A Segmental CRF Approach to Large Vocabulary Continuous Speech Recognition

Geoffrey Zweig and Patrick Nguyen
Microsoft Research, Redmond, WA
{gzweig,panguyen}@microsoft.com

***Abstract*—This paper proposes a segmental conditional random field framework for large vocabulary continuous speech recognition. Fundamental to this approach is the use of acoustic detectors as the basic input, and the automatic construction of a versatile set of segment-level features. The detector streams operate at multiple time 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 are naturally geared to explain long span phenomena such as formant trajectories, duration, and syllable stress patterns. Generalization to unseen words is possible through the use of decomposable consistency features [1], [2], and our framework allows for the joint or separate discriminative training of the acoustic and language models. An initial evaluation of this framework with voice search data from the Bing Mobile (BM) application results in a 2% absolute improvement over an HMM baseline.**

***Index Terms*—speech recognition, conditional random field, direct modeling, detector features**

I. Introduction

The use of Hidden Markov Models with MFCC or PLP derived features marks a state-of-the-art in speech recognition systems which is hard to beat. With discriminative training and speaker-adaptation techniques, advanced systems are both efficient and effective. Nevertheless, in spite of the power of current methods, there has been a significant amount of interest in alternative approaches from at least two different points of view: feature definition and underlying statistical models. In terms of features, it is reasonable to expect a performance improvement if we can inject additional knowledge into the system via the use of multiple feature representations and signal processing methods. Examples of work in this area include the use of Neural Net features [3], [4], landmark based ASR [5], and recent work with acoustic-detector based systems [6], [7], [8]. In terms of the underlying statistical models, there has been significant interest in non-generative techniques, also referred to as "direct models." Examples of this work include Maximum Entropy Markov Models [9], Conditional Random Fields [10], [11], [12], [13], and Flat Direct Models [1], [2].

In this work, we merge elements of both these research directions, and develop a segmental CRF approach for speech recognition. From the modeling perspective, this approach extends previous direct modeling work by operating at the *segment* rather than frame level, similar to segment level extensions previously made in a maximum likelihood framework, e.g. [14], [15]. The states in our model represent words, and features are defined on the combination of a hypothesized word and a temporal *span of observations*. This allows for the clean integration of a full ngram language model in the training and decoding processes. The necessary dynamic programming recursions are derived and presented along with a gradient-descent based training process. From the feature perspective, we use multi-scale detector streams, and, critically, propose a process for automatically generating several classes of segment-level features based on these streams. The resulting framework allows for the consistent integration of numerous detector streams, and joint or separate discriminative training of the language and acoustic models.

The remainder of this paper is organized as follows. In Section II, we present the mathematical formulation of segmental CRFs and relate it to past work. Section III then describes the specializations necessary to adapt it to the speech recognition task. Section IV presents the dynamic programming recursions necessary for computing with SCRFs. Section V continues by outlining the feature classes we use, and Section VI presents a set of experimental results using the complete framework.

II. Segmental CRF Formulation

Segmental Conditional Random Fields - also known as Semi-Markov Random Fields [16] or SCRFs - underlie our approach. They relax the Markov assumption from the frame-state level to the word level, where states now correspond with a variable and automatically derived time span. To explain these, we begin with the standard Conditional Random Field model [17], 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

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

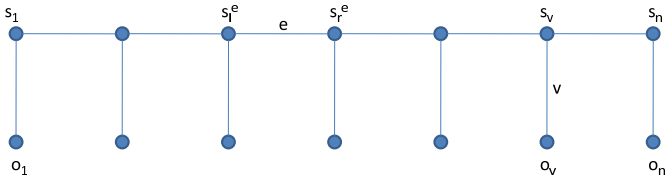

Fig. 1. Graphical representation of a CRF.

In speech recognition applications, the labels of interest, words, span multiple observation vectors, and the exact labeling of each observation is unknown. Hidden CRFs [18] address this issue by summing over all labelings consistent with a known or hypothesized word sequence. However, in the recursions presented in [18], the Markov property is applied at the individual state level, with the result that segmental properties are not modeled. Further, there is an inherent mismatch between the scale of the labels of interest (words) and the scale of the observations (100 per second). In this work, we are instead interested in making a direct association between a word-level state variable, and a word-scale span of observations. This is motivated by a desire to use long-span features that directly relate segment level acoustic properties to the word label, and to incorporate these in a model in as straightforward way as possible.

To do this, we adopt the formalism of segmental CRFs. 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 $l \leq N$ 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 are defined as with standard CRFs. Because of the segmental nature of the model, transitions only occur at logical points, and it is clear what span of observations to use to model a given symbol.

In considering Fig. 1, one may note that the same graphical representation has in the literature also been used for joint rather than conditional distributions. Further, the CRF formulation allows feature functions involving *all* the observations at once. Despite this ambiguity, we maintain the structure of [17]. Similarly, in Figs. 2 and 3, feature functions can in principle involve the entire observation span; the blocking structure is best thought of as indicating the *actual* observations which are compuationally used by the feature functions.

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 structure has the further benefit of allowing us to drop the distinction between g and f functions. To denote a block of original observations, we will use o_i^j to refer to observations i through j inclusive.

In the semi-CRF work of [16], the segmentation of the training data is known. However, in speech recognition applications, this is not the case. Therefore, in computing sequence likelihood, we must consider all segmentations consistent with

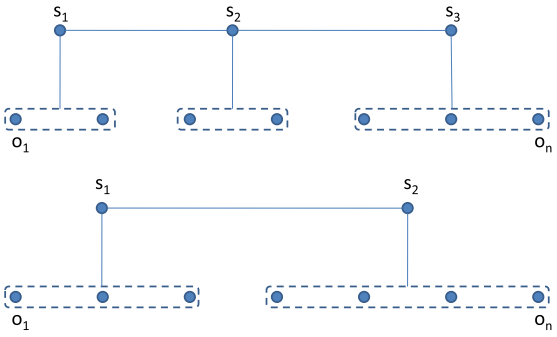

Fig. 2. A Segmental CRF and two different segmentations.

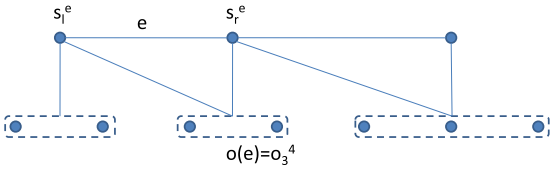

Fig. 3. Incorporating last-state information in a SCRF.

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 (horizontal) 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

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

Training is done by gradient descent using Rprop [19]. Taking the derivative of $\mathcal{L} = \log P(\mathbf{s}|\mathbf{o})$ with respect to λ_k we obtain the necessary gradient:

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

where $T_k(\mathbf{q}) = \sum_{e\in\mathbf{q}} f_k(s_l^e, s_r^e, o(e))$ and $T'_k(\mathbf{q}) = \sum_{e\in\mathbf{q}} f_k(s_l'^e, s_r'^e, o(e))$. This derivative can be computed efficiently with dynamic programming and a 1st pass state space reduction, using the recursions described in Section IV. In practice, we add L1 and L2 regularization terms to $\mathcal{L}$ to obtain an regularized objective function.

III. Adaptations For Speech Recognition

A. Language Modeling

Specific to the speech recognition task, we define the state transition functions with reference to a finite state representation of an ARPA language model. In our formulation, the states in the SCRF correspond to language model states rather than words *per se*, with the necessary word identity being implicit in the language model state. There is a state for each $0 \dots n-1$ gram word sequence in the language model. Thus, from a hypothetical 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. Any word sequence is possible, due to the presence of backoff arcs, ultimately to the null-history state. Note that this does not imply an exponential number of language model states: the number is limited to those seen in the training data, and in general count cutoffs limit the number further. We have experimented with two types of language model features. The first uses just one language model feature function, which simply returns the appropriate transition probability from the language model:

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

Note that this is not restricted to a bigram language model; for example the language model state might refer to a 5-gram history in a 6-gram model.

In the second approach, we jointly (and discriminatively) train the acoustic and language models - see also [20], [21]. To do this, we introduce a binary feature for each arc in a finite state representation of the language model. This feature is 1 if the arc is traversed in transitioning from one language model state to another on a hypothesized word. Note that in general this may involve traversing backoff arcs as well as word-labeled arcs. This approach is similar to [22], but integrated with acoustic training.

B. Pruning

In the segmental framework, it is theoretically necessary to consider the possible existence of a segment between any pair of observations. The runtime is quadratic in the number of observations, linear in the vocabulary, and linear in the number of language states. Thus, the computation is excessive unless constrained in some way. To implement this constraint, we use a form of fast-match [23] and use 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.

IV. Computation with SCRF models

A. Forward Backward Recursions

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

- An ARPA ngram backoff language model. We consider the language model to have a start state A - that associated with the ngram <s> - and a set of final states $\mathcal{F}$ - consisting of the ngram states ending in </s>.
- $start(t)$, which is a function that returns a set of words likely to start at observation t, along with their endtimes.
- $succ(s, w)$ delivers the language model state that results from seeing word w in state s.
- $features(s, s', st, et)$ returns a set of feature indices $\mathcal{K}$ and the corresponding feature values $f_k(s, s', o^{et}_{st})$.

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

$$\alpha(i,s) = \sum_{\mathbf{s} \in S^s_A} \sum_{\mathbf{q} \in Q^i_1 \text{s.t.} |\mathbf{q}|=|\mathbf{s}|} \exp\Big(\sum_{e \in \mathbf{q}, k} \lambda_k f_k(s^e_l, s^e_r, o(e))\Big).$$

When the sums are unconstrained, the normalization constant needed in training is given by $\sum_{s \in \mathcal{F}} \alpha(N, s)$ We define $\beta(i, s)$ as

$$\beta(i,s) = \sum_{\mathbf{s} \in S^{\omega \in \mathcal{F}}_s} \sum_{\mathbf{q} \in Q^N_{i+1} \text{s.t.} |\mathbf{q}|=|\mathbf{s}|} \exp\Big(\sum_{e \in \mathbf{q}, k} \lambda_k f_k(s^e_l, s^e_r, o(e))\Big).$$

The normalization score can again be extracted as $\beta(0, startstate)$. Now let $U(i, j, s)$ be the set of joint segmentations and state assignments such that a segment exists from time i though j inclusive, and is labeled by s. The sum of the path scores of all segmentations and state assignments in U is given by

$$\sum_{s_a} \alpha(i-1, s_a)\beta(j, s) \exp\Big(\sum_k \lambda_k f_k(s_a, s, o^j_i)\Big).$$

This fact is used in the computation of the gradient.

The following pseudocode outlines the efficient computation of the α and β quantities. 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 $i = 0 \dots N-1$
  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 $i = N \dots 1$
  foreach s s.t. $\beta(i, s) \neq 0$
    foreach $(ps, st) \in pred(s, i)$
      $\mathcal{K} = features(ps, s, st+1, i)$
      $\beta(st, ps) += \beta(i, s) \exp(\sum_{k \in \mathcal{K}} \lambda_k f_k(ps, s, o^i_{st+1}))$

B. Gradient Computation

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

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

For each utterance u, w e compute this once with constraints corresponding to the correct words to obtain $F_k^{cw}(u)$. 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 $F_k^{aw}(u)$. The gradient is given by:

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

Decoding proceeds analogously to the alpha recursion, with sums replaced by maxima, and only the best predecessor maintained.

V. Feature Construction

With the segmental CRF framework, we are now able to define features which span multiple observations. This contrasts with previous work, *e.g.* [18] which uses frame level features derived from the MFCCs, [9], which uses frame-level Gaussian rank features, and [11], [12], [13], which use frame-level neural net outputs as the basis for features. The process of feature creation is now described in detail. First, we describe the inputs which extract information from the utterance in the form of a sequence of symbols. Then, we describe how this information is then parameterized into the features which are used in our system.

A. Detector Inputs

The inputs to the feature creation process consist of streams of detector events, and optionally dictionaries that specify the detection sequences that are expected for the words. Each atomic detector stream provides a sequence of detector events, which consist of a unit which is detected and a time at which the detection occurs. Each stream defines its own unique unit set, and these are not shared across streams. In this work, detector inputs are generated from HMM systems, though this has been done for expediency and any detector events can be plugged in as easily.

A dictionary providing canonical word pronunciations can be 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. These allow for generalization to words not seen in the training data.

B. Feature Parameterization

With the inputs above, we are able to automatically define a variety of different feature types:

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 simply indicate whether a detector unit exists in a word's span. They are of the form:

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

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, are also automatically constructed. Since the total number of existence features is the number of words times the number of unit ngrams, we must constrain the creation of such features in some way. Therefore, we create an existence feature in two circumstances only:

- when a word and ngram of units exists together in a dictionary
- when a word exists in a transcription file, and an ngram exists in a corresponding detector file

2) Ngram Expectation Features: Denote the pronunciation of a word in terms of atomic units as $pron(w)$. Expectation features represent one of three events: the correct-accept, false-reject, or false accept of an ngram of units within a word's span. In order, these are of the form:

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

Expectation features are indicators of consistency between the units expected given a word ($pron(w)$), and those that are actually in the seen 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 also be automatically generated.

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.

Unit ngram features are again restricted to ngrams occurring in the training data.

3) Levenshtein Features: Levenshtein features are the strongest way of measuring the consistency between expected and observed detections. 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:

$$f_u^{match} = \text{number of times } u \text{ is matched}$$
$$f_u^{sub} = \text{number of times } u \text{ (in pronunciation) is substituted}$$
$$f_u^{del} = \text{number of times } u \text{ is deleted}$$
$$f_u^{ins} = \text{number of times } u \text{ is inserted}$$

In the context of Levenshtein features, the use of expanded ngram units does not make sense and is not used. 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) Language Model Features: As mentioned in Sec. III-A, the language model features are:

- The language model scores of word transitions $f(s, s', o_{st}^{et}) = LM(s, s')$
- A feature indicating whether the word is <unk>
- Optionally, a feature for each language model arc, indicating whether it is traversed in the path from s to s'.

5) Baseline Features: In order to leverage the existence of high-quality baseline HMM systems, we have also added a baseline feature. This is essentially a detector stream that specifies the 1-best word output of a baseline system. The time associated with each word in this stream 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 number of segments is equal to the number of baseline words, and the labeling of the segments is identical to the baseline labeling. Thus, as we can assign a weight approaching infinity to the baseline feature, baseline performance is guaranteed. In practice, of course, the baseline weighting is learned and its value will depend on the relative power of the additional features.

VI. Experiments

A. Corpus

To evaluate our approach, we have conducted a series of experiments with data from the Bing Mobile voice-search application (formerly known as Live Search for Mobile) [24]. This application allows users to request local businesses by voice, from their mobile phones, resulting in queries like "McDonald's," "Fairwood nails," and "Alabama State University." Key to our acquisition of training data, once a query is spoken, an N-best list of alternatives is presented for user validation. Speech comes in various challenging conditions, including outside noise, music, side-speech, sloppy pronunciation, and different acquisition channels.

For the purpose of this paper, we set aside 12,758 human-transcribed interactions for evaluation. For parameter tuning, we used a development set of 8,777 utterances. For training, we availed ourselves of roughly 3M spoken queries – 2100 hours of speech. The transcriptions used for these queries were the items the users selected from the N-best lists. We estimate that this form of supervision is about 90% accurate. Furthermore, we divided the 3M training set into two parts: a hyper-parameter training set, and a model training set. We reserved the former set to build feature detectors, while the SCRF weights were optimized on the model training set. The setup is similar to [2].

The baseline system is a conventional maximum likelihood trained HMM system with a trigram LM, using utterance-level mean normalized MFCCs and clustered cross-word triphones. It has 11k context dependent states, and 260k Gaussians. The baseline produces an error rate of 37.1%. The same HMM acoustic model generated the detector streams. Also, it generated an 10-best word sequence list for each utterance for use with the $start$ function. The oracle error rate of this system on the test data is 24.6%. We note that a discriminatively trained HMM baseline would provide a better baseline, but this baseline was unavailable, and would also be reflected in the SCRF system through the baseline feature. Our concern here is to validate the basic detector based segmental CRF approach. In contrast to [18], our features are detector and segment based, and not conveniently representable with frame based MMI training; thus discriminatively training a conventional system (e.g. HTK) with the same features is not a practicable alternative.

B. Results

After an initial set of experiments, we chose to use order 2 phone features (bi-phones) and order 1 multi-phone features; increasing either made little difference. In Table I, we report sentence error rate as detector streams are added. In all experiments, the baseline feature of Section V-B5 is used; its use alone reproduces the performance of the HMM baseline.

First, we measure the effect of complementing the baseline system with phone detections. We used existence, expectation, and Levenshtein features of order 2 as described in Section V-B. This results in a 0.9% absolute improvement,

	System	SER
1	HMM (baseline feature)	37.1%%
2	+phone	36.2%
3	+multi-phone	35.7%
4	+phone +multi-phone	35.4%
5	+phone +multi-phone (3-best)	35.2%
6	+phone +multi-phone (3-best) +full LM	35.0%

TABLE I
SENTENCE ERROR RATE (SER) AS DETECTOR STREAMS ARE ADDED. LINE 6 IMPLEMENTS DISCRIMINATIVE LM TRAINING AS WELL.

	System	SER
1	HMM +phone +multi-phone (3-best)	35.2%
2	Line 1, less existence features	35.7%
3	Line 1, less expectation features	35.8%
4	Line 1, less Levenshtein features	35.5%

TABLE II
THE EFFECT OF REMOVING INDIVIDUAL FEATURES.

Line 2. We repeat the experiment with a multi-phone stream described in [2], and features of order 1, to get a larger 1.4% improvement, Line 3. This is a validation of the multi-phone units, which are designed explicitly to maximize the mutual information between units and words. As we can see after combination of both streams as Line 4, the information provided by the phone detections is mostly contained in the multi-phone stream. Recognizing that the multi-phone segmentation may be ambiguous, we added the second and third best strings from the multi-phone decoding. They are treated entirely independently and share no weights in the model. This is reported in Line 5. Finally, on Line 6, we see a small further improvement from discriminative language model training. Altogether we achieve 2.1% of the 12.5% theoretically achievable given our search constraints.

To gauge the contribution of existence, expectation and Levenshtein features separately, we removed each from the system to obtain Table II. Removal of any feature results in a degradation of 0.3 to 0.6%.

VII. CONCLUSION

In this paper, we have proposed a segmental CRF approach to LVCSR. The approach has the following key characteristics:

- The Markov assumption is relaxed to the word level.
- The use of multi-scale detector events as the observations.
- Joint discriminative training of the acoustic and language models.
- The use of three broad classes of automatically derived features: Existence, Expectation and Levenshtein.
- The use of a simple baseline feature to facilitate integration with existing systems.
- Containment of the computational complexity through the use of a "fast-match" [23] like $start$ function.

Initial results on a state-of-the-art real-world voice-search task show an absolute improvement of 2% in sentence error rate.

Future directions include linguistically motivated features, features not based on HMMs, and a fully contained fast-match not reliant on a pre-existing HMM system.

ACKNOWLEDGMENT

We thank Asela Gunawardana for his advice and insight, and anonymous reviewers for their suggestions.

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] H. Hermansky, D. Ellis, and S. Sharma, "Tandem Connectionist Feature Stream Extraction for Conventional HMM Systems," in *ICASSP*, 2000.

[4] Q. Zhu, A. Stolcke, B. Chen, and N. Morgan, "Using MLP Features in SRI's Conversational Speech Recognition System," in *Interspeech*, 2005.

[5] M. Hasegawa-Johnson and et al., "Landmark-based speech recognition: Report of the 2004 johns hopkins summer workshop," in *ICASSP*, 2005.

[6] C-H. Lee, "From knowledge-ignorant o knowledge-rich modeling: A new speech research paradigm for next generation automatic speech recognition," in *ICSLP*, 2004.

[7] Ilana Bromberg, Qiang Fu, Jun Hou, Jinyu Li, Chengyuan Ma, Brett Matthews, Antonio Moreno-Daniel, Jeremy Morris, Sabato Marco Siniscalchi, Yu Tsao, and Yu Wang, "Detection-Based ASR in the Automatic Speech Attribute Transcription Project," in *Interspeech*, 2007.

[8] L. Hetherington, H. Shu, and J. Glass, "Flexible Multi-Stream Framework for Speech Recognition using Multi-Tape Finite-State Transducers," in *ICASSP*, 2006.

[9] H-K. J. Kuo and Y. Gao, "Maximum Entropy Direct Models for Speech Recognition," in *Proc. of ASRU*, 2003.

[10] A. Gunawardana, M. Mahajan, A. Acero, and J. Platt, "Hidden Conditional Random Fields for Phone Classification," in *Proc. of Interspeech*, 2005.

[11] J. Morris and E. Fosler-Lussier, "Discriminative Phonetic Recognition with Conditional Random Fields," in *HLT-NAACL*, 2006.

[12] J. Morris and E. Fosler-Lussier, "Further Experiments with Detector Based Conditional Random Fields in Phonetic Recognition," in *ICASSP*, 2007.

[13] J. Morris and E. Fosler-Lussier, "CRAMDEN Systems: Conditional Random Field Acoustic Models for Hidden Markov Models," in *ICASSP*, 2008.

[14] M. Ostendorf, V. Digalakis, and O. A. Kimball, "From hmms to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, 1996.

[15] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer, Speech and Language*, vol. 17, no. 2, 2003.

[16] S. Sarawagi and W. Cohen, "Semi-Markov Conditional Random Fields for Information Extraction," in *Proc. NIPS*, 2005.

[17] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proc. ICML*, 2001.

[18] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden Conditional Random Fields for Phone Classification," in *Interspeech*, 2005.

[19] M. Reidmiller, "Rprop - Description and Implementation Details," Tech. Rep., University of Karlsruhe, January 1994.

[20] H-K. Kuo, E. Fosler-Lussier, H. Jiang, and C-H. Lee, "Discriminative training of language models for speech recognition," in *ICASSP*, 2002.

[21] B. Roark, M. Saraclar, M. Collins, and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm," in *Proc. ACL*, 2004.

[22] H. Kuo, B. Kingsbury, and G. Zweig, "Discriminative Training of Decoding Graphs for Large Vocabulary Continuous Speech Recognition," in *ICASSP*, 2005.

[23] L.R. Bahl, P.V. de Souza, P.S. Gopalakrishnan, D. Nahamoo, and M. Picheny, "A Fast Match for Continuous Speech Recognition using Allophonic Models," in *ICASSP*, 1992.

[24] A. Acero and et al., "Live Search for Mobile: Web Services by Voice on the Cellphone," in *Proc. of ICASSP*, 2007.