and used via a command line option. Since the total number of existence features is the number of words times the number of units, we must constrain the creation of such features in some way. Therefore, we create an existence feature in two circumstances only:

1. when a word and ngram of units exists together in a dictionary
2. when a word exists in a transcription file, and a unit exists in a corresponding detector file (regardless of position)

4.2.2 Ngram Expectation Features

Denote the pronunciation of a word in terms of atomic units as $pron(w)$. Expectation features are of the form:

$$f_u(s, s', o_{st}^{et}) = \delta(u \in pron(w(s'))\delta(u \in span(st, et)) \quad \text{correct accept}$$

and

$$f_u(s, s', o_{st}^{et}) = \delta(u \in pron(w(s'))\delta(u \notin span(st, et)) \text{ false reject}$$

and

$$f_u(s, s', o_{st}^{et}) = \delta(u \notin pron(w(s'))\delta(u \in span(st, et)) \text{ false accept}$$

These are indicators of consistency between the units expected given a word ($pron(w)$), and those that are actually in the specified observation span. There is one of these features for each unit and they are *independent* of word identity. Therefore these features provide important generalization ability. Even if a particular word is not seen in the training data, or if a new word is added to the dictionary, they are still well defined, and the λs previously learned can still be used. To measure higher-order levels of consistency, bigrams and trigrams of the atomic detector units can be automatically generated via command line option. The pronunciations in the corresponding dictionary are automatically expanded to the correct n-gram level. Thus, the user only needs to produce atomic detector streams.

The case where a word has multiple pronunciations requires special attention. In this case,

- A correct accept is triggered if *any* pronunciation contains an observed unit sequence.
- A false accept is triggered if *no* pronunciation contains an observed unit sequence.
- A false reject is triggered if all pronunciations contain a unit sequence, and it is not present in the detector stream.

4.2.3 Levenshtein Features

Levenshtein features are the strongest way of measuring the consistency between expected and observed detections, given a word. To construct these, we compute the edit distance between the units present in a segment and the units in the pronunciation(s) of a word. We then create the following features:

$$f_u^{match} = \text{number of times } u \text{ is matched}$$

and

$$f_u^{sub} = \text{ number of times } u \text{ (in pronunciation) is substituted}$$

and

$$f_u^{del} = \text{ number of times } u \text{ is deleted}$$

and

$$f_u^{ins} = \text{ number of times } u \text{ is inserted}$$

In the context of Levenshtein features, the use of expanded ngram units does not make sense and is not supported. Like the expectation features, Levenshtein

features provide a powerful generalization ability as they are well-defined for words that have not been seen in training.

When multiple pronunciations of a given word are present, the one with the smallest edit distance is selected for the levenshtein features.

4.2.4 Language Model Features

The language model features are:

1. the language model scores of word transitions $f(s, s', o_{st}^{et}) = LM(s, s')$
2. a feature indicating whether the word is `<unk>`

With the `-full-lm` flag set, the features are specific to the (s, s') transition (thus there are a total of $|S| + 1$ language model features. With the `-unit-lm` flag set, there are just two language model features - the log-likelihood feature of the (s, s') transition, and the unknown-word indicator. This option is appropriate for training when multiple language models will be used with the acoustic model.

4.2.5 Baseline Features

It is often the case that speech researchers have high-performing baseline systems, and it can behoove the implementer of a new technique to leverage such a baseline. To facilitate this, SCARF allows the use of a special baseline detector stream in conjunction with a baseline feature. The baseline stream contains the one-best output of a baseline system. It has the format:
baseline
(word time)+
The time associated with a word is its midpoint. Denote the number of baseline detections in a timespan from st to et by $C(st, et)$. In the case that there is just one, let its value be denoted by $B(st, et)$. The baseline feature is defined as:

$$f_b(s, s', o_{st}^{et}) = \begin{cases} 1 & \text{if } C(st, et) = 1 \text{ and } B(st, et) = w(s') \\ -1 & \text{otherwise} \end{cases}$$

Thus, the baseline feature is 1 when a segment spans just one baseline word, and the label of the segment matches the baseline word. It can be seen that the contribution of the baseline features to a path score will be maximized when the segment length is equal to the number of baseline words, and the labeling of the segments is identical to the baseline labeling. Thus, by fixing a high enough weight on the baseline feature, baseline performance is guaranteed. In practice, the baseline weighting is learned and its value will depend on the relative power of the additional features.

5 Advantages

We conclude by pointing out some advantages of SCARF and some research directions. These include:

- The framework is built around the notion of segmental models, allowing a direct mapping from a region of audio to words without explicit subword units.
- At the same time, generalization ability is achieved through the use of expectation features and consistency features in general [1, 2]. In contrast to that work, SCARF allows for continuous speech recognition.
- Joint discriminative training of the acoustic and language models is possible if desired, and unnecessary if not desired. It is possible to train systems so that it is possible to change language models and dictionaries without retraining.
- Left word context is made available without extra computational burden so that observation functions can be a function of both the current and previous word.
- Implicit left and right context is available through the observations in surrounding segments.
- Multiple detector streams are supported.
- A wide variety of derived features can be automatically generated for the user.

It is hoped that through this functionality, researchers will be able to focus on the construction of effective segmental features, and test them in a complete continuous speech recognition system without incurring complex overhead.

Acknowledgements

We thank Asela Gunawardana for his advice and insight.

References

[1] G. Heigold, G. Zweig, X. Li, and P. Nguyen, "A flat direct model for speech recognition," in *Proc. ICASSP*, 2009.

[2] G. Zweig and P. Nguyen, "Maximum mutual information multiphone units in direct modeling," in *Proc. Interspeech*, 2009.

[3] S. Sarawagi and W. Cohen, "Semi-markov conditional random fields for information extraction," in *Proc. NIPS*, 2005.

[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. ICML*, 2001.

[5] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Interspeech*, 2005.

[6] G. Zweig, "Bayesian network structures and inference techniques for automatic speech recognition," *Computer Speech and Language*, 2003.

[7] G. Zweig, J. Bilmes, and et al., "Structurally discriminative graphical models for automatic speech recognition: Results from the 2001 johns hopkins summer workshop," in *ICASSP*, 2002.