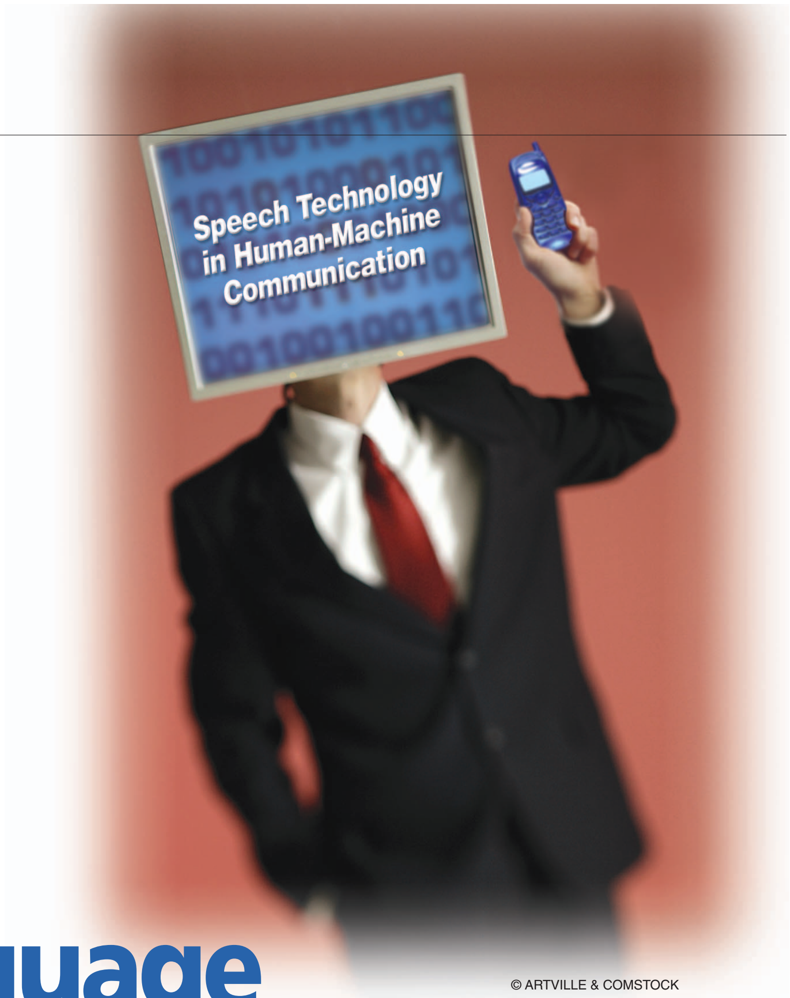
Ye-Yi Wang, Li Deng, and Alex Acero



© ARTVILLE & COMSTOCK

An introduction to the statistical framework

Spoken Language Understanding

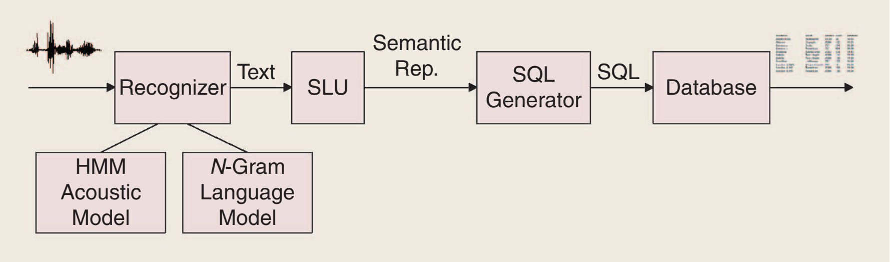
Besides dictation, there are many other practical applications for speech recognition, including command and control, spoken dialog systems [1], [2], speech-to-speech translation [3], and multimodal interaction [4]–[6]. The success of these applications relies on the correct recognition not only of what is said but also of what is meant. In contrast to automatic speech recognition (ASR), which converts a speaker's spoken utterance into a text string, spoken language understanding (SLU) is aimed at interpreting user's intentions from their speech utterances. Traditionally, this has been accomplished by writing context-free grammars (CFGs) or unification grammars (UGs) by hand. The manual grammar authoring process is laborious and expensive, requiring much expertise. In recent years, many data-driven models have been proposed for this problem. The main purpose of this article is to provide an introduction to the statistical framework common in SLU, which has not been widely revealed to signal processing readers in the past.

SLU is closely related to natural language understanding (NLU), a field that has been studied for half a century. However, the problem of SLU has its own characteristics. Unlike general-domain NLU, SLU (in the current state of technology) focuses only on specific application domains. Hence, many domain-specific constraints can be included in the understanding model. Ostensibly, this may make the problem easier to solve. Unfortunately, spoken language is much noisier than written language. The inputs to an SLU system are not as well formed as those to an NLU system. They often do not comply with rigid syntactic constraints. Disfluencies such as false starts, repairs, and hesitations are pervasive, especially in conversational speech, and errors made by speech recognizers are inevitable. Therefore, robustness is one of the most important issues in SLU. On the other hand, a robust solution tends to over-generalize and introduce ambiguities, leading to reduction of understanding accuracy. A major challenge to SLU is to strike

1053-5888/05/$20.00©2005IEEE

an optimal balance between the robustness and the constraints that prevent overgeneralizations and reduce ambiguities.

The study of SLU began in the 1970s with the Defense Advanced Research Projects Agency (DARPA) Speech Understanding Research (SUR) and then the Resource Management (RM) tasks. At this early stage, NLU techniques like finite state machine (FSM) and augmented transition networks (ATNs) were applied for SLU [7]. The study of SLU surged in the 1990s, with the DARPA-sponsored Air Travel Information System (ATIS) evaluations [8]. Many of the SLU technologies developed for ATIS were used in the more recent DARPA Communicator program, which focused on the rapid and cost-effective development of multimodal, speech-enabled dialog systems [2]. Multiple research labs from both academia and industry developed systems that attempted to understand users' spontaneous spoken queries for air travel information (including flight information, ground transportation information, airport service information, etc.) and then obtain the answer from a standard database.



[FIG1] A typical ATIS system consists of 1) a speech recognizer with both the acoustic model and language model trained with the ATIS-specific data, 2) an SLU system that extracts the semantic representation (meaning) from the recognized text, and 3) a SQL generator that automatically generates the database query based on the semantic representation.

Figure 1 shows the typical architecture of an ATIS system. The systems were evaluated, component wise and end to end, with a common test set. Because spontaneous speech was used, robust understanding was the focus of the SLU study. About half a dozen robust SLU systems were developed during the evaluation timeframe.

Both knowledge-based and statistical systems were developed for the ATIS domain. In knowledge-based systems [9]–[11], developers write a syntactic/semantic grammar, and a robust parser analyzes the input text with the grammar. The major advantage of this approach is that it works without a large amount of training data. However, a knowledge-based system is expensive to develop for four reasons: 1) grammar development is an error-prone process; 2) it takes multiple rounds to fine-tune a grammar; 3) combined linguistic and engineering expertise is required to construct a grammar with good coverage and optimized performance; and 4) such a grammar is difficult and expensive to maintain. Ideally, an SLU system should be able to automatically adapt to the real data collected after its deployment. In contrast, knowledge-based systems require an expert's involvement, and sometimes even the involvement of the original system designer, in the adaptation loop.

SLU approaches based on statistical learning directly address many of the problems associated with knowledge-based systems. Statistical SLU systems can automatically learn from example sentences with their corresponding semantics. Compared with manual grammar authoring, the annotations are much easier to create and do not require specialized knowledge. The statistical approach can adapt to new data, possibly via unsupervised learning. One disadvantage of such statistical approaches, however, is the data-sparseness problem. The requirement of a large amount of training data is not very practical for real-world applications, which are quite different from those studied in research labs. This is especially the case at the early stage of system development.

In the remainder of this article, we will introduce a general framework for statistical SLU and review various types of statistical SLU models in the literature. We assume that most readers have been exposed to hidden Markov models (HMMs), as they are commonly used in speech recognition and signal processing. We begin with an introduction to the representation of meanings, the pattern to be recognized based on the surface acoustic observations in SLU. We then lay down a general pattern recognition framework of statistical SLU and review several implementations of this general framework. We describe in detail a HMM/CFG composite model, which combines the traditional knowledge-based approach with the statistical approach, alleviating the data sparseness problem in statistical learning. The model not only reduces the authoring load but also results in lower error rates. We then discuss the implementation in one- or two-pass ASR/SLU systems and the issues involved. For completeness in covering SLU, we also summarize the related research work that does not directly fall into the general framework discussed in this article. For the readers with a background in ASR, "An Analogous View of SLU and ASR Terminologies" draws analogies between key ASR and SLU terminologies, facilitating comparisons and contrasts between these two closely related areas of technology. Throughout this article, we strive to use simple examples to illustrate major concepts and models in SLU and to show parallel or contrastive concepts between SLU and ASR.

SEMANTIC REPRESENTATION AND SEMANTIC FRAME

What is the goal of SLU? How can we tell whether a system's understanding is appropriate or not? Ultimately, the appropriateness of the understanding can be measured by the system's responses or by the actions taken by the system after it "understands" an input utterance. In the ATIS domain, for example, this was measured by the accuracy of the flight information returned from a system after a spoken query was made by a user. However, generating the final flight information results involves more than just SLU. For better engineering practice and scientific studies, it is desirable to modularize the end-to-end system and to isolate the SLU module. For this purpose, an intermediate semantic

AN ANALOGOUS VIEW OF SLU AND ASR TERMINOLOGIES

A list of key concepts in SLU and their corresponding counterparts in ASR is provided in the following table. The correspondence is based on their respective roles in the common general framework described in the text. The analogies may not be interpreted literally in other contexts, but we have strived to highlight their commonality to our best ability. And where differences arise, we provide comments on the third column.

SLU	ASR	COMMENTS
SEMANTIC PRIOR MODEL	LANGUAGE MODEL	BOTH ARE THE PRIOR MODEL IN THE BAYESIAN PATTERN RECOGNITION FRAMEWORK. SEMANTIC PRIOR MODELS REGULATE THE RECOGNIZED MEANING REPRESENTATION; LANGUAGE MODELS REGULATE THE RECOGNIZED TEXT.
LEXICALIZATION MODEL	ACOUSTIC MODEL	BOTH MODEL THE GENERATIVE PROCESS FOR THE OBSERVATIONS. THE LEXICALIZATION MODEL DEPICTS THE PROCESS OF GENERATING A WORD SEQUENCE FROM AN UNDERLYING MEANING REPRESENTATION (SEMANTICS); THE ACOUSTIC MODEL DELINEATES THE PROCESS OF GENERATING THE ACOUSTIC FEATURE VECTOR SEQUENCES FROM AN UNDERLYING WORD (SEQUENCE).
CONTEXT-DEPENDENT LEXICALIZATION MODEL	SEGMENT MODEL	BOTH ARE SPECIFIC TYPES OF THE OBSERVATION GENERATION MODEL THAT CAPTURE THE DEPENDENCY AMONG THE OBSERVATIONS. WHILE THE SEGMENT MODELS OFTEN REPRESENT THE SEGMENT LENGTH EXPLICITLY, THE CONTEXT-DEPENDENT LEXICALIZATION MODEL CHARACTERIZES THE DURATION IMPLICITLY ACCORDING TO THE EXPECTED PROBABILITY OF OBSERVING THE SENTENCE END SYMBOL "</S>" IN AN *n*-GRAM MODEL.
SEMANTIC SCHEMA	DICTIONARY	BOTH CONTAIN A LIST OF ALL LEGITIMATE ITEMS TO BE RECOGNIZED. SEMANTIC SCHEMA CONTAINS A LIST OF SEMANTIC FRAMES; DICTIONARY CONTAINS A LIST OF PRONUNCIATIONS.
SEMANTIC FRAME	PRONUNCIATION	BOTH DEFINE THE LEGITIMATE COMPOSITION OF THE ITEMS TO BE RECOGNIZED. WHILE PRONUNCIATION GENERALLY USES LINEAR PHONEME OR ALLOPHONE SEQUENCE(S) TO MODEL WORDS, SEMANTIC FRAMES TYPICALLY MODEL THE MEANING HIERARCHICALLY. AND THERE IS GREATER FLEXIBILITY IN THE TEMPORAL ORDERING OF THE CONSTITUENTS IN THE SEMANTIC FRAME (SLOTS AND PHRASES) THAN THE CONSTITUENTS IN THE PRONUNCIATION (PHONES).
TYPED SLOT	PHONEME	BOTH ARE THE UNITS IN CONSTRUCTING THE ITEMS TO BE RECOGNIZED. SLOTS ARE THE BUILDING BLOCKS OF THE SEMANTIC FRAMES; PHONEMES ARE THE BUILDING BLOCKS OF WORD PRONUNCIATIONS. THE DIFFERENCE IS THAT A SLOT MAY HAVE EMBEDDED STRUCTURES.
SLOT FILLER	ALLOPHONE	BOTH ARE THE DIFFERENT REALIZATIONS OF THE ABSTRACT BUILDING UNITS. SLOT FILLERS ARE THE INSTANTIATIONS OF THE SLOTS, WHILE ALLOPHONES ARE THE PHONETIC VARIANTS OF A PHONEME.
PARSER	DECODER	THE APPARATUS THAT TAKES THE OBSERVATION AS INPUT AND MAPS IT TO THE RECOGNIZED ITEM BY USING THE MODELS.
MEANING REPRESENTATION	RECOGNIZED WORDS	THE TARGET OF PATTERN RECOGNITION.

representation is introduced to serve as the interface between different modules. Most spoken language systems have their own semantic representations. However, all of them can be abstracted as the semantic frame-based representation, which we now introduce.

The semantic structure of an application domain is defined in terms of *semantic frames*. Figure 2 shows a simplified example of three semantic frames for the ATIS domain. Each frame contains several typed components called *slots*. The type of a slot specifies what kind of fillers it is expecting. For example, the **subject** slot of the *ShowFlight* frame can be filled with the semantic terminal FLIGHT (expressed by words like "flight" and "flights") or the FARE semantic terminal (expressed by words like "fare" and "cost"), which specify what particular information a user needs.

The meaning of an input sentence is an instantiation of the semantic frames. Figure 3 shows the meaning representation for the sentence "Show me flights from Seattle to Boston." Here the frame *ShowFlight* contains the subframe *Flight*. Some SLU systems do not allow any substructures in a frame. In such a case, the semantic representation is simplified as a list of *attribute-value pairs*, which are also called *keyword pairs* [12] or *flat concept* representation (Figure 4).

The hierarchical representation is more expressive and enables the sharing of substructures. For example, the *Flight* frame in Figure 2 can be shared by both *ShowFlight* and *CancelFlight* (not shown) frames. The flat concept representation is simpler and often results in a simpler statistical model.

```
<frame name="ShowFlight" type="Void">
        <slot name="subject" type="Subject"/>
        <slot name="flight" type="Flight"/>
</frame>
<frame name="GroundTrans" type="Void">
        <slot name="city" type="City"/>
        <slot name="type" type="TransType"/>
</frame>
<frame name="Flight" type="Flight">
        <slot name ="DCity" type="City"/>
        <slot name="ACity" type="City"/>
        <slot name="DDate" type="Date"/>
</frame>
```

[FIG2] This figure illustrates three (simplified) semantic frames defined in the ATIS domain. Here DCity stands for "departure city" and ACity stands for "arrival city." These two slots require objects with "City" type as their fillers, which can be, for example, a city name or a city code. The *Flight* frame has the type "Flight," so it can be the filler of the *flight* slot of the top-level frame *ShowFlight.* Often, the semantic frame is related to and derived from the schema of the application database.

SLU AS STATISTICAL PATTERN RECOGNITION

In the statistical modeling approach, SLU can be formalized as a pattern recognition problem. Given the word sequence W, the goal of SLU is to find the semantic representation of the meaning M that has the maximum a posteriori probability $P(M|W)$:

$$\hat{M} = \arg\max_{M} P(M|W) = \arg\max_{M} P(W|M)P(M). \qquad (1)$$

Two separate models exist in this generative framework. The *semantic prior* model $P(M)$ assigns probability to an underlying semantic structure or meaning M. The *lexicalization* model $P(W|M)$, sometimes called *lexical generation* or *realization* model [13], assigns probability to the surface sentence (i.e., word/lexical sequence) W given the semantic structure. As an example, the HMM tagging model is a simple-minded implementation of (1), in which a Markov chain consisting of the states that bear semantic meanings models the semantic prior and the emission from the states models the lexicalization process. The alignment between the observations (words) and the states is hidden (Figure 5). The tagging model finds the Viterbi alignment between the states and the words, and the meaning associated with a state becomes the semantic tag of the aligned word.

In this simple tagging model, observations (words) depend only on the states. They do not depend on the words in the context. This independence assumption does not work well with language; according to this assumption, "Show me flights" and "me Show flights" are equally likely. Most statistical SLU systems attempt to overcome this problem by allowing a state to emit one "segment" of multiple observations (words) at a time. In this case, the generative process for the observations is

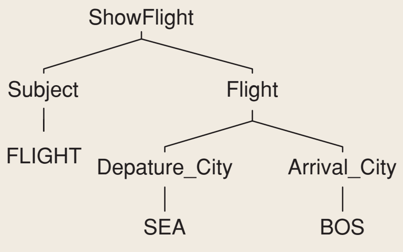
- Generate a set of segments $S = (s_1, \ldots, s_n)$ according to the semantic structure $M = q_1, \ldots, q_m$.
- Determine the alignment $A = (a_1, \ldots, a_n)$ that associates each segment with a state.
- Determine the length $L = (l_1, \ldots, l_n)$ for each segment.
- Generate l_i words for each segment s_i, for $i = 1, \ldots, n$, assuming the words are generally correlated and are sensitive to temporal ordering.

Since S, A, and L are not observed, they are hidden variables that can be marginalized out according to

$$\begin{aligned} P(W|M) &= \sum_{S,A,L} P(W, S, A, L|M) \\ &= \sum_{S,A,L} P(S|M)P(A|M, S)P(L|A, M, S) \\ &\quad \times P(W|L, M, S, A) \qquad (2) \end{aligned}$$

This type of lexicalization modeling for correlated word observation sequences is analogous to the segment model in acoustic modeling for ASR [14]. One main difference, however, is that in SLU the alignment process is more elaborated than that in ASR. This complexity arises from the syntactic constraints that allow generation of multiple phrases from a single state and placement of the phrases in an order that is far more flexible

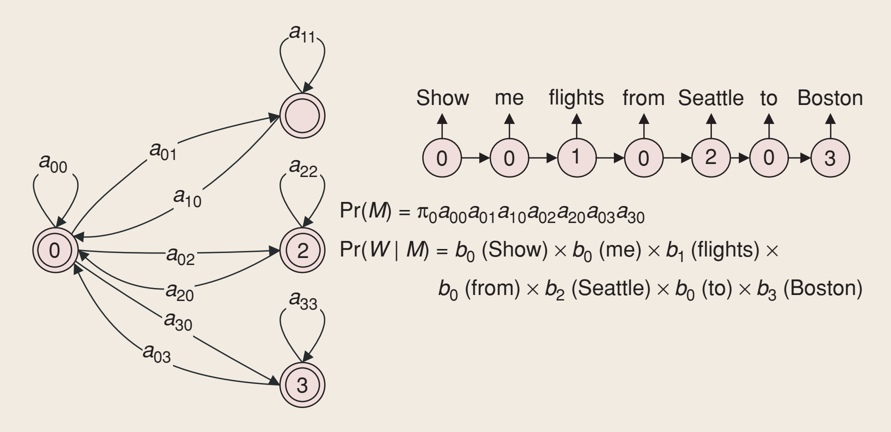
```
<ShowFlight type="Void">
  <subject type="Subject">FLIGHT</subject>
  <flight frame="Flight" type="Flight">
     <DCity type="City>SEA</DCity>
     <ACity type="City>BOS</ACity>
  </flight>
</ShowFlight>
```



[FIG3] The semantic representation for "Show me flights from Seattle to Boston" is an instantiation of the semantic frames in Figure 2. On the right is its tree representation. The instantiation picks a frame that represents the meaning conveyed in the sentence and fills slots accordingly.

(Command DISPLAY) (Subject FLIGHT) (DCity SEA) (ACity BOS)

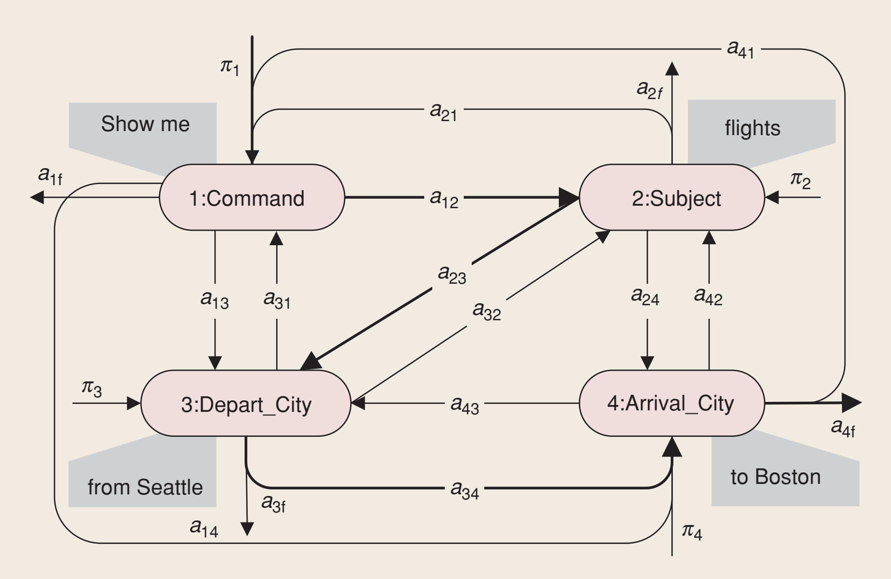
[FIG4] Attribute-value representation is a special case of the frame representation where no embedded structure is allowed. Here is an attribute-value representation for "Show me the flights from Seattle to Boston."


a_{11}
Show
me
flights
from
Seattle
to
Boston
0
0
1
0
2
0
3
a_{00}
a_{01}
a_{10}
a_{22}
0
2
a_{02}
a_{20}
a_{30}
a_{03}
a_{33}
3
$\Pr(M) = \pi_0 a_{00} a_{01} a_{10} a_{02} a_{20} a_{03} a_{30}$
$\Pr(W \mid M) = b_0$ (Show) $\times b_0$ (me) $\times b_1$ (flights) $\times$
b_0 (from) $\times b_2$ (Seattle) $\times b_0$ (to) $\times b_3$ (Boston)

[FIG5] An HMM tagging model for SLU. State 0 represents "command," state 1 represents the "subject" slot, state 2 represents the "DCity" slot, and state 3 represents the "ACity" slot. The HMM topology and the transition probabilities a_{ij} form the prior. The meaning of a sentence is represented by its underlying state sequence. The emission distributions b_k of the states form the lexicalization model. On the right is a possible state sequence that aligns to the sentence "Show me flights from Seattle to Boston." Theoretically, the state should encode the semantics of the attribute-value pair like <ACity SEA>. However, since this increases the state space significantly, sometimes infinitely, practical SLU systems collapse the states corresponding to the same attribute, and extract the value semantics from the observation aligned to the slot state in a postprocessing step.

than pronunciations in a dictionary. The latter is largely coded as a left-to-right sequence as defined by the dictionary, which is much less variable than the many different ways a fixed meaning may be expressed as a composite of phrases/words in varying orders. The purpose of alignment in Step 2 above is to account for this type of variability, which is largely absent in ASR problems.

Several SLU systems have implemented (2) differently. We will discuss this topic of lexicalization modeling in detail after first surveying semantic-prior modeling.



[FIG6] The topology of the statistical model that adopts the flat concept semantic representation. Each state represents a concept. States are fully interconnected. The initial state distribution $\pi_i = P(i)$ and the state transition distribution $a_{ij} = P(j \mid i)$ comprise the semantic prior of the model. (The final state f = "$< /s >$" is not shown in the topology.) The thicker lines illustrate the state transitions for the sentence "Show me flights from Seattle to Boston." The example sentence is also aligned to the states.

SEMANTIC PRIORS IN UNDERSTANDING MODELS

In statistical SLU modeling cross-word contextual dependency, each state represents a slot in a semantic frame. For the systems that use the flat concepts for semantic representation, such as AT&T's CHRONUS [12] and IBM's fertility model [15], the topology of the model is a fully connected network (Figure 6). Unlike the topology in Figure 5, there is no self-loop on a state because it is already modeling a segment of words with the length information already encoded.

The prior probability of the semantic structure underlying the representation in Figure 4 (i.e., the flat-concept sequence "command subject DCity ACity") can be calculated as the product of the probabilities of the marked transitions in Figure 6:

$$\begin{aligned} P(M) &= P(\mathbf{command} \mid <s>) P(\mathbf{subject} \mid \mathbf{command}) \\ &\quad P(\mathbf{ACity} \mid \mathbf{subject}) P(\mathbf{ACity} \mid \mathbf{DCity}) P(</s> \mid \mathbf{ACity}) \\ &= \pi_1 a_{12} a_{23} a_{34} a_{4\mathrm{f},} \end{aligned} \tag{3}$$

where $<s>$ represents the start of the semantic concept list and $</s>$ represents the end of the flat concept list.

For models that use hierarchical semantic structures, including BBN's hidden understanding model [13] and Microsoft Research's HMM/CFG composite model [16], the semantic prior is a natural extension of (3). Figure 7 shows the topology of the underlying states for the semantic frames in Figure 2. The left part of the diagram shows the top-level network topology and the right part shows a zoomed-in subnetwork for state 2, which represents the embedded *Flight* frame. The initial state probabilities $\pi_1 = P(ShowFlight)$ and $\pi_5 = P(GroundTrans)$ comprise the prior distribution over the top-level semantic frames. The transitional weights a_{12} and a_{13} comprise the initial slot distribution for the *ShowFlight* frame. The transitional weights a_{56} and a_{57} comprise the initial slot distribution for the *GroundTrans* frame, and the transitional weights a_{C9}, a_{CA} and a_{CB} in the subnetwork comprise the initial slot distribution for the *Flight* frame.

Given this topology, the semantic prior for the semantic structure underlying the meaning representation in Figure 3 is the product of the Markov transitional probabilities across the different levels in the semantic hierarchy (the marked path in Figure 7):

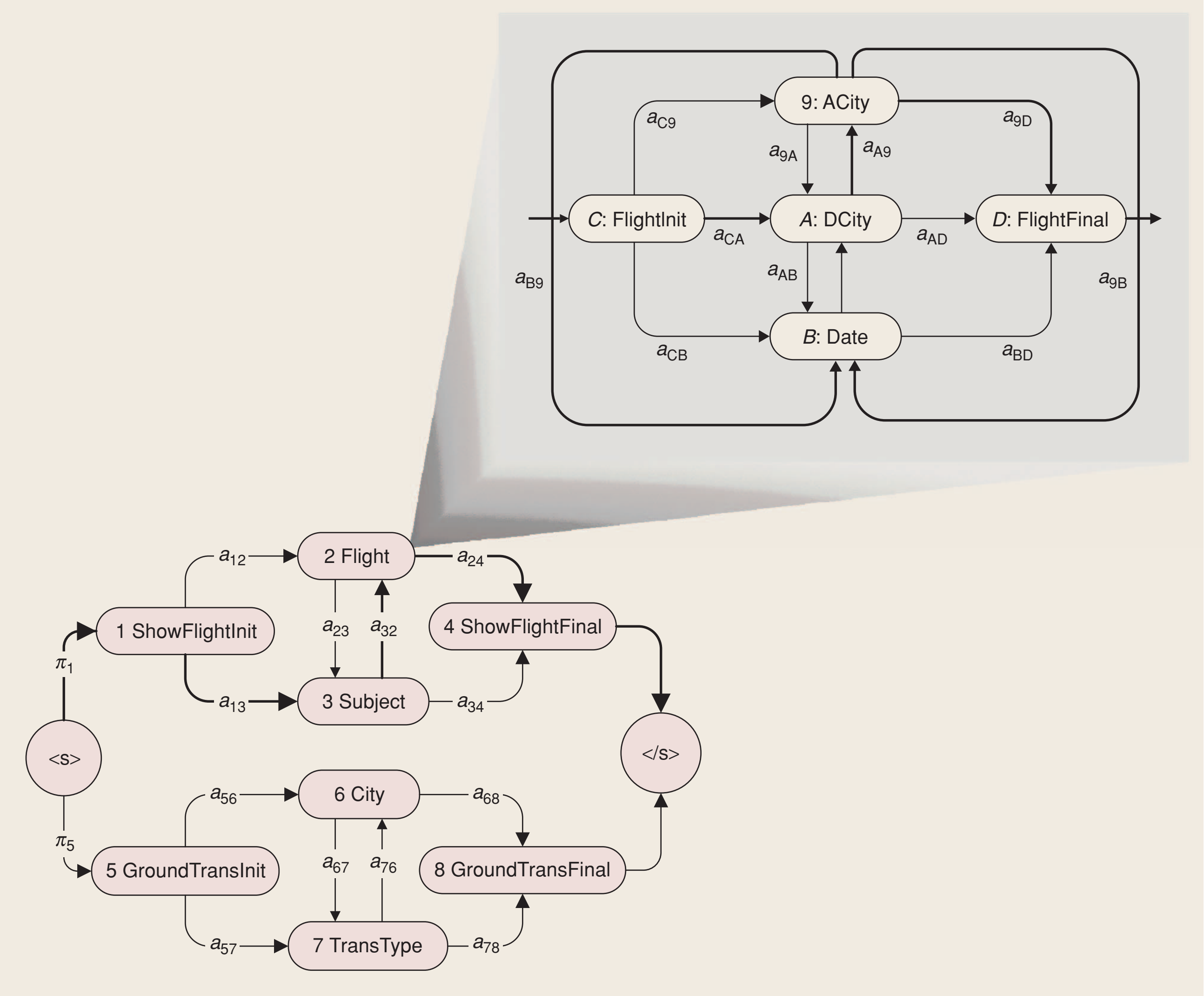
$$\begin{aligned} P(M) = {} & P(\mathit{ShowFlight})P(\mathbf{subject} \mid <\mathrm{s}>; \mathit{ShowFlight}) \\ & P(\mathbf{Flight} \mid \mathbf{subject};\ \mathit{ShowFlight}) \\ & P(\mathbf{DCity} \mid <\mathrm{s}>; \mathit{Flight}) P(\mathbf{ACity} \mid \mathbf{DCity}; \mathit{Flight}) \\ & P(</\mathrm{s}> \mid \mathbf{ACity}; \mathit{Flight}) P(</\mathrm{s}> \mid \mathbf{Flight}; \mathit{ShowFlight}) \\ = {} & \pi_1 a_{13} a_{32} a_{CA} a_{A9} a_{9D} a_{24}. \end{aligned} \qquad (4)$$

Generally,

$$P(M) = \prod_{i=1}^{|M|+1} P(C_M(i)|C_M(i-1)P(M(i))). \qquad (5)$$

Here $|M|$ is the number of the instantiated slots in M; $C_M(i)$ is the name of the ith slot in M ($C_M(0) = <\mathrm{s}>$ and $C_M(|M|+1) = </\mathrm{s}>$ stand for the beginning and the end of a frame, respectively); and $M(i)$ is the substructure that fills the ith slot in M. (5) recursively calculates the prior probabilities of the substructures and includes them in the prior probability of the parent semantic structure.

Cambridge University's hidden vector state model [17] shows another way to model the semantic prior with hierarchical structures. Named the hidden vector states, the states in the Markov chain represent the stack status of the preterminal nodes (the nodes immediately above the terminal words) in a semantic tree, as illustrated in Figure 8. The hidden vector



[FIG7] Hierarchical model topology for the semantic frames in Figure 2. On the left is the top-level network structure. On the right is the subnetwork of the semantic frame *Flight*. Since the semantic frames are defined in a finite state language, the subnetwork can substitute for state 2. Substitutions by the same subnetwork may share the parameters of the subnetwork.

states encode all the structure information about the tree, so the semantic tree structure (without the terminal words) can be reconstructed from the hidden vector state sequence. The model imposes a hard limit on the maximum depth of the stack, so the number of states becomes finite and the prior model becomes the Markov chain in an HMM. The difference from the earlier examples in Figure 6 and Figure 7 is that the state transition in this Markov chain is now modeled by the stack operations that transform one stack vector state to another. The stack operations include: 1) Pop(n), which pops the top n elements from the stack; and 2) Push(C), which pushes a new concept C into the stack. For example, the transition from the state represented by the fifth block in Figure 8 can be made with two operations: Pop(2) and Push(ON). The pop operation depends on the stack contents of the state from which the transition exits, and the push operation depends on the stack contents right before the element is pushed in. Hence, the probability of the transition from the fifth block is

$$\begin{aligned} & P_{\text{pop}}(2|<\text{CITY TOLOC RETURN SS}>) \\ & \times P_{\text{push}}(\text{ON}\,|<\text{RETURN SS}>). \qquad (6) \end{aligned}$$

SPOKEN LANGUAGE UNDERSTANDING IS AIMED AT INTERPRETING USER'S INTENTIONS FROM THEIR SPEECH UTTERANCES.

Let $|S|$ denote the depth of the stack S. Given the stack vector states S_{t-1} and S_t, $|S_{t-1}| - |S_t| + 1$ elements have to be popped from S_{t-1} to transform S_{t-1} to S_t. Thus, the semantic prior is the product of the transition probabilities in the Markov chain

$$\begin{aligned} P(M) &= \prod_{t=1}^{|M|} P(S_t|S_{t-1}) \\ &= \prod_{t-1}^{|M|} P_{\text{pop}}(|S_{t-1}| - |S_t| + 1|S_{t-1}) \\ &\quad \times P_{\text{push}}(\text{TOP}[S_t]|\text{POP}[1, S_t]) \\ &\quad \times \delta(\text{POP}[|S_{t-1}| - |S_t| + 1, S_{t-1}], \text{POP}[1, S_t]). \qquad (7) \end{aligned}$$

Here $|M|$ is the number of stack vector states in M, including the sentence end state $S_{|M|}$; S_0 is the sentence initial state; TOP[S] is the top element of the stack S; and POP[n, S] is the new stack after the top n elements are popped out of the stack S. $\delta(x, y) = 1$ if $x = y$, otherwise $\delta(x, y) = 0$. This guarantees that $S_{t-1} \rightarrow S_t$ is a legal transition by demanding that all the elements in the two stacks are the same, except for those popped out of S_{t-1} and the one pushed into S_t.

The decomposition of the transition probability into P_{pop} and P_{push} enables different transitions to share the parameters. Therefore, it can potentially reduce the number of parameters in the prior model.

It is important to note that the prior model does not have to be static. It can change depending on the context. For example, at a dialog state when the system asks users for the departure city information, the probability for the DCity slot can be significantly boosted. The prior model can also be personalized [18].

LEXICALIZATION MODELS

In this section we review the lexicalization models under the general framework of (2), which captures dependency and temporal ordering for the observed words within the same state.

N-GRAM LEXICALIZATION MODEL

The first lexicalization model, used by both CHRONUS [12] and the hidden understanding model [13], assumes a deterministic one-to-one correspondence between model states and the segments, i.e., there is only one segment per state, and the order of the segments follows the order of the states. This effectively gets rid of the hidden variables S and A in (2):

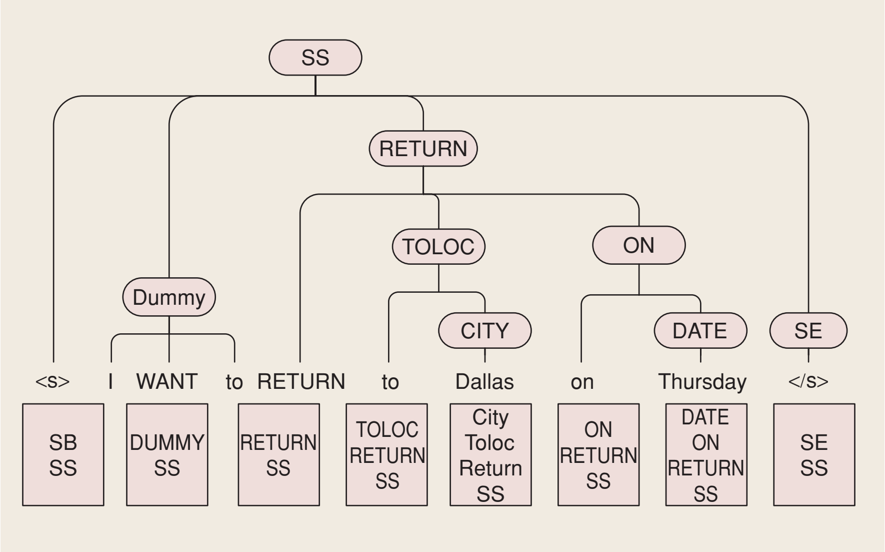
$$\begin{aligned} P(W|M) &= \sum_{L} P(W, L|q_1, \ldots, q_m) \\ &= \sum_{\pi=\varphi_1,\ldots,\varphi_m} P(\pi\,|q_1, \ldots, q_m) \\ &\approx \sum_{\pi=\varphi_1,\ldots,\varphi_m} \prod_{i-1}^{m} P(\varphi_i|q_i). \qquad (8) \end{aligned}$$

Here the joint event (W, L) corresponds to a partition $\pi = \varphi_1, \ldots, \varphi_m$ of W, where φ_1 is the first l_1 words of W, φ_2 is the next l_2 words of W, and the concatenation of the substrings $\varphi_1, \ldots, \varphi_m$ equals W.

Both CHRONUS [12] and the hidden understanding model [13] exploited state-specific n-grams to model $P(\varphi|q)$. Let's use the flat-concept semantic structure in Figure 6 as an example. It illustrates a partition $\pi =$ show me, flights, from Seattle, to Boston. The probability of the surface sentence under this partition is

$$\begin{aligned} & P(\pi = \text{Showme, flights, fromSeattle, toBoston}| \\ & \qquad M = \textbf{command, subject, DCity, ACity}) \\ & \quad = P(\text{Show}\,|<\text{s}>;\ \textbf{command})P(\text{me}\,|\,\text{Show};\ \textbf{command}) \\ & \qquad \times P(</\text{s}>|\text{me};\ \textbf{command}) \\ & \qquad \times P(\text{flights}\,|<\text{s}>;\ \textbf{subject})P(</\text{s}>|\text{flight};\ \textbf{subject}) \\ & \qquad \times P(\text{from}\,|<\text{s}>;\ \textbf{DCity})P(\text{Seattle}\,|\,\text{from};\ \textbf{DCity}) \\ & \qquad \times P(</\text{s}>|\text{Seattle};\ \textbf{DCity}) \\ & \qquad \times P(\text{to}\,|<\text{s}>;\ \textbf{ACity})P(\text{Boston}\,|\,\text{to};\ \textbf{ACity}) \\ & \qquad \times P(</\text{s}>|\text{Boston};\ \textbf{ACity}). \qquad (9) \end{aligned}$$

In (9), cross-state lexical dependency is not modeled. A word at a state only depends on the history that belongs to the same state (or on the context cue "<s>" if it is the first word emitted from a state). One can opt to model the cross-state lexical dependency as well. In this case, instead of depending on "<s>," the initial words from a state may depend on the last few words (actual number determined by the n-gram order) from the previous state.



[FIG8] In the hidden vector state model, the states (illustrated by the blocks) represent the stack status of the preterminal nodes (the parent node of the terminal words) in the semantic tree.

FERTILITY LEXICALIZATION MODEL

IBM's *fertility model* [15] is another implementation of (2). It is based on their statistical machine translation work. Similar models were proposed in [19]. The generative process of the model can be illustrated by an example. Let the semantic structure be the flat sequence of four states, "DISPLAY FLIGHTS TO ARRIVAL_CITY." The model first picks the number of segments for each state, for example, (2, 1, 1, 1). This results in five segments, which are permutated to form the ordered sequence $S =$ <SEG$_{\text{DISPLAY}}$, SEG$_{\text{FLIGHTS}}$, SEG$_{\text{TO}}$, SEG$_{\text{ARRIVAL\_CITY}}$, SEG$_{\text{DISPLAY}}$>. Each permutation corresponds to an alignment. The permutation in the example corresponds to the alignment $A = (1\ 2\ 3\ 4\ 1)$, where each element points to the state that gives rise to the segment. Since there are (5!/2!1!1!1!) different permutations, each possible alignment has the uniform probability 2!/5!. The model then picks the length in each segment, (2, 2, 1, 1, 1), and accordingly generates two words "I want" for the first segment, "to fly" for the second segment, "to" for the third segment, "Boston" for the fourth segment, and "please" for the last segment. This produces the final surface sentence "I want to fly to Boston please." As illustrated by this example, a state in this model can emit nonconsecutive word sequences.

The fertility model makes the following assumptions in the aforementioned process:

1) $P(S|M=q_1,\dots,q_m)\approx\prod_{i=1}^{m} f(n_i|q_i)=\prod_{i=1}^{m} e^{-\lambda_{q_i}}\lambda_{q_i}^{n_i}/n_i!$; i.e., each state q_i generates n_i segments according to a Poisson distribution called the *fertility model*. Here $m = |M|$ is the number of states in M and $n = \sum_{i=1}^{m} n_i = |S|$ is the total number of segments generated from the m states.

2) The alignment model $P(A = a_1, \dots, a_n|M = q_1, \dots, q_m, S = s_1, \dots, s_n)$ follows a uniform distribution. Here a_j is the index of the state in M to which the jth segment is aligned. In other words, according to A, segment s_j is aligned to the state q_{a_j}.

3) $P(L = l_1, \dots, l_n \mid A = a_1, \dots, a_n, M = q_1, \dots, q_m, S = s_1, \dots, s_n) \approx \prod_{j=1}^{n} n(l_j \mid q_{a_j})$, i.e., the length of a segment only depends on the state it is aligned to.

4) $P(W \mid L, M, S, A) \approx \prod_{j=1}^{n}\prod_{k=1}^{l_j} P(w_{jk} \mid q_{a_j})$; i.e., each word in a segment is generated with the dependency on the state to which the segment is aligned. Here w_{jk} is the kth word in segment s_j, which has length l_j.

Given the above assumptions,

$$
\begin{aligned}
&P(W,S,A,L \mid M)\\
&\quad = P(S \mid M)P(A \mid M,S)\ P(L \mid M,S,A)P(W \mid M,S,A,L) \quad (10)\\
&\quad = \left(\prod_{i=1}^{|M|} \frac{e^{-\lambda_{q_i}}\lambda_{q_i}^{n_i}}{n_i\,!}\right) \times \left(\frac{1}{n\,!}\prod_{i=1}^{|M|} n_i\,!\right) \times \left(\prod_{j=1}^{n} n\left(l_j \mid q_{a_j}\right)\right)\\
&\qquad \times \left(\prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right)\right) \quad (11)\\
&\quad = \frac{1}{n\,!}\prod_{i=1}^{|M|} e^{-\lambda_{q_i}}\lambda_{q_i}^{n_i}\prod_{j=1}^{n} n\left(l_j \mid q_{a_j}\right)\prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right) \quad (12)\\
&\quad = \frac{1}{n\,!}\prod_{i=1}^{|M|} e^{-\lambda_{q_i}}\prod_{j=1}^{n} \lambda_{q_{a_j}} n\left(l_j \mid q_{a_j}\right)\prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right) \quad (13)\\
&\quad = \frac{1}{n\,!}\prod_{i=1}^{|M|} e^{-\lambda_{q_i}}\prod_{j=1}^{n} r\left(l_j, s_j \mid q_{a_j}\right). \quad (14)
\end{aligned}
$$

In (11), the second factor is the permutation (alignment) probability, which is the inverse of the multinomial coefficient. Equation (13) distributes the λs inside the second product. In (14), $r(l, s \mid q)\overset{\Delta}{=}\lambda_q n(l \mid q)\prod_{k=1}^{l} p(w_{jk} \mid q)$ is proportional to the probability that a segment $s = w_{j1}, \dots, w_{jl}$ is generated from the state q. In this form, one can sum over all possible alignments A in polynomial time

$$\begin{aligned}P(W,S,L\mid M) &= \sum_{A} P(W,S,A,L\mid M)\\ &= \sum_{A}\frac{1}{n!}\prod_{i=1}^{|M|} e^{-\lambda_{q_i}}\prod_{j=1}^{n} r\left(l_j, s_j \mid q_{a_j}\right)\\ &= \frac{1}{n!}\prod_{i=1}^{|M|} e^{-\lambda_{q_i}}\prod_{j=1}^{n}\sum_{q\in M} r\left(l_j, s_j \mid q\right). \qquad (15)\end{aligned}$$

The last step above can be shown by algebraic operations.

The expectation-maximization (EM) algorithm is used to estimate the parameters in the model, and a dynamic programming algorithm is applied to search for the meaning according to

$$\begin{aligned}\hat{M} &= \arg\max_{M} P(W\mid M)P(M)\\ &= \arg\max_{M}\sum_{S,L} P(W,S,L\mid M)P(M) \qquad (16)\end{aligned}$$

with marginalization over all possible segment sequences S and the length for the segments, L.

MODEL TRAINING

Maximum likelihood (ML) estimation can be used to estimate parameters in both the semantic prior model and the lexicalization model. In supervised training, if each word is labeled with the state it belongs to, as in CHRONUS in [12] and the hidden understanding model in [13], then both Markov transition probabilities and the state-conditioned n-gram models can be directly estimated from the data by simply counting the relative frequencies. We will demonstrate how the ML estimation is applied when only strictly semantic partial annotation is available. In such a case, many word/state alignments are hidden and the EM algorithm is used to estimate the parameters.

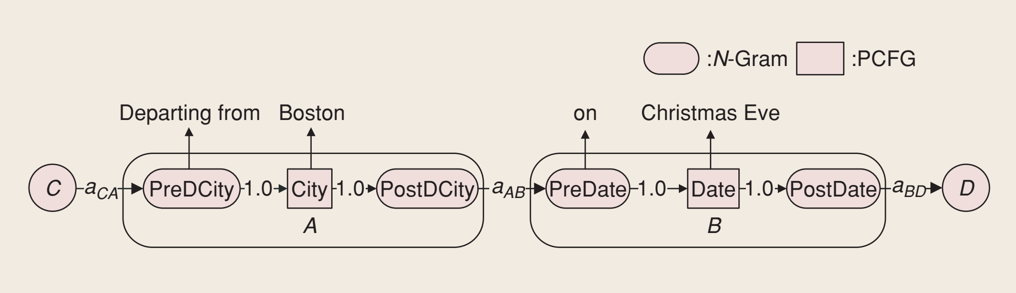
STATISTICAL LEARNING ASSISTED BY DOMAIN/LINGUISTIC KNOWLEDGE

One disadvantage of purely data-driven, statistical SLU is the requirement of a large amount of training data. To overcome this problem, many systems utilize a preprocessing step to identify the "superwords" from the input stream with pattern matching. This step includes: 1) replacing the words in a semantic class with the class name, e.g., "Seattle" and "Boston" are replaced with the superword "_cityname"; and 2) replacing a word sequence that matches a regular expression with a superword, e.g., "one hundred twenty five" is replaced with the superword "_number."

Two problems arise, however, with this traditional solution. First, as it will be discussed in the section titled "Use of Understanding Models for Speech Recognition," it is beneficial to use the SLU model as the language model for speech recognition. However, with actual words being replaced by the superwords that are often modeled with complicated CFGs instead of word lists as in class-based language models [20], the model can no longer be used for speech recognition. Second, the "superword" solution does not handle ambiguities gracefully. Although a sophisticated preprocessor can produce a lattice that includes ambiguous tagging of the superwords, they are not fairly evaluated by the understanding model. For example, in "Schedule a meeting tomorrow ten to eleven," the phrase "ten to eleven" may be ambiguously tagged as "_time" as in the interpretation "10:50" or "_time to _time" as in the interpretation "from 10:00 to 11:00." Since the phrase is treated as one superword in the first interpretation but as three words in the second interpretation, only one transition and one emission probability need to be applied for the first interpretation, while multiple transition and emission probabilities have to be applied for the second one. Therefore, the SLU model will be biased towards the first interpretation.

HMM/CFG COMPOSITE MODEL

Both problems mentioned above can be attributed to the fact that the preprocessing step is not modeled statistically as an integral part of the SLU model. The lack of information about the preprocessing model makes the statistical SLU model unable to predicate the words for speech recognition. It also prohibits the model from properly normalizing the probabilities because the actual length of the segment replaced by the superword is unknown to the SLU model. An HMM/CFG composite model has been introduced in [16], which we review here. This model uses the same semantic prior for hierarchical Markov topology as in (5). The underlying state corresponding to a slot in the semantic frame is expanded into a preamble-filler-postamble, three-state sequence (Figure 9). The preamble and postamble serve as the contextual clue for the identity of the slot, while the slot filler decides its value. The lexicalization model follows (8), similar to CHRONUS and the hidden understanding model. The difference here is that it uses either n-gram models or probabilistic context-free grammars (PCFGs) for $P(\varphi\mid q)$. If q is a state corresponding to a slot filler, a PCFG is used for $P(\varphi\mid q)$. The PCFG embeds domain-specific and domain-independent knowledge, like a city



[FIG9] State alignment for the phrase "departing from Boston on Christmas Eve" according to the topology in Figure 7. The original states A and B are expanded to three states.

name list and a date grammar. The CFG rules can be populated with database entries or prebuilt in a grammar library for domain-independent concepts (e.g., date and time). The lexicalization of other states is modeled with n-grams. Figure 9 shows the state alignment for an example phrase according to the network topology in Figure 7. Note the introduction of the preamble and postamble states does not change the semantic prior model because the transition probabilities from the preambles and to the postambles are always 1.0.

Formally, the lexicalization probability in this composite model is

$$\begin{aligned} P(W \mid M) &= \sum_{\pi=\varphi_1,\ldots,\varphi_m} P(\pi \mid q_1, \ldots, q_m) \\ &= \sum_{\pi=\varphi_1,\ldots,\varphi_m} \prod_{i=1}^{m} P_{q_i}(\varphi_i \mid q_i). \end{aligned} \tag{17}$$

Here P_q is an n-gram model if q is a preamble or a postamble state; it is a PCFG if q is a slot filler. It is possible that for some $q \neq r$, $P_q = P_r$. For example, the PCFG for "cityname" are shared by the fillers for the DCity and ACity slots.

PARAMETER ESTIMATION FOR HMM/CFG MODEL

The composite model can be formalized as (Σ, A, G), where Σ is a finite set of states, A is the state transition probability distributions, and G is a set of emission grammars associated with the states. Σ is determined by the semantic frame. Parameters of A and the n-gram parts of G have to be estimated from the training data. Figure 10 illustrates the process of estimating those parameters. The estimation requires manually annotated data, an example of which is shown in step 2. To make the annotation simple and easy to perform, only semantic-level information (i.e., frames and slot fillers) is marked. Given an annotation, its state sequence is fixed in the model topology (the path denoted by thick arrow lines in step 3). The counts of state transitions can be collected from the state sequence, and ML estimation is applied for the state transition probabilities (step 4). Here a prior count c is used to smooth the transition distributions. The parameter can be optimized with held-out data. To estimate the lexicalization parameters, one possibility is to extend the forward-backward algorithm used in discrete HMM training [21]. Note that in discrete HMM training, the posterior

$$\gamma_t(q) = P_\phi(q_t = q \mid W) = \frac{\alpha_t(q)\beta_t(q)}{\sum_{q'} \alpha_t(q')\beta_t(q')} \tag{18}$$

can be calculated with the forward-backward algorithm. And the emission probability can be then estimated by

$$b_q(w) = \sum_{t=1, s,t, w_t=w}^{T} \gamma_t(q) \Big/ \sum_{t=1}^{T} \gamma_t(q). \tag{19}$$

When a segment of words can be generated from a single state and n-gram is used to model this generation process, the forward-backward algorithm can be extended. Using bigram as an example, the posterior

$$\begin{aligned} \xi_t(q, r) &= P(q_{t-1} = q, q_t = r \mid W) \\ &= \frac{\alpha_{t-1}(q) t(q, r, w_{t-1}) P_r(w_t \mid h(q, r, w_{t-1}))\beta_t(r)}{\sum_{q'} \sum_{r'} \alpha_{t-1}(q') t(q', r', w_{t-1}) P_{r'}(w_t \mid h(q', r', w_{t-1}))\beta_t(r')} \end{aligned}$$

$$h(q, r, w) = \begin{cases} <\mathrm{s}> & \text{when } q \neq r \\ w & \text{when } q = r \end{cases}$$

$$t(q, r, w) = \begin{cases} P_q(</\mathrm{s}> \mid w) a_{qr} & \text{when } q \neq r \\ 1 & \text{when } q = r \end{cases} \tag{20}$$

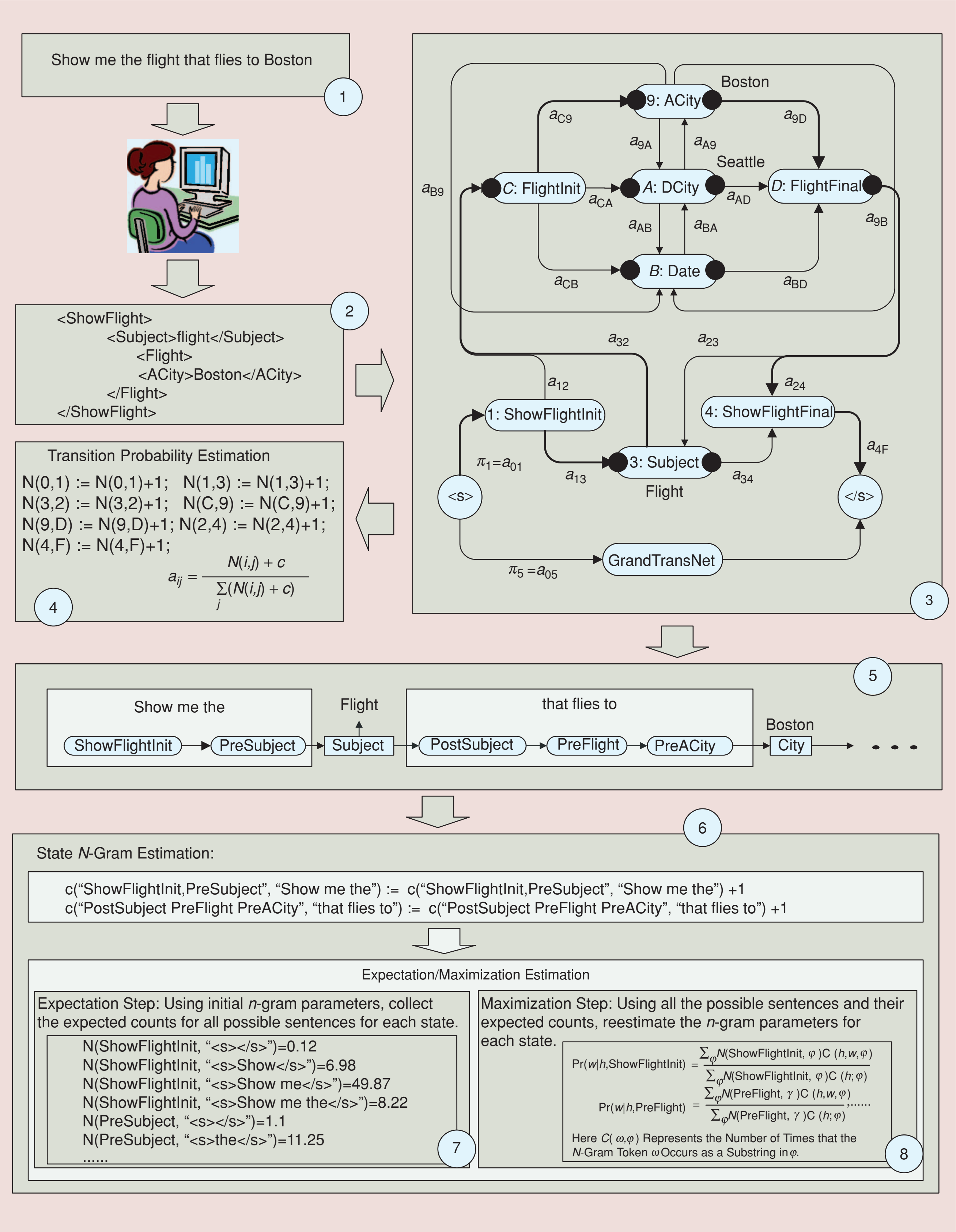
can also be calculated with the forward-backward algorithm. Here, there are several differences from the standard HMM training. First, the emission probability now depends on $h(q, r, w_{t-1})$, the history of the previous word from the same state or the context cue "<s>" for the initial word in a segment. Second, in the segment model, there is no self-loop transition at a state. The generative process stays at the same state unless the end of the segment is predicted by the bigram model for the state. This segment ending probability must be included in the transition probability, $t(q, r, w_{t-1})$, when a state transition is made. The computation of the forward and backward probabilities, $\alpha_t(q)$ and $\beta_t(q)$, should be changed accordingly. With the posterior $\xi_t(q, r)$ defined, the bigram probability can be obtained by

$$P_q(v|w) = \sum_{t=2, s.t. w_{t-1}=w, w_t=v}^{T} \xi_t(q, q) \Big/ \sum_{t=2, s.t. w_t=v}^{T} \xi_t(q, q) \tag{21}$$

for $w \neq <\mathrm{s}>$, and

$$\begin{aligned} P_q(v \mid <\mathrm{s}>) = &\left(\gamma_1(q)\delta(w_1, v) + \sum_{r \neq q} \sum_{t=2, s.t. W_t=v}^{T} \xi_t(r, q)\right) \\ &\Big/ \left(\gamma_1(q) + \sum_{r \neq q} \sum_{t=2}^{T} \xi_t(r, q)\right). \end{aligned} \tag{22}$$

This solution is complicated to implement, given all the boundary conditions in $h(q, r, w_{t-1})$ and $t(q, r, w)$, especially with higher order n-grams. One simpler solution, given the fact that many n-gram training and smoothing implementations are already available, is to obtain all the strings φ that state q can generate, obtain the count $N(q, \varphi)$ (i.e., the number of times that q generates φ), and then compute the bigram probability with the standard ML estimation:



Show me the flight that flies to Boston

```
<ShowFlight>
        <Subject>flight</Subject>
            <Flight>
             <ACity>Boston</ACity>
        </Flight>
</ShowFlight>
```

Transition Probability Estimation

N(0,1) := N(0,1)+1;   N(1,3) := N(1,3)+1;
N(3,2) := N(3,2)+1;   N(C,9) := N(C,9)+1;
N(9,D) := N(9,D)+1; N(2,4) := N(2,4)+1;
N(4,F) := N(4,F)+1;

$$a_{ij}=\frac{N(i,j)+c}{\sum_j(N(i,j)+c)}$$

State *N*-Gram Estimation:

c("ShowFlightInit,PreSubject", "Show me the") := c("ShowFlightInit,PreSubject", "Show me the") +1
c("PostSubject PreFlight PreACity", "that flies to") := c("PostSubject PreFlight PreACity", "that flies to") +1

Expectation/Maximization Estimation

Expectation Step: Using initial *n*-gram parameters, collect the expected counts for all possible sentences for each state.

N(ShowFlightInit, "<s></s>")=0.12
N(ShowFlightInit, "<s>Show</s>")=6.98
N(ShowFlightInit, "<s>Show me</s>")=49.87
N(ShowFlightInit, "<s>Show me the</s>")=8.22
N(PreSubject, "<s></s>")=1.1
N(PreSubject, "<s>the</s>")=11.25
......

Maximization Step: Using all the possible sentences and their expected counts, reestimate the *n*-gram parameters for each state.

$$\Pr(w|h,\text{ShowFlightInit})=\frac{\sum_\varphi N(\text{ShowFlightInit},\varphi)C(h,w,\varphi)}{\sum_\varphi N(\text{ShowFlightInit},\varphi)C(h;\varphi)}$$

$$\Pr(w|h,\text{PreFlight})=\frac{\sum_\varphi N(\text{PreFlight},\gamma)C(h,w,\varphi)}{\sum_\varphi N(\text{PreFlight},\gamma)C(h;\varphi)},\ldots\ldots$$

Here $C(\omega,\varphi)$ Represents the Number of Times that the *N*-Gram Token ω Occurs as a Substring in φ.

[FIG10] An example giving detailed illustration of the various steps involved in estimating the transition probability and *n*-gram lexicalization probability in the HMM/CFG composite model based on partially annotated training data. See text for details.

$$P_q(w_k \mid w_{k-n+1}, \ldots, w_{k-1}) = \frac{\sum\limits_{\varphi} N(q, \varphi) C(w_{k-n+1}, \ldots, w_{k;} \varphi)}{\sum\limits_{\varphi} N(q, \varphi) C(w_{k-n+1}, \ldots, w_{k-1;} \varphi)}. \tag{23}$$

Here $C(\omega; \varphi)$ is the number of times that word sequence ω occurs in φ.

When training samples are fully annotated (i.e., every word is marked with its aligned state), $N(q, \varphi)$ can be obtained by simple counting. When only partial annotation is available, as illustrated by the example in step 2 in Figure 10, $N(q, \varphi)$ can be viewed as the expected count of the event that state q generates the string φ. Notice that the annotation pegs the slot fillers to the slot states in step 3 in Figure 10, which restricts the alignments of the remaining words and states, so a subsequence of the observation W (e.g., $W =$ "that flies to") can only align to a state subsequence Q (e.g., $Q =$ "PostSubject PreFlight PreACity"), as shown in step 5. The counts of these subsequence alignments, $c(Q, W)$, can be collected from all the annotated training examples. From these statistics, the EM algorithm is applied to estimate the n-gram parameters for each state q. In E-step (step 7), $N(q, \varphi)$ is computed with the current model parameters (initially based on the uniform distribution). In M-step (step 8), (23) is applied to estimate the n-gram probability (with proper smoothing such as deleted interpolation [22]). The calculation of $N(q, \varphi)$ is illustrated in "Derivation for the Computation of the Expected Count $N(q, \varphi)$."

DERIVATION FOR THE COMPUTATION OF THE EXPECTED COUNT $N(q, \varphi)$

The expected count for state q to generate segment φ is $N(q, \varphi) = \sum_{Q,W} c(Q, W) \sum_{\pi} P_{\phi}(\pi | Q, W) c(q \uparrow \varphi; \pi, Q, W)$. Here $c(Q, W)$ is the number of occurrences that state sequence $Q = q_1, q_2, \ldots, q_m$ co-occurs with word sequence $W = w_1, \ldots, w_k$ in training data as in step 6 in Figure 10. π is a partition that breaks W into m nonoverlapping segments, each segment corresponds (aligns) to a state in Q. The segment may be an empty string. $c(q \uparrow \varphi; \pi, Q, W)$ is the number of times that state q appears in Q and aligns to the substring φ according to the partition π. To remove the summation to obtain $N(q, \varphi)$, noting that because $P_{\phi}(\pi, Q, W) = \prod_{q,\varphi} P_{\phi}(\varphi | q)^{c(q \uparrow \varphi; \pi, Q, W)}$,

$$\begin{aligned}
\frac{\partial P_{\phi}(Q, W)}{\partial P_{\phi}(\varphi \mid q)} &= \frac{\partial \sum\limits_{\pi} P_{\phi}(\pi, Q, W)}{\partial P_{\phi}(\varphi \mid q)} \\
&= \sum_{\pi} \frac{c(q \uparrow \varphi; \pi, Q, W) \prod\limits_{q', \varphi'} P_{\phi}(\varphi' \mid q)^{c(q' \uparrow \varphi'; \pi, Q, W)}}{P_{\phi}(\varphi \mid q)} \\
&= \sum_{\pi} \frac{P_{\phi}(\pi, Q, W) c(q \uparrow \varphi; \pi, Q, W)}{P_{\phi}(\varphi \mid q)} \\
&= P_{\phi}(Q, W) \frac{\sum\limits_{\pi} P_{\phi}(\pi \mid Q, W) c(q \uparrow \varphi; \pi, Q, W)}{P_{\phi}(\varphi \mid q).}
\end{aligned} \tag{24}$$

Rearranging (24), we obtain

$$\sum_{\pi} P_{\phi}(\pi \mid Q, W) c(q \uparrow \varphi; \pi, Q, W) = \frac{P_{\phi}(\varphi \mid q)}{P_{\phi}(Q, W)} \frac{\partial P_{\phi}(Q, W)}{\partial P_{\phi}(\varphi \mid q)}. \tag{25}$$

On the left of (25) is the summation that occurs in the expected count. On the right there is no summation at all. The problem now becomes how to compute $P_{\phi}(Q, W)$ and $\partial P_{\phi}(Q, W)]/[\partial P_{\phi}(\varphi \mid q)$ efficiently. For that purpose, define $\alpha_k(i) =$ $\Pr(\pi = \varphi_1, \ldots, \varphi_m$ and $\varphi_k = \ldots, w_i; W \mid Q)$ to be the probability of all the segmentations π that align the end of q_k in to the ith word in W, and $\beta_k(i) = P(\pi = \varphi_1, \ldots, \varphi_m$ and $\varphi_k = \ldots, w_i; W \mid Q)$ to be the probability of all the segmentations that align the beginning of q_k to the ith word in W. Then

$$\begin{aligned}
P_{\phi}(Q, W) &= \alpha_m(n) P_{\phi}(Q) \\
&= P_{\phi}(Q) \sum_{ij} \alpha_{k-1}(i-1) P_{\phi}(w_i, \ldots, w_j \mid q_k) \\
&\quad \times \beta_{k+1}(j+1) \quad \text{for } \forall k.
\end{aligned} \tag{26}$$

According to (26),

$$\frac{\partial P_{\phi}(Q, W)}{\partial P_{\phi}(\varphi \mid q)} = P_{\phi}(Q) \sum_{\substack{k: q_k = q \\ ij: \varphi = w_i, \ldots, w_j}} \alpha_{k-1}(i-1) \beta_{k+1}(j+1). \tag{27}$$

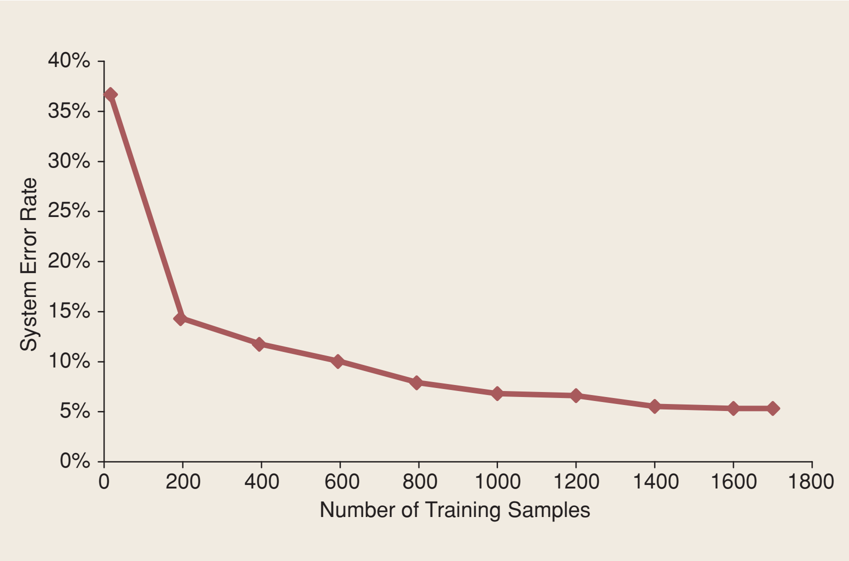
Combining (25), (26) and (27), the expected count

$$N(q, \varphi) = P_{\phi}(\varphi \mid q) \sum_{Q,W} \frac{c(Q, W)}{\alpha_{|Q|}(|W|)} \sum_{\substack{k: q_k = q \\ ij: \varphi = w_i, \ldots, w_j}} \alpha_{k-1}(i-1) \beta_{k+1}(j+1). \tag{28}$$

$\alpha_k(i)$ and $\beta_k(i)$ can be computed efficiently with dynamic programming according to (29):

$$\begin{aligned}
\alpha_0(i) &= P_{\phi}(w_1, \ldots, w_i \mid q_0); \\
\alpha_k(i) &= \sum_{r \le i} \alpha_{k-1}(r) P_{\phi}(w_{r+1}, \ldots, w_i \mid q_k); \\
\beta_m(i) &= P_{\phi}(w_i, \ldots, w_n \mid q_m); \\
\beta_k(i) &= \sum_{r \ge i} \beta_{k-1}(r) P_{\phi}(w_i, \ldots, w_{r-1} \mid q_k);
\end{aligned} \tag{29}$$

Here $P_{\phi}(w_i, \ldots, w_j \mid q)$ can be obtained according to the n-gram (initially with the uniform distribution) specific to state q. When $r = i, P_{\phi}(w_{r+1}, \ldots, w_i \mid q) = P_{\phi}(</s> \mid <s>; q)$ is the probability to generate an empty string from q.



[FIG11] ATIS end-to-end system error rate on text input versus training set size for the HMM/CFG composite model.

THE PERFORMANCE OF THE HMM/CFG COMPOSITE MODEL

The HMM/CFG composite model balances the tradeoff between robustness and the constraints on overgeneralizations/ambiguities with the different models for preambles/ postambles and slot fillers. The CFG model imposes a relatively rigid restriction on the slot fillers, which are more crucial for correct understanding and less subject to the disfluencies because they are semantically coherent units. The fillers are often domain specific and can be obtained from the application database, such as the city names and airport names in the ATIS domain. Alternatively, they are common domain-independent concepts like phone numbers, dates, and times, which are already modeled in a grammar library. Fillers can also be automatically generated according to some high-level description like a regular expression for an alphanumeric concept [23]. The nonfiller states serve as the "glue" that sticks different slot fillers together. This type of interconcept language is normally domain dependent, hard to prebuild a model for, and subject to more disfluencies. It varies significantly across different speakers. The n-gram model is more robust and thus suitable for this sublanguage. Furthermore, the knowledge introduced by the CFG sub-model greatly compensates for the data sparseness problem (e.g., it is very unlikely to see all city names occur in all contexts in training data).

[SLU IS CLOSELY RELATED TO NATURAL LANGUAGE UNDERSTANDING (NLU).]

We conducted the end-to-end experiment illustrated by Figure 1, with the manual transcription used as the first step instead of speech recognition. The vocabulary contains about 1,000 words. The database query results were compared with the reference using the standard ATIS evaluation tool. Figure 11 plots the slot error rate of the HMM/CFG composite model with the ATIS 93 category A test set (utterances whose interpretation is independent of the context), with respect to the amount of the training data it used. Here the accuracy of the model trained with half of the 1993 ATIS3 training data is close to that of the model trained with all the 1993 data (~1,700 sentences). With all the training data, the end-to-end error rate is 5.3%, which is comparable to the best manually developed grammar and better than the best data-driven statistical models that used all the ATIS2 and ATIS3 training data (over 6,000 sentences). This demonstrates that the inclusion of domain knowledge in the statistical model reduced the training requirement.

USE OF UNDERSTANDING MODELS FOR SPEECH RECOGNITION

SUBOPTIMALITY OF TWO-PASS SLU

In (1), it is assumed that the sentence or word sequence W is the observation. This is true for typed-in language. However, for SLU, the observation is the acoustic observation sequence O. Hence, the optimal meaning for a speech utterance O is

$$\hat{M} = \arg\max_{M} P(M \mid O) = \arg\max_{M} P(O \mid M)P(M). \qquad (30)$$

Often, (30) is implemented with a two-pass approach. In the first pass, a "pseudo" word observation sequence

$$\hat{W} = \arg\max_{W} P(W \mid O) = \arg\max_{W} P(O \mid W)P(W)$$

is obtained with maximum a posteriori probability by ASR. In the second pass, meaning $\hat{M}$ is extracted from $\hat{W}$ by plugging $\hat{W}$ into (1):

$$\begin{aligned}\hat{M} &= \arg\max_{M} P(M \mid \hat{W})\\ &= \arg\max_{M} P\left(M \mid \arg\max_{W}(P(O \mid W)P(W))\right)\\ &= \arg\max_{M} P(M)P\left(\arg\max_{W}(P(O \mid W)P(W))|M\right). \qquad (31)\end{aligned}$$

The two-pass solution is simpler from an engineering point of view, and it runs faster because there is no need to keep separate search paths with the same prefix strings but different semantic structures in speech decoding. However, it is a suboptimal solution because the dependency link between O and M via W is broken. Instead, O is generated via a different language model $P(W)$ and acoustic model $P(O \mid W)$ that have nothing to do with M. In an extreme case, we may have $P(W) = 0$ whenever $P(W \mid M) \neq 0$, so no optimal solution can be found at all. In research spoken dialog systems, this problem has often been heuristically addressed by performing SLU on the n-best outputs from a recognizer. Another disadvantage of the two-pass solution lies in the difficulty of adapting to changes. If new cities are added into a database or a new type of service is introduced, the language model in a two-pass system has to be retrained unless a class-based language model is used.

IDEALLY, AN SLU SYSTEM SHOULD BE ABLE TO AUTOMATICALLY ADAPT TO THE REAL DATA COLLECTED AFTER ITS DEPLOYMENT.

An alternative to this two-pass solution is

$$\begin{aligned}\hat{M} &= \arg\max_M \Pr(M \mid O) = \arg\max_M P(O \mid M)P(M)\\ &= \arg\max_M \sum_W P(O, W \mid M)P(M)\\ &= \arg\max_M \sum_W P(O \mid W, M)P(W \mid M)P(M)\\ &\approx \arg\max_M \max_W P(O \mid W)P(W \mid M)P(M). \end{aligned} \tag{32}$$

Here the understanding model $P(W \mid M)P(M)$, which preserves the dependency link between M and O, is directly used in place of the generic language model in ASR.

USING UNDERSTANDING MODEL AS THE LM FOR SPEECH RECOGNITION

For a knowledge-based system that uses CFG for SLU, it may appear to be easy to use the SLU model for ASR since many speech recognizers take PCFGs directly as the language model. Even with a non-CFG formalism like the UG, it can be converted to a CFG for ASR [24], [25]. However, in this case, simplicity remains only at the engineering level. A fundamental problem is that the knowledge-based models are generally not robust to disfluencies in spontaneous speech and recognition errors. They depend on the robust mechanism of the parser in the understanding component to deal with the inputs not covered by the model. This robust mechanism is not available in speech recognizers.

In statistical SLU, robustness is built into the model itself through proper smoothing. The remaining problems have more of an engineering nature, addressing how a statistical model like the HMM/CFG composite model can be converted into a format that a recognizer can take. In [26], the HMM/CFG composite model is converted to a CFG as follows: the backbone Markov chain is basically a statistical FSM, which is a subclass of the PCFG. The efficient conversion of the n-gram observation model follows the work in [27], and the CFG observation model is used directly. The composite model, in the format of PCFG, was applied under the framework of (32). The results were compared with the two-pass recognition/understanding paradigm under the framework of (31), where a domain-specific trigram was used as the language model $P(W)$ in speech recognition and the HMM/CFG composite model was used for the second-pass understanding.

Table 1 shows the findings with a commercial decoder and a research decoder. For the commercial decoder, even though the composite model's word error rate (WER) is over 46% higher than the trigram model, its SLU error rate (SLUER) is 17% lower. With the research decoder, which is less aggressive in pruning, the WER of the HMM/CFG model is about 27% higher than the trigram model. However, the SLU error rate is still marginally lower.

The results clearly demonstrate the suboptimality of separating the models for ASR and SLU. In this approach, the trigram is trained to optimize the likelihood of the training sentences. If the test data is drawn from the same distribution, the trigram model assigns higher likelihood to the correct transcriptions and, hence, reduces the WER. On the other hand, the objective of the HMM/CFG composite model training is to maximize the likelihood of the observed semantic representations. Thus, the correct semantic representations of the test sentences will be assigned higher probability. It is important to note that the trigram model used all the ATIS2 and ATIS3 training data, while the HMM/CFG composite model only used the 1993 ATIS3 training data. Although the comparison is not fair to the HMM/CFG composite model, it is informative because unannotated training data is much easier to obtain than the annotated data. When only the 1,700 training samples were used for trigram training, the two-pass system had a 10.4% WER and a 13.1% SLUER with the commercial recognizer, and a 7.5% WER and a 10.2% SLUER with the research recognizer.

The results from other research work also provide evidence for the importance of keeping the dependency link between acoustic observations and semantics. In [28], a language model that interpolated the word n-gram with n-grams containing semantically salient phrases was used for an automatic call-routing (ACR) task. A slight word accuracy improvement from the new language model resulted in a disproportionately substantial improvement in understanding. In [29], a single-pass ASR/ACR system, in which the ACR statistical model was used as

[TABLE 1] THE ASR WORD ERROR RATE AND THE SLU ERROR RATE (SLOT INS-DEL-SUB) OF THE TRIGRAM MODEL (TWO PASSES) AND THE HMM/CFG COMPOSITE MODE (ONE PASS). "TRANSCRIPTION" COLUMN SHOWS THE SLU ERROR RATE ON THE TRUE TEXT INPUT. BOTH AUTOMATIC AND MANUAL TRANSCRIPTIONS WERE SENT TO THE SAME HMM/CFG MODEL FOR A SECOND-PASS SLU.

DECODER		TRIGRAM	HMM/CFG	TRANSCRIPTION
COMMERCIAL DECODER	WER	8.2%	12.0%	—
	SLUER	11.6%	9.8%	5.1%
RESEARCH DECODER	WER	6.0%	7.6%	—
	SLUER	9.0%	8.8%	5.1%

the language model for ASR as well, resulted in a worse WER but better call classification accuracy. In [30], a concept decoder that adopted a model similar to the HMM/CFG model also yielded better understanding results.

OTHER STATISTICAL SPOKEN LANGUAGE RESEARCH

Although the majority of the SLU work falls into the general framework of (1), which we have focused on in this article, some other noteworthy research in SLU is based on different frameworks. For completeness of covering SLU, we briefly review such research in this section. The first class of such work is that described in [31] and [32], where semiautomatic grammar induction for SLU was developed based on statistical clustering techniques. The grammars obtained from these purely bottom-up, data-driven grammar induction algorithms require manual postprocessing for practical use.

Another class of SLU problems solved by a framework other than that of (1) is ACR. The goal of ACR is to classify and then route a user's call to one of several possible destinations through an interactive voice response (IVR) system. Typically, there is little limitation on what a user can say, and the system maps free-form language robustly to a simple semantic space, namely a small set of routing destinations for incoming phone calls. Some call-routing systems can optionally identify key concepts, but concept identification is not the major focus of research in ACR. Therefore, ACR is mainly a classification problem. Many, if not all, systems use statistical classifications rather than developing a semantic grammar [33]–[35], and the classifiers are often trained with discriminative technologies rather than maximum likelihood training. The work described in [36] provides an overview of one of the most well-known call-routing systems.

With the availability of (overwhelmingly) large multimedia contents including speech information, the understanding and organization of such content becomes a critical research topic to enable efficient use of the information. In this case, the understanding is no longer limited to a small domain. The work described in [37] is an introduction to the research in this area.

Since the late 1990s, there has been increasing interest in data-driven statistical parsing, based on the model learned from manually annotated tree banks [38], [39]. While the research focused on general syntactic parsing for NLP, parsing can potentially be applied to SLU when a large corpus is available. Details of that research are beyond the scope of this article.

SUMMARY AND CONCLUSION

This article is intended to serve as an introduction to the field of statistical SLU, based on the mainstream statistical modeling approach that shares a similar mathematical framework with many other statistical pattern recognition applications such as speech recognition. In particular, we formulated a number of statistical models for SLU in the literature as extensions to HMMs as segment models, where a multiple-word block (segment) with word dependency is generated from each underlying Markov state corresponding to each individual semantic slot defined from the application domain.

In the past, due partly to its nature of symbolic rather than numeric processing, the important field of SLU in human language technology has not been widely exposed to the signal processing research community. However, many key techniques in SLU originated from statistical signal processing. And because SLU is becoming increasingly important, as one major target application area of ASR that has been dear to many signal processing researchers, we contribute this article to provide a natural bridge between ASR and SLU in methodological and mathematical foundation. It is our hope that when the mathematical basis of SLU becomes well known through this introductory article, more powerful techniques established by signal processing researchers may further advance SLU to form a solid application area, making speech technology a successful component for intelligent human-machine communication.

AUTHORS

Ye-Yi Wang received a B.S. degree in computer science from Shanghai Jiao Tong University in 1985, a master's degree in computer science from Shanghai Jiao Tong University in 1988, a master's degree in computational linguistics from Carnegie Mellon University in 1992, and a Ph.D. degree in human language technology from Carnegie Mellon University in 1998. He joined Microsoft Research in 1998. His research interests include spoken dialog systems, natural language processing, language modeling, statistical machine translation, and machine learning. He was on the editorial board of the *Chinese Contemporary Linguistic Theory Series*. He is a coauthor of *Introduction to Computational Linguistics* (China Social Sciences Publishing House, 1997) and he has published over 30 technical papers. He is a Senior Member of the IEEE.

Li Deng received the B.S. degree from the University of Science and Technology of China in 1982 and the master's and Ph.D. degrees from the University of Wisconsin-Madison in 1984 and 1986, respectively. He worked on large-vocabulary ASR in Montreal, Canada, from 1986–1989. In 1989, he joined the Department of Electrical and Computer Engineering at the University of Waterloo, Ontario, Canada. In 1999, he joined Microsoft Research, Redmond, Washington as a senior researcher. His research interests include acoustic-phonetic modeling of speech, speech and speaker recognition, speech synthesis and enhancement, statistical methods and machine learning, spoken language systems, multimedia signal processing, and multimodal human-computer interaction. He has published over 200 technical papers and book chapters and is inventor and coinventor of numerous patents. He was on the IEEE Signal Processing Society Education Committee and Speech Processing Technical Committee, technical chair of ICASSP'04, and was associate editor for *IEEE Transactions on Speech and Audio Processing*. He serves on the Multimedia Signal Processing Technical Committee. He is Fellow of the IEEE and the Acoustical Society of America.

Alex Acero received a master's degree from the Polytechnic University of Madrid in 1985, another master's degree from Rice University in 1987, and a Ph.D. from Carnegie Mellon University in 1990, all in electrical engineering. He worked in Apple Computer's Advanced Technology Group from 1990–1991. In 1992, he joined Telefonica, Madrid, as manager of the speech technology group. In 1994, he joined Microsoft Research, Redmond, where he is a research area manager. He is currently an affiliate professor of electrical engineering at the University of Washington. He authored the books *Acoustical and Environmental Robustness in Automatic Speech Recognition* (Kluwer, 1993) and *Spoken Language Processing* (Prentice Hall, 2001) and has written invited chapters in three edited books as well as over 100 technical papers. He holds ten U.S. patents. His research interests include speech recognition, synthesis and enhancement, language modeling, spoken language systems, multimedia signal processing, and multimodal human-computer interaction. He was on the IEEE Signal Processing Society Speech Technical Committee, publications chair of ICASSP98, sponsorship chair of the 1999 IEEE Workshop on Automatic Speech Recognition and Understanding, and general cochair of the 2001 IEEE Workshop on Automatic Speech Recognition and Understanding. He was associate editor for *IEEE Signal Processing Letters* and is presently associate editor for *Computer Speech and Language* and *IEEE Transactions of Speech and Audio Processing*. He is a Fellow of the IEEE.

REFERENCES

[1] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "GALAXY—II: A reference architecture for conversational system development," in *Proc. Int. Conf. Speech and Language Processing,* Sydney, Australia, 1998, pp. 931–934.

[2] M.A. Walker, A. Rudnicky, R. Prasad, J. Aberdeen, E.O. Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellom, A. Potamianos, R. Passonneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard, "DARPA communicator evaluation: Progress from 2000 to 2001," in *Proc. Int. Conf. Speech and Language Processing,* Denver, Colorado, 2002, pp. 273–276.

[3] A. Waibel, "Interactive translation of conversational speech," *Computer*, vol. 29, no. 7, pp. 41–48, 1996.

[4] X. Huang, A. Acero, C. Chelba, L. Deng, D. Duchene, J. Goodman, H.-W. Hon, D. Jacoby, L. Jiang, R. Loynd, M. Mahajan, P. Mau, S. Meredith, S. Mughal, S. Neto, M. Plumpe, K. Wang, and Y.-Y. Wang, "MiPad: A next generation PDA prototype," in *Proc. Int. Conf. Speech and Language Processing*, Beijing, China, 2000, pp. 33–36.

[5] K. Wang, "Semantic synchronous understanding for robust spoken language applications," in *Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, St. Thomas, U.S. Virgin Islands, 2003, pp. 640–645.

[6] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "Match: An architecture for multimodal dialogue systems," in *Proc. 40th Ann. Meeting Association for Computational Linguistics*, 2002, pp. 376–383.

[7] W.A. Woods, "Language processing for speech understanding," in *Computer Speech Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1983, pp. 305–334.

[8] P. Price, "Evaluation of spoken language system: The ATIS domain," in *Proc. DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, 1990, pp. 91–95.

[9] W. Ward, "Recent improvements in the CMU spoken language understanding system," in *Proc. Human Language Technology Workshop*, Plainsboro, NJ, 1994, pp. 213–216.

[10] S. Seneff, "TINA: A natural language system for spoken language applications," *Comput. Linguistics*, vol. 18, no. 1, pp. 61–86, 1992.

[11] J. Dowding, J.M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran, "Gemini: A natural language system for spoken-language understanding," in *Proc. 31st Ann. Meeting Association for Computational Linguistics*, Columbus, Ohio, 1993, pp. 54–61.

[12] R. Pieraccini and E. Levin, "A learning approach to natural language understanding," in *Proc. 1993 NATO ASI Summer School*, Bubion, Spain, 1993.

[13] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden understanding models of natural language," in *Proc. 31st Ann. Meeting Association for Computational Linguistics*, New Mexico State University, 1994, pp. 25–32.

[14] M. Ostendorf, V. Digalakis, and O. Kimball, "From HMMs to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, no. 5, pp. 360–378, 1996.

[15] S. Della Pietra, M. Epstein, S. Roukos, and T. Ward, "Fertility models for statistical natural language understanding," in *Proc. 35th Ann. Meeting Association for Computational Linguistics*, Madrid, Spain, 1997, pp. 168–173.

[16] Y.-Y. Wang and A. Acero, "Combination of CFG and N-gram modeling in semantic grammar learning," in *Proc. Eurospeech 2003*, Geneva, Switzerland, 2003, pp. 2809–2812.

[17] Y. He and S. Young, "Hidden vector state model for hierarchical semantic parsing," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Hong Kong, China, 2003, pp. 268–271.

[18] D. Yu, K. Wang, M. Mahajan, P. Mau, and A. Acero, "Improved name recognition with user modeling," in *Proc. Eurospeech*, Geneva, Switzerland, 2003, pp. 1229–1232.

[19] K. Macherey, F.J. Och, and H. Ney, "Natural language understanding using statistical machine translation," in *Proc. Eurospeech*, 2001, pp. 2205–2208.

[20] P.F. Brown, V.J. Della-Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer, "Class-based n-gram models of natural language," *Comput. Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[21] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[22] F. Jelinek and E.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Pattern Recognition in Practice*, D. Gelsema and L. Kanal, Eds. Amsterdam: North-Holland, 1980, pp. 381–402.

[23] Y.-Y. Wang and Y.-C. Ju, "Creating speech recognition grammars from regular expressions for alphanumeric concepts," in *Proc. Int. Conf. Speech and Language Processing*, Jeju, Korea, 2004, pp. 2161–2164.

[24] R. Moore, "Using natural langauge knowledge sources in speech recognition," in *Proc. NATO Advanced Studies Institute*, 1998.

[25] M. Rayner, J. Dowding, and B.A. Hockey, "A baseline method for compiling typed unification grammars into context free language models," in *Proc. Eurospeech 2001*, Aalborg, Denmark, 2001, pp. 729–733.

[26] Y.-Y. Wang and A. Acero, "Is word error rate a good indicator for spoken language understanding accuracy," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, St. Thomas, U.S. Virgin Islands, 2003, pp. 577–582.

[27] G. Riccardi, R. Pieraccini, and E. Bocchieri, "Stochastic automata for language modeling," *Computer Speech and Language*, vol. 10, no. 4, pp. 265–293, 1996.

[28] G. Riccardi and A.L. Gorin, "Stochastic language models for speech recognition and understanding," in *Proc. Int. Conf. Speech and Language Processing*, Sidney, Australia, 1998.

[29] C. Chelba, M. Mahajan, and A. Acero, "Speech utterance classification," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Hong Kong, China, 2003, pp. 280–283.

[30] Y. Estève, C. Raymond, F. Bechet, and R.D. Mori, "Conceptual decoding for spoken dialog systems," in *Proc. Eurospeech 2003*, Geneva, Switzerland, 2003, pp. 617–620.

[31] C.-C. Wong and H. Meng, "Improvements on a semi-automatic grammar induction framework," in *Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, Madonna di Campiglio, Italy, 2001.

[32] A. Pargellis, E. Fosler-Lussier, A. Potamianos, and C.-H. Lee, "A comparison of four metrics for auto-inducing semantic classes," in *Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, Madonna di Campiglio, Italy, 2001.

[33] B. Carpenter and J. Chu-Carroll, "Natural language call routing: A robust, self-organizing approach," in *Proc. Int. Conf. Speech and Language Processing*, Sydney, Australia, 1998.

[34] D. Hakkani-Tür, G. Tur, M. Rahim, and G. Riccardi, "Unsupervised and active learning in automatic speech recognition for call classification," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Montreal, Canada, 2004, pp. 429–432.

[35] H.-K.J. Kuo, I. Zitouni, E. Fosler-Lussier, E. Ammicht, and C.-H. Lee, "Discriminative training for call classification and routing," in *Proc. Int. Conf. Speech and Language Processing*, Denver, CO, 2002, pp. 1145–1148.

[36] M. Gilbert, J. Wilpon, B. Stern, and G.D. Fabbrizio, "Intelligent virtual agents for contact center automation," *IEEE Signal Processing Mag.*, vol. 22, no. 5, pp. 32–41, Sept. 2005.

[37] L.-S. Lee and B. Chen, "Spoken document understanding and organization," *IEEE Signal Processing Mag.*, vol. 22, no. 5, pp. 42–60, Sept. 2005.

[38] E. Charniak, "A maximum-entropy-inspired parser," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics*, 2000.

[39] M. Collins, "Head-driven statistical models for natural language parsing," *Comput. Linguistics*, vol. 29, no. 4, pp. 589–637, 2003.