COLING/ACL06, pp. 882–889, Association for Computational Linguistics, Sydney, Australia, 2006

Combining Statistical and Knowledge-based Spoken Language Understanding in Conditional Models

Ye-Yi Wang, Alex Acero, Milind Mahajan
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{yeyiwang,alexac,milindm}@microsoft.com

John Lee
Spoken Language Systems
MIT CSAIL
Cambridge, MA 02139, USA
jsylee@csail.mit.edu

Abstract

Spoken Language Understanding (SLU) addresses the problem of extracting semantic meaning conveyed in an utterance. The traditional knowledge-based approach to this problem is very expensive -- it requires joint expertise in natural language processing and speech recognition, and best practices in language engineering for every new domain. On the other hand, a statistical learning approach needs a large amount of annotated data for model training, which is seldom available in practical applications outside of large research labs. A generative HMM/CFG composite model, which integrates easy-to-obtain domain knowledge into a data-driven statistical learning framework, has previously been introduced to reduce data requirement. The major contribution of this paper is the investigation of integrating prior knowledge and statistical learning in a conditional model framework. We also study and compare conditional random fields (CRFs) with perceptron learning for SLU. Experimental results show that the conditional models achieve more than 20% relative reduction in slot error rate over the HMM/CFG model, which had already achieved an SLU accuracy at the same level as the best results reported on the ATIS data.

1 Introduction

Spoken Language Understanding (SLU) addresses the problem of extracting meaning conveyed in an utterance. Traditionally, the problem is solved with a knowledge-based approach, which requires joint expertise in natural language processing and speech recognition, and best practices in language engineering for every new domain. In the past decade many statistical learning approaches have been proposed, most of which exploit generative models, as surveyed in (Wang, Deng et al., 2005). While the data-driven approach addresses the difficulties in knowledge engineering, it requires a large amount of labeled data for model training, which is seldom available in practical applications outside of large research labs. To alleviate the problem, a generative HMM/CFG composite model has previously been introduced (Wang, Deng et al., 2005). It integrates a knowledge-based approach into a statistical learning framework, utilizing prior knowledge to compensate for the dearth of training data. In the ATIS evaluation (Price, 1990), this model achieves the same level of understanding accuracy (5.3% error rate on standard ATIS evaluation) as the best system (5.5% error rate), which is a semantic parsing system based on a manually developed grammar.

Discriminative training has been widely used for acoustic modeling in speech recognition (Bahl, Brown et al., 1986; Juang, Chou et al., 1997; Povey and Woodland, 2002). Most of the methods use the same generative model framework, exploit the same features, and apply discriminative training for parameter optimization. Along the same lines, we have recently exploited conditional models by directly porting the HMM/CFG model to Hidden Conditional Random Fields (HCRFs) (Gunawardana, Mahajan et al., 2005), but failed to obtain any improvement. This is mainly due to the vast parameter space, with the parameters settling at local optima. We then simplified the original model structure by removing the hidden variables, and introduced a number of important overlapping and non-homogeneous features. The resulting Conditional Random Fields (CRFs) (Lafferty, McCallum et al., 2001) yielded a 21% relative improvement in SLU accuracy. We also applied a much simpler perceptron learning algorithm on the conditional model and observed improved SLU accuracy as well.

In this paper, we will first introduce the generative HMM/CFG composite model, then discuss the problem of directly porting the model to HCRFs, and finally introduce the CRFs and

the features that obtain the best SLU result on ATIS test data. We compare the CRF and perceptron training performances on the task.

2 Generative Models

The HMM/CFG composite model (Wang, Deng et al., 2005) adopts a pattern recognition approach to SLU. Given a word sequence W, an SLU component needs to find the semantic representation of the meaning M that has the maximum *a posteriori* probability $\Pr(M|W)$:

$$\hat{M} = \arg\max_M \Pr(M|W)$$
$$= \arg\max_M \Pr(W|M) \cdot \Pr(M)$$

The composite model integrates domain knowledge by setting the topology of the prior model, $\Pr(M)$, according to the domain semantics; and by using PCFG rules as part of the lexicalization model $\Pr(W|M)$.

The domain semantics define an application's semantic structure with semantic frames. Figure 1 shows a simplified example of three semantic frames in the ATIS domain. The two frames with the "toplevel" attribute are also known as commands. The "filler" attribute of a slot specifies the semantic object that can fill it. Each slot may be associated with a CFG rule, and the filler semantic object must be instantiated by a word string that is covered by that rule. For example, the string "Seattle" is covered by the "City" rule in a CFG. It can therefore fill the ACity (ArrivalCity) or the DCity (DepartureCity) slot, and instantiate a Flight frame. This frame can then fill the Flight slot of a ShowFlight frame. Figure 2 shows a semantic representation according to these frames.

```
< frame name="ShowFlight" toplevel="1" >
    <slot name="Flight" filler="Flight"/ >
</ frame >
< frame name="GroundTrans" toplevel="1" >
    < slot name="City" filler="City"/ >
</ frame >
< frame name="Flight" >
    <slot name="DCity" filler="City"/ >
    < slot name="ACity" filler="City"/ >
</ frame >
```

Figure 1. Simplified domain semantics for the ATIS domain.

The semantic prior model comprises the HMM topology and state transition probabilities. The topology is determined by the domain semantics, and the transition probabilities can be estimated from training data. Figure 3 shows the topology of the underlying states in the statistical model for the semantic frames in Figure 1. On top is the transition network for the two top-level commands. At the bottom is a zoomed-in view for the "Flight" sub-network. State 1 and state 4 are called precommands. State 3 and state 6 are called postcommands. States 2, 5, 8 and 9 represent slots. A slot is actually a three-state sequence — the slot state is preceded by a preamble state and followed by a postamble state, both represented by black circles. They provide contextual clues for the slot's identity.

```
< ShowFlight >
    < Flight >
        < DCity filler="City" > Seattle < /DCity >
        < ACity filler="City" > Boston < /ACity >
    < /Flight >
< /ShowFlight >
```

Figure 2. The semantic representation for "Show me the flights departing from Seattle arriving at Boston" is an instantiation of the semantic frames in Figure 1.

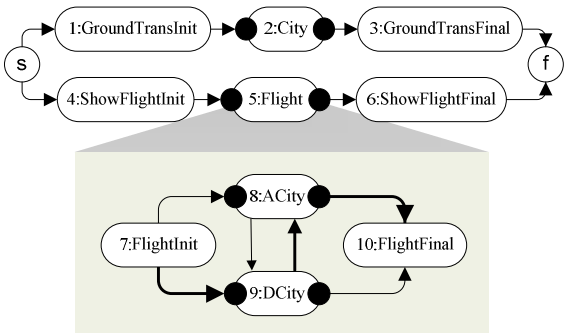

Figure 3. The HMM/CFG model's state topology, as determined by the semantic frames in Figure 1.

The lexicalization model, $\Pr(W|M)$, depicts the process of sentence generation from the topology by estimating the distribution of words emitted by a state. It uses state-dependent n-grams to model the precommands, postcommands, preambles and postambles, and uses knowledge-based CFG rules to model the slot fillers. These rules help compensate for the dearth of domain-specific data. In the remainder of this paper we will say a string is "covered by a CFG non-terminal (NT)", or equivalently, is "CFG-covered for *s*" if the string can be parsed by the CFG rule corresponding to the slot *s*.

Given the semantic representation in Figure 2, the state sequence through the model topology in

Figure 3 is deterministic, as shown in Figure 4. However, the words are not aligned to the states in the shaded boxes. The parameters in their corresponding n-gram models can be estimated with an EM algorithm that treats the alignments as hidden variables.

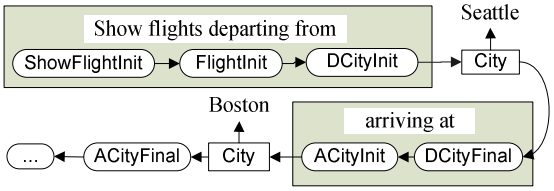

Figure 4. Word/state alignments. The segmentation of the word sequences in the shaded region is hidden.

The HMM/CFG composite model was evaluated in the ATIS domain (Price, 1990). The model was trained with ATIS3 category A training data (~1700 annotated sentences) and tested with the 1993 ATIS3 category A test sentences (470 sentences with 1702 reference slots). The slot insertion-deletion-substitution error rate (SER) of the test set is 5.0%, leading to a 5.3% semantic error rate in the standard end-to-end ATIS evaluation, which is slightly better than the best manually developed system (5.5%). Moreover, a steep drop in the error rate is observed after training with only the first two hundred sentences. This demonstrates that the inclusion of prior knowledge in the statistical model helps alleviate the data sparseness problem.

3 Conditional Models

We investigated the application of conditional models to SLU. The problem is formulated as assigning a label l to each element in an observation $\mathbf{o}$. Here, $\mathbf{o}$ consists of a word sequence $\mathbf{o}_1^\tau$ and a list of CFG non-terminals (NT) that cover its subsequences, as illustrated in Figure 5. The task is to label "two" as the "Num-of-tickets" slot of the "ShowFlight" command, and "Washington D.C." as the ArrivalCity slot for the same command. To do so, the model must be able to resolve several kinds of ambiguities:

1. Filler/non-filler ambiguity, e.g., "two" can either fill a Num-of-tickets slot, or its homonym "to" can form part of the preamble of an ArrivalCity slot.
2. CFG ambiguity, e.g., "Washington" can be CFG-covered as either City or State.
3. Segmentation ambiguity, e.g., [Washington] [D.C.] vs. [Washington D.C.].
4. Semantic label ambiguity, e.g., "Washington D.C." can fill either an ArrivalCity or DepartureCity slot.

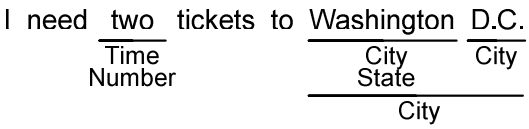

Figure 5. The observation includes a word sequence and the subsequences covered by CFG non-terminals.

3.1 CRFs and HCRFs

Conditional Random Fields (CRFs) (Lafferty, McCallum et al., 2001) are undirected conditional graphical models that assign the conditional probability of a state (label) sequence s_1^τ with respect to a vector of features $\mathbf{f}(s_1^\tau, \mathbf{o}_1^\tau)$. They are of the following form:

$$p(s_1^\tau \mid \mathbf{o}; \lambda) = \frac{1}{z(\mathbf{o}; \lambda)} \exp\left(\lambda \cdot \mathbf{f}(s_1^\tau, \mathbf{o})\right). \quad (1)$$

Here $z(\mathbf{o}; \lambda) = \sum_{s_1^\tau} \exp\left(\lambda \cdot \mathbf{f}(s_1^\tau, \mathbf{o})\right)$ normalizes the distribution over all possible state sequences. The parameter vector λ is trained conditionally (discriminatively). If we assume that s_1^τ is a Markov chain given $\mathbf{o}$ and the feature functions only depend on two adjacent states, then

$$p(s_1^\tau \mid \mathbf{o}; \lambda) = \frac{1}{z(\mathbf{o}; \lambda)} \exp\left(\sum_k \lambda_k \sum_{t=1}^{\tau} f_k(s^{(t-1)}, s^{(t)}, \mathbf{o}, t)\right) \quad (2)$$

In some cases, it may be natural to exploit features on variables that are not directly observed. For example, a feature for the Flight preamble may be defined in terms of an observed word and an unobserved state in the shaded region in Figure 4:

$$f_{\text{FlightInit,flights}}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t) = \begin{cases} 1 & \text{if } s^{(t)} = \text{FlightInit} \wedge \mathbf{o}^t = \text{flights}; \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

In this case, the state sequence s_1^τ is only partially observed in the meaning representation $M : M(s_5) = \text{"DCity"} \wedge M(s_8) = \text{"ACity"}$ for the words "Seattle" and "Boston". The states for the remaining words are hidden. Let $\Gamma(M)$ represent the set of all state sequences that satisfy the constraints imposed by M. To obtain the conditional probability of M, we need to sum over all possible labels for the hidden states:

$$p(M \mid \mathbf{o}; \lambda) = \frac{1}{z(\mathbf{o}; \lambda)} \sum_{s_1^\tau \in \Gamma(M)} \exp\left(\sum_k \lambda_k \sum_{t=1}^{\tau} f_k(s^{(t-1)}, s^{(t)}, \mathbf{o}, t) \right)$$

CRFs with features dependent on hidden state variables are called Hidden Conditional Random Fields (HCRFs). They have been applied to tasks such as phonetic classification (Gunawardana, Mahajan et al., 2005) and object recognition (Quattoni, Collins et al., 2004).

3.2 Conditional Model Training

We train CRFs and HCRFs with gradient-based optimization algorithms that maximize the log posterior. The gradient of the objective function is

$$\nabla_\lambda L(\lambda) = \mathbf{E}_{\tilde{P}(\mathbf{l},\mathbf{o})P(s_1^\tau \mid \mathbf{l},\mathbf{o})}\left[\mathbf{f}(s_1^\tau, \mathbf{o}); \lambda\right] - \mathbf{E}_{\tilde{P}(\mathbf{o})P(s_1^\tau \mid \mathbf{o})}\left[\mathbf{f}(s_1^\tau, \mathbf{o}); \lambda\right]$$

which is the difference between the conditional expectation of the feature vector given the observation sequence and label sequence, and the conditional expectation given the observation sequence alone. With the Markov assumption in Eq. (2), these expectations can be computed using a forward-backward-like dynamic programming algorithm. For CRFs, whose features do not depend on hidden state sequences, the first expectation is simply the feature counts given the observation and label sequences. In this work, we applied stochastic gradient descent (SGD) (Kushner and Yin, 1997) for parameter optimization. In our experiments on several different tasks, it is faster than L-BFGS (Nocedal and Wright, 1999), a quasi-Newton optimization algorithm.

3.3 CRFs and Perceptron Learning

Perceptron training for conditional models (Collins, 2002) is an approximation to the SGD algorithm, using feature counts from the Viterbi label sequence in lieu of expected feature counts. It eliminates the need of a forward-backward algorithm to collect the expected counts, hence greatly speeds up model training. This algorithm can be viewed as using the minimum margin of a training example (i.e., the difference in the log conditional probability of the reference label sequence and the Viterbi label sequence) as the objective function instead of the conditional probability:

$$L'(\lambda) = \log P(\mathbf{l} \mid \mathbf{o}; \lambda) - \max_{\mathbf{l}'} \log P(\mathbf{l}' \mid \mathbf{o}; \lambda)$$

Here again, **o** is the observation and **l** is its reference label sequence. In perceptron training, the parameter updating stops when the Viterbi label sequence is the same as the reference label sequence. In contrast, the optimization based on the log posterior probability objective function keeps pulling probability mass from all incorrect label sequences to the reference label sequence until convergence.

In both perceptron and CRF training, we average the parameters over training iterations (Collins, 2002).

4 Porting HMM/CFG Model to HCRFs

In our first experiment, we would like to exploit the discriminative training capability of a conditional model without changing the HMM/CFG model's topology and feature set. Since the state sequence is only partially labeled, an HCRF is used to model the conditional distribution of the labels.

4.1 Features

We used the same state topology and features as those in the HMM/CFG composite model. The following indicator features are included:

Command prior features capture the *a priori* likelihood of different top-level commands:

$$f_c^{PR}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t) = \begin{cases} 1 & \text{if } t{=}0 \wedge \mathrm{C}(s^{(t)}) = c \\ 0 & \text{otherwise} \end{cases}, \forall c \in \mathrm{CommandSet}$$

Here C(s) stands for the name of the command that corresponds to the transition network containing state s.

State Transition features capture the likelihood of transition from one state to another:

$$f_{s_1,s_2}^{TR}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t) = \begin{cases} 1 & \text{if } s^{(t-1)} = s_1, s^{(t)} = s_2 \\ 0 & \text{otherwise} \end{cases},$$

where $s_1 \to s_2$ is a legal transition according to the state topology.

Unigram and *Bigram* features capture the likelihoods of words emitted by a state:

$$f^{UG}_{s,w}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = \begin{cases} 1 & \text{if } s^{(t)} = s \wedge \mathbf{o}^t = w \\ 0 & \text{otherwise} \end{cases},$$

$$f^{BG}_{s,w_1,w_2}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = \begin{cases} 1 & \text{if } s^{(t-1)} = s \wedge s^{(t)} = s \wedge \mathbf{o}^{t-1} = w_1 \wedge \mathbf{o}^t = w_2 \\ 0 & \text{otherwise} \end{cases},$$

$$\forall s \mid \neg \text{isFiller}(s); \forall w, w_1 w_2 \in \text{TrainingData}$$

The condition isFiller(s_1) restricts s_1 to be a slot state and not a pre- or postamble state.

4.2 Experiments

The model is trained with SGD with the parameters initialized in two ways. The *flat start* initialization sets all parameters to 0. The *generative model* initialization uses the parameters trained by the HMM/CFG model.

Figure 6 shows the test set slot error rates (SER) at different training iterations. With the flat start initialization (top curve), the error rate never comes close to the 5% baseline error rate of the HMM/CFG model. With the generative model initialization, the error rate is reduced to 4.8% at the second iteration, but the model quickly gets over-trained afterwards.

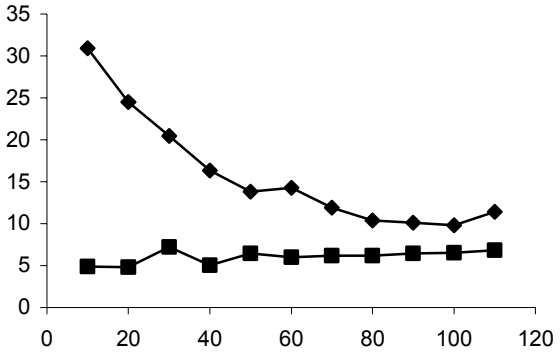

Figure 6. Test set slot error rates (in %) at different training iterations. The top curve is for the flat start initialization, the bottom for the generative model initialization.

The failure of the direct porting of the generative model to the conditional model can be attributed to the following reasons:

- The conditional log-likelihood function is no longer a convex function due to the summation over hidden variables. This makes the model highly likely to settle on a local optimum. The fact that the flat start initialization failed to achieve the accuracy of the generative model initialization is a clear indication of the problem.
- In order to account for words in the test data, the n-grams in the generative model are properly smoothed with back-offs to the uniform distribution over the vocabulary. This results in a huge number of parameters, many of which cannot be estimated reliably in the conditional model, given that model regularization is not as well studied as in n-grams.
- The hidden variables make parameter estimation less reliable, given only a small amount of training data.

5 CRFs for SLU

An important lesson we have learned from the previous experiment is that we should not think generatively when applying conditional models. While it is important to find cues that help identify the slots, there is no need to exhaustively model the generation of every word in a sentence. Hence, the distinctions between pre- and postcommands, and pre- and postambles are no longer necessary. Every word that appears between two slots is labeled as the preamble state of the second slot, as illustrated in Figure 7. This labeling scheme effectively removes the hidden variables and simplifies the model to a CRF. It not only expedites model training, but also prevents parameters from settling at a local optimum, because the log conditional probability is now a convex function.

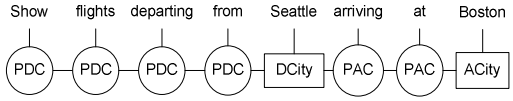

Figure 7. Once the slots are marked in the simplified model topology, the state sequence is fully marked, leaving no hidden variables and resulting in a CRF. Here, PAC stands for "preamble for arrival city," and PDC for "preamble for departure city."

The command prior and state transition features (with fewer states) are the same as in the HCRF model. For unigrams and bigrams, only those that occur in front of a CFG-covered string are considered. If the string is CFG-covered for slot s, then the unigram and bigram features for the preamble state of s are included. Suppose the words "that departs" occur at positions $t-1$ and t in front of the word "Seattle," which is CFG-covered by the non-terminal **City**. Since **City** can fill a DepartureCity or ArrivalCity slot, the four following features are introduced:

$$f^{UG}_{\text{PDC,that}}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = f^{UG}_{\text{PAC,that}}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = 1$$

And

$$f^{BG}_{\text{PDC,that,departs}}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) =$$
$$f^{BG}_{\text{PAC,that,departs}}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = 1$$

Formally,

$$f^{UG}_{s,w}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t) = \begin{cases} 1 & \text{if } s^{(t)} = s \wedge \mathbf{o}^t = w \\ 0 & \text{otherwise} \end{cases},$$

$$f^{BG}_{s,w_1,w_2}(s^{(t-1)}, s^{(t)}, \mathbf{o}^{\tau}_1, t)$$
$$= \begin{cases} 1 & \text{if } s^{(t-1)} = s^{(t)} = s \wedge \mathbf{o}^{t-1} = w_1 \wedge \mathbf{o}^t = w_2 \\ 0 & \text{otherwise} \end{cases},$$

$\forall s \mid \neg\text{isFiller}(s)$;

$\forall w, w_1 w_2 \mid$ in the training data, w and $w_1 w_2$ appears in front of sequence that is CFG-covered for s.

5.1 Additional Features

One advantage of CRFs over generative models is the ease with which overlapping features can be incorporated. In this section, we describe three additional feature sets.

The first set addresses a side effect of not modeling the generation of every word in a sentence. Suppose a preamble state has never occurred in a position that is confusable with a slot state *s*, and a word that is CFG-covered for *s* has never occurred as part of the preamble state in the training data. Then, the unigram feature of the word for that preamble state has weight 0, and there is thus no penalty for mislabeling the word as the preamble. This is one of the most common errors observed in the development set. The *chunk coverage for preamble words* feature introduced to model the likelihood of a CFG-covered word being labeled as a preamble:

$$f^{CC}_{c,NT}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t)$$
$$= \begin{cases} 1 & \text{if } \mathrm{C}(s^{(t)}) = c \wedge \text{covers}(NT, \mathbf{o}^t) \wedge \text{isPre}(s^{(t)}) \\ 0 & \text{otherwise} \end{cases}$$

where isPre(*s*) indicates that *s* is a preamble state.

Often, the identity of a slot depends on the preambles of the previous slot. For example, "at two PM" is a DepartureTime in "flight from Seattle to Boston at two PM", but it is an ArrivalTime in "flight departing from Seattle arriving in Boston at two PM." In both cases, the previous slot is ArrivalCity, so the state transition features are not helpful for disambiguation. The identity of the time slot depends not on the ArrivalCity slot, but on its preamble. Our second feature set, *previous-slot context*, introduces this dependency to the model:

$$f^{PC}_{s_1,s_2,w}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t)$$
$$= \begin{cases} 1 & \text{if } s^{(t-1)} = s_1 \wedge s^{(t)} = s_2 \wedge w \in \Theta(s_1, \mathbf{o}, t-1) \\ & \wedge\ \text{isFiller}(s_1) \wedge\ \text{Slot}(s_1) \neq \text{Slot}(s_2) \\ 0 & \text{otherwise} \end{cases}$$

Here Slot(s) stands for the slot associated with the state *s*, which can be a filler state or a preamble state, as shown in Figure 7. $\Theta(s_1, \mathbf{o}, t-1)$ is the set of *k* words (where *k* is an adjustable window size) in front of the longest sequence that ends at position $t-1$ and that is CFG-covered by $\text{Slot}(s_1)$.

The third feature set is intended to penalize erroneous segmentation, such as segmenting "Washington D.C." into two separate City slots. The *chunk coverage for slot boundary* feature is activated when a slot boundary is covered by a CFG non-terminal *NT*, i.e., when words in two consecutive slots ("Washington" and "D.C.") can also be covered by one single slot:

$$f^{SB}_{c,NT}(s^{(t-1)}, s^{(t)}, \mathbf{o}, t)$$
$$= \begin{cases} 1 & \text{if } \mathrm{C}(s^{(t)}) = c \wedge \text{covers}(NT, \mathbf{o}^t_{t-1}) \\ & \wedge\ \text{isFiller}(s^{(t-1)}) \wedge\ \text{isFiller}(s^{(t)}) \\ & \wedge\ s^{(t-1)} \neq s^{(t)} \\ 0 & \text{otherwise} \end{cases}$$

This feature set shares its weights with the *chunk coverage features for preamble words*, and does not introduce any new parameters.

Features	# of Param.	SER
Command Prior	6	
+State Transition	+1377	18.68%
+Unigrams	+14433	7.29%
+Bigrams	+58191	7.23%
+Chunk Cov Preamble Word	+156	6.87%
+Previous-Slot Context	+290	5.46%
+Chunk Cov Slot Boundaries	+0	3.94%

Table 1. Number of additional parameters and the slot error rate after each new feature set is introduced.

5.2 Experiments

Since the objective function is convex, the optimization algorithm does not make any significant difference on SLU accuracy. We

trained the model with SGD. Other optimization algorithm like Stochastic Meta-Decent (Vishwanathan, Schraudolph et al., 2006) can be used to speed up the convergence. The training stopping criterion is cross-validated with the development set.

Table 1 shows the number of new parameters and the slot error rate (SER) on the test data, after each new feature set is introduced. The new features improve the prediction of slot identities and reduce the SER by 21%, relative to the generative HMM/CFG composite model.

The figures below show in detail the impact of the n-gram, previous-slot context and chunk coverage features. The chunk coverage feature has three settings: 0 stands for no chunk coverage features; 1 for chunk coverage features for preamble words only; and 2 for both words and slot boundaries.

Figure 8 shows the impact of the order of n-gram features. Zero-order means no lexical features for preamble states are included. As the figure illustrates, the inclusion of CFG rules for slot filler states and domain-specific knowledge about command priors and slot transitions have already produced a reasonable SER under 15%. Unigram features for preamble states cut the error by more than 50%, while the impact of bigram features is not consistent -- it yields a small positive or negative difference depending on other experimental parameter settings.

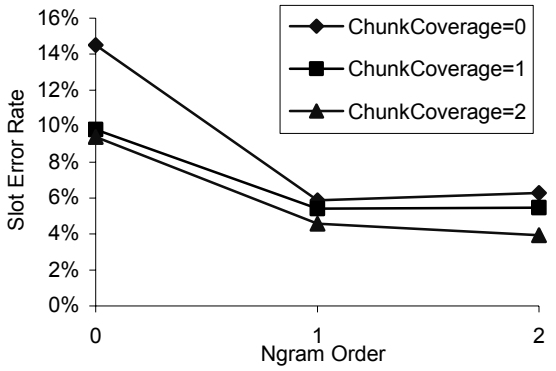

Figure 8. Effects of the order of n-grams on SER. The window size for the previous-slot context features is 2.

Figure 9 shows the impact of the CFG chunk coverage feature. Coverage for both preamble words and slot boundaries help improve the SLU accuracy.

Figure 10 shows the impact of the window size for the previous-slot context feature. Here, 0 means that the previous-slot context feature is not used. When the window size is *k*, the *k* words in front of the longest previous CFG-covered word sequence are included as the previous-slot unigram context features. As the figure illustrates, this feature significantly reduces SER, while the window size does not make any significant difference.

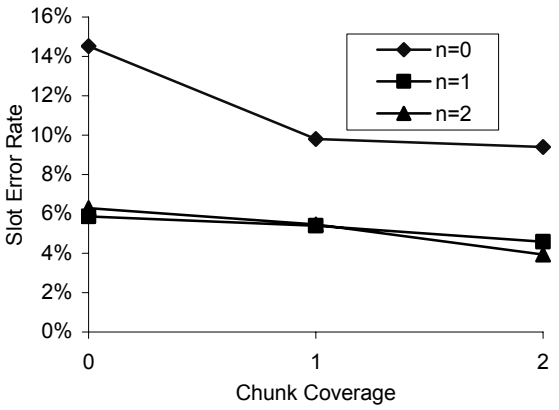

Figure 9. Effects of the chunk coverage feature. The window size for the previous-slot context feature is 2. The three lines correspond to different n-gram orders, where 0-gram indicates that no preamble lexical features are used.

It is important to note that overlapping features like f^{CC}, f^{SB} and f^{PC} could not be easily incorporated into a generative model.

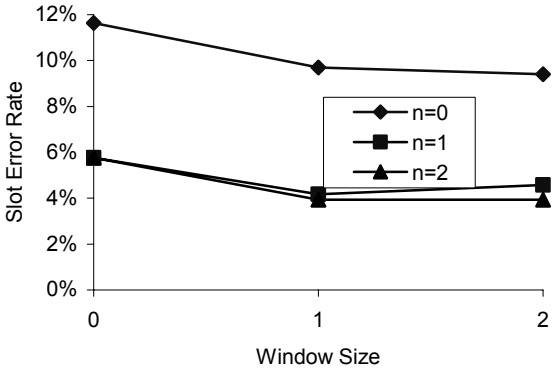

Figure 10. Effects of the window size of the previous-slot context feature. The three lines represent different orders of n-grams (0, 1, and 2). Chunk coverage features for both preamble words and slot boundaries are used.

5.3 CRFs vs. Perceptrons

Table 2 compares the perceptron and CRF training algorithms, using chunk coverage features for both preamble words and slot boundaries, with which the best accuracy results

are achieved. Both improve upon the 5% baseline SER from the generative HMM/CFG model. CRF training outperforms the perceptron in most settings, except for the one with unigram features for preamble states and with window size 1 -- the model with the fewest parameters. One possible explanation is as follows. The objective function in CRFs is a convex function, and so SGD can find the single global optimum for it. In contrast, the objective function for the perceptron, which is the difference between two convex functions, is not convex. The gradient ascent approach in perceptron training is hence more likely to settle on a local optimum as the model becomes more complicated.

	PSWSize=1		PSWSize=2	
	Perceptron	CRFs	Perceptron	CRFs
n=1	3.76%	4.11%	4.23%	3.94%
n=2	4.76%	4.41%	4.58%	3.94%

Table 2. Perceptron vs. CRF training. Chunk coverage features are used for both preamble words and slot boundaries. PSWSize stands for the window size of the previous-slot context feature. N is the order of the n-gram features.

The biggest advantage of perceptron learning is its speed. It directly counts the occurrence of features given an observation and its reference label sequence and Viterbi label sequence, with no need to collect expected feature counts with a forward-backward-like algorithm. Not only is each iteration faster, but fewer iterations are required, when using SLU accuracy on a cross-validation set as the stopping criterion. Overall, perceptron training is 5 to 8 times faster than CRF training.

6 Conclusions

This paper has introduced a conditional model framework that integrates statistical learning with a knowledge-based approach to SLU. We have shown that a conditional model reduces SLU slot error rate by more than 20% over the generative HMM/CFG composite model. The improvement was mostly due to the introduction of new overlapping features into the model. We have also discussed our experience in directly porting a generative model to a conditional model, and demonstrated that it may not be beneficial at all if we still think generatively in conditional modeling; more specifically, replicating the feature set of a generative model in a conditional model may not help much. The key benefit of conditional models is the ease with which they can incorporate overlapping and non-homogeneous features. This is consistent with the finding in the application of conditional models for POS tagging (Lafferty, McCallum et al., 2001). The paper also compares different training algorithms for conditional models. In most cases, CRF training is more accurate, however, perceptron training is much faster.

References

Bahl, L., P. Brown, et al. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. IEEE International Conference on Acoustics, Speech, and Signal Processing.

Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. EMNLP, Philadelphia, PA.

Gunawardana, A., M. Mahajan, et al. 2005. Hidden conditional random fields for phone classification. Eurospeech, Lisbon, Portugal.

Juang, B.-H., W. Chou, et al. 1997. "Minimum classification error rate methods for speech recognition." IEEE Transactions on Speech and Audio Processing 5(3): 257-265.

Kushner, H. J. and G. G. Yin. 1997. Stochastic approximation algorithms and applications, Springer-Verlag.

Lafferty, J., A. McCallum, et al. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. ICML.

Nocedal, J. and S. J. Wright. 1999. Numerical optimization, Springer-Verlag.

Povey, D. and P. C. Woodland. 2002. Minimum phone error and I-smoothing for improved discriminative training. IEEE International Conference on Acoustics, Speech, and Signal Processing.

Price, P. 1990. Evaluation of spoken language system: the ATIS domain. DARPA Speech and Natural Language Workshop, Hidden Valley, PA.

Quattoni, A., M. Collins and T. Darrell. 2004. Conditional Random Fields for Object Recognition. NIPS.

Vishwanathan, S. V. N., N. N. Schraudolph, et al. 2006. Accelerated Training of conditional random fields with stochastic meta-descent. The Learning Workshop, Snowbird, Utah.

Wang, Y.-Y., L. Deng, et al. 2005. "Spoken language understanding --- an introduction to the statistical framework." IEEE Signal Processing Magazine 22(5): 16-31.