Spoken Languge Understanding

CONTENTS

PREFACE

The John Wiley LaTeX style file has been created to provide authors with a template for supplying their manuscript in a format that will be similar to the final typeset version. It has been produced to help authors to concentrate on the content without having to become too concerned about the style. We recommend that you read through this document carefully before you begin the LaTeX source for your book. By following the hints and style points set out in this document, you will be able to present your work in a way that will meet many of the required specifications for publication. These guidelines are not intended as an introduction to LaTeX. Further sources of information about LaTeX can be found in the appendix and bibliography at the end of this document.

1

Semantic Frame Based Spoken Language Understanding

Ye-Yi Wang, Li Deng and Alex Acero

Microsoft Research

Semantic frame based spoken language understanding (frame-based SLU) is one of the most commonly applied and well studied SLU technology for human-computer interaction. It has been used in many speech language processing tasks, in particular the transactional dialog systems, where various pieces of information need to be collected from users. A frame-based SLU system is often limited to a specific domain, which has a well-defined, relatively small semantic space. The structure of the semantic space can be represented by a set of templates called *semantic frames*, each contains some important component variables that are often referred as *slots*. The goal of the frame-based SLU is to choose the correct semantic frame for an utterance, and extract from the utterance the values of its component slots.

1.1 Background

1.1.1 History of the Frame-based SLU

In the United States, the study of the frame-based SLU started in the 1970's in the DARPA Speech Understanding Research (SUR) and then the Resource Management (RM) tasks. At this early stage, natural language understanding (NLU) techniques like finite state machine (FSM) and augmented transition networks (ATNs) were applied for SLU (Woods 1983). The study of SLU surged in the 90's, with the DARPA sponsored Air Travel Information System (ATIS) evaluations (Dahl et al. 1994; Hemphill et al. 1990). Multiple research labs from both academia and industry, including AT&T, BBN, Carnegie Mellon University, MIT and SRI, 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 answers from a standard database. ATIS is

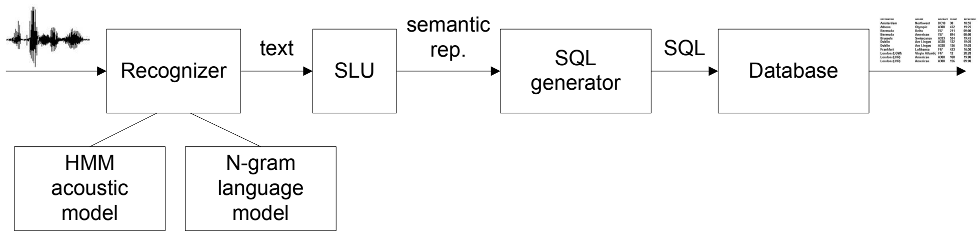


Figure 1.1 Frame-based SLU in a typical ATIS system, which consists of 1) a speech recognizer with both the acoustic model and language model trained with the ATIS specific data; 2) a SLU system that extracts the semantic representation (meaning) from the recognized text; and 3) a SQL generator that automatically generates the database query based on the semantic representation.

an important milestone for the frame-based SLU, largely thanks to its rigorous component-wise and end-to-end evaluation, participated by multiple institutions, with a common test set. Figure 1.1 shows the role of the frame-based SLU component in a typical ATIS system.

While ATIS focused more or less on the understanding of a single-turn utterance, the more recent DARPA Communicator program (Walker et al. 2001) focused on the rapid and cost-effective development of multi-modal speech enabled dialog systems, in which general infrastructures for dialog systems were developed, where different component systems for ASR, SLU, DM and TTS can be plugged in and evaluated. Naturally, many SLU technologies developed in ATIS were used in the SLU component of the Communicator program. Eight systems from AT&T, BBN, University of Colorado, Carnegie Mellon University, IBM, Lucent Bell Labs, MIT, and SRI participated in the 2001 evaluation (Walker et al. 2002). In the mean time, the AI community had separate effort in building a conversational planning agent, such as the TRAINS system (Allen et al. 1996b).

Parallel efforts were made on the other side of the Atlantic. The French EVALDA/MEDIA project aimed at designing and testing the evaluation methodology to compare and diagnose the context-dependent and independent SLU capability in spoken language dialogs. Participants included both academic organizations (IRIT, LIA, LIMSI, LORIA, VALORIA, CLIPS) and industrial institutions (FRANCE TELECOM R&D, TELIP). Like ATIS, the domain of this study was restricted to database queries for tourist and hotel information.

The more recent LUNA project sponsored by the European Union focused on the problem of real-time understanding of spontaneous speech in the context of advanced telecom services. Its major objective is the development of a robust SLU toolkit for dialog systems, which enhances users experience by allowing natural human-machine interactions via spontaneous and unconstrained speech. One special characteristic of the project, which is absent in the similar projects in the US, is its emphasis on multilingual portability of the SLU components.

Traditionally, the frame-based SLU has adopted a knowledge-based solution. The problem is tackled by writing context free (CFG) or unification grammars (UG) by hand. The manual grammar authoring process is laborious, expensive and requires a lot of expertise. In the early 90's, both knowledge-based and data-driven approaches have been applied in different ATIS systems. Currently most commercial applications use the knowledge-based solutions, while most research systems adopt a data-driven, statistical learning approach to SLU. Attempts

```
<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>
```

Figure 1.2 Simplified semantic class schema for the ATIS domain. **DCity** stands for "departure city" and **ACity** stands for "arrival city".

have also been made to incorporate knowledge in a data-driven system.

1.1.2 Semantic Representation and Semantic Frame

What is the goal for 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 "understood" an input utterance. For the frame-based SLU tasks, using the ATIS domain as an example, this can be measured by the accuracy of the air travel related information returned from a system after a spoken query is made by a user. However, generating the information involves more than the SLU component. For better engineering practice and scientific studies, it is desirable to modularize the end-to-end system and isolate the SLU module. For this purpose, an intermediate semantic representation is introduced to serve as the interface between different modules. Many spoken language systems adopt their own semantic representations. However, most of them can be abstracted as the semantic frame-based representation, which we introduce now.

The semantic structure of an application domain is defined in terms of *semantic frames*. Figure 1.2 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 the words like "flight", "flights") or the FARE semantic terminal (expressed by the words like "fare", "cost") that specifies what particular information a user needs. In the *Flight* frame, **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 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.

The meaning of an input sentence is an instantiation of the semantic frames. Figure 1.3 shows the meaning representation for the sentence "Show me flights from Seattle to Boston

```
<ShowFlight>
    <subject type="Subject">FLIGHT</subject>
    <Flight frame="Flight" type="Flight">
        <DCity type="City">SEA</DCity>
        <ACity type="City">BOS</ACity>
        <DDate Type="Date">12/24</DDate>
    </Flight>
</ShowFlight>
```

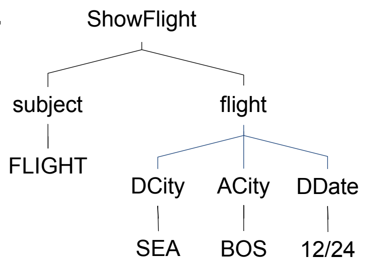


Figure 1.3 The semantic representation for "Show me flights from Seattle to Boston on Christmas Eve" is an instantiation of the semantic frames in Figure 1.2. On the right is its tree representation. The instantiation picks a frame that represents the meaning conveyed in the sentence and fills its slots accordingly.

[Command: DISPLAY] [Subject: FLIGHT] [DCity: SEA] [ACity: BOS][DDate: 12/24]

Figure 1.4 The 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 on Christmas Eve."

on Christmas Eve." Here the frame "*ShowFlight*" contains the sub-frame "*Flight*". Some SLU systems do not allow any sub-structures 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* (Pieraccini and Levin 1993) or *flat concept* representation (Figure 1.4).

The hierarchical representation is more expressive and allows the sharing of substructures. For example, the *Flight* frame in Figure 1.2 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.

The semantic representation in the French MEDIA project adopts an attribute-value list to represent the hierarchical semantic information (Bonneau-Maynard et al. 2005), as shown in Figure 1.5. This representation, used in the official annotation and evaluation, is ostensibly quite different from the frame based representation. However, a hierarchical representation in Figure 1.6 can be constructed from such a representation, which is much similar to the frame-based representation. This effectively brings the expression power of the MEDIA annotation/evaluation scheme much similar to the frame-based representation. In Figure 1.5, each segment of the sentence is tagged with a *mode* that takes four possible values: affirmative (+), negative (-), interrogative (?) or optional (@). While most segments are labeled with the "+" mode, the "le" and "tarif" segments are labeled as "?", indicating this ("the rate") is the information that the user has asked for. The *attribute names* encode the hierarchical semantic information, and *normalized values* represents the canonical values for the attributes. To encode the hierarchical information, an attribute name contains multiple parts separated by hyphens, each part represents an attribute belonging to an attribute class: The *database attributes* class contains the attributes from a database table. For attributes A and B in this class, A-B implies that B is an attribute in the substructure of A. For example,

words	mode	attribute name	normalized value
donnez-moi	+	`null`	
le	?	`refLink-coRef`	`singular`
tarif	?	`object`	`payment-amount-room`
puisque	+	`connectProp`	`imply`
je voudrais	+	`null`	
une chambre	+	`number-room`	`1`
qui coûte	+	`object`	`payment-amount-room`
pas plus de	+	`comparative-payment`	`less than`
cinquante	+	`payment-amount-integer-room`	`50`
euros	+	`payment-unit`	`euro`

Figure 1.5 Semantic concept (attribute/value) representation for the utterance *"give me the rate for I'd like a room charged not more than fifty euros."* (Courtesy of Fabrice Lefèvre).

```
refLink:  co-ref.
          singular
object:   room
          payment:  amount:       ?
imply
object:   room
          number:   1
          payment:  comparative:  less than
                    amount:       integer:   50
                    unit:         euro
```

Figure 1.6 Hierarchical representation derived from the attribute-value list in Figure 1.5.

`payment-amount-integer` indicates that `amount` is an attribute in the substructure of `payment`, while `integer` is an attribute in the substructure of `amount`. Similarly, `payment-unit` implies that `unit` is another attribute in the substructure of `payment`. A *modifier attribute* M is linked to a database attribute A in the form of M-A, indicating that M modifies A. For example, `comparative-payment` states that `comparative` is a modifier of `payment`, hence it is part of the `payment` structure. To fully reconstruct the hierarchical representation, additional information is necessary to specify which components should be joined together to form a structure. For that purpose, Bonneau-Maynard et al. (2005) introduced the *specifiers* that can be attached to the end of a hierarchical attribute name. For example, the specifier `room` in Figure 1.5 indicates that `number` and `payment` should be grouped together under the `room` specifier, as shown in Figure 1.6.

1.1.3 *Technical Challenges*

The frame-based SLU is closely related to natural language understanding (NLU), a field that has been studied for more than half a century. NLU focus mainly on understanding of general domain written texts. Because there is not a specific application domain for the general purposed NLU, the semantics in NLU have to be defined in a broader sense, such as thematic roles (agents, patients, etc.) In contrast, the frame-based SLU has, in the current state of technology, focused only on specific application domains. The semantics are defined very specifically according to the application domain, as illustrated by the above examples of semantic frames. Many domain-specific constraints can be included in the understanding model. Ostensibly, this may make the problem easier to solve. Unfortunately, there are many new challenges for spoken language understanding, including

- **Extra-grammaticality** – spoken languages are not as well-formed as written languages. People are in general less careful with speech than with writings. They often do not comply with rigid syntactic constraints.
- **Disfluencies** – false starts, repairs, and hesitations are pervasive, especially in conversational speech.
- **Speech recognition errors** – Speech recognition technologies are far from perfect. Environment noise, speaker's accent, domain specific terminologies, all make speech recognition errors inevitable. It is common to see that a generic speech recognizer has over 30% word error rates on domain specific data.
- **Out-of-domain utterances** – a dialog system can never restrict a user from saying anything out of a specific domain, even in a system-initiated dialog, where users are prompted for answers to specific questions. Because the frame-based SLU focuses on a specific application domain, out-of-domain utterances are not well modeled and can often be confused as an in-domain utterance. Detecting the out-of-domain utterances is not an easy task – it is complicated by the extra-grammaticality, disfluencies and ASR errors of in-domain utterances.

In summary, robustness is one of the most important issues in SLU. A system should be able to gracefully handle the unexpected inputs. If an input string is not accepted by a grammar/model, it is still desirable to identify the well-formed concepts in the input that carry important information for a given domain. On the other hand, a robust solution tends to over-generalize and introduce ambiguities, leading to reduction in understanding accuracy. A major challenge to the frame-based SLU is thus to strike an optimal balance between the robustness and the constraints that prevent over-generalizations and reduce ambiguities.

1.1.4 *Standard Data Sets*

While there are many commercial applications for the frame-based SLU, most of the data used in those applications are proprietary. The ATIS corpus (Dahl et al. 1994; Hemphill et al. 1990) is the most broadly used data set by SLU researchers. It is more realistic compared to the previous speech corpus in the sense that it is the spontaneous spoken language instead of the read speech, therefore it contains disfluencies, corrections, and colloquial pronunciations. It was collected in a normal office setting, with a Wizard of the Oz interaction between a system and a subject who issued spoken queries for air travel information. Utterances were recorded, manually transcribed, and manually categorized into three different classes: class A for queries that can be interpreted without the context information; class D for queries that can be interpreted with the context information; class X for unanswerable queries (e.g., out of domain utterances). The corpus also contains a back-end database of US domestic flights. Each utterance in class A or D is also manually associated with a SQL query for the database, together with the reference answer, which is the corresponding query result from the database.

The ATIS data set was designed to advance the state-of-the-art in speech recognition and understanding in early '90s. The DARPA Communicator project in 00's focused on the rapid and cost-effective development of multi-modal speech-enabled dialog systems with advanced conversational capabilities. The data collected from the project are richly annotated with dialog related information, including sessions/turns/operation information, etc.

Both ATIS and Communicator corpus are available through the Linguistic Data Consortium (LDC) (LDC n.d.). The data collected by the TRAIN project (Allen et al. 1996a) is also available at LDC.

For the European SLU projects, the French corpus MEDIA has been annotated in terms of semantic structures as in Figure 1.5 For the LUNA project that focuses on multilingual SLU and language specific aspects for language modeling and understanding, new corpora with complex human-human dialogs have been acquired in Italian and Polish. They are transcribed and annotated in terms of the syntactic constituents and semantic structures.

The MEDIA data is available via the European Language Resource Association (ELRA) (ELRA n.d.).

1.1.5 Evaluation Metrics

Various metrics are used in the evaluation of the frame-based SLU systems. Some metrics focus on component-wise performance of the SLU sub-system, others emphasize the impact of the SLU component on the performance of the end-to-end system. Here we list some commonly used evaluation metrics.

- **Sentence/utterance Level Semantic Accuracy** (SLSA): this metric measures the percentage of correct semantic representations assigned to a sentence/utterance. An intermediate reference semantic representation is required with this metric.

$$SLSA = \frac{\text{\# of sentences assigned correct semantic representation}}{\text{\# of sentences}} \tag{1.1}$$

- **Slot Error Rate** (SER): slot error rate (a.k.a concept error rate (CER)) measures the slot level performance of a frame-based SLU system. It aligns the semantic representation of a sentence with its reference representation, find the number of slots incorrectly identified by the SLU system. These include the inserted slots (present in the SLU output, absent in the reference representation), deleted slots (absent in the SLU output, present in the reference) and substituted slots (slots that are aligned to each other between the SLU output and the reference differ in either the slot labels or the sentence segments they cover).

$$SER = \frac{\text{\# of inserted/deleted/substituted slots}}{\text{\# of slots in reference semantic representations}} \quad (1.2)$$

- **Slot Precision/Recall/F$_1$ Score**: Precision/Recall is another way to measure the slot level SLU performance. F$_1$ combines precision and recall into a single score as their harmonic mean:

$$Precision = \frac{\text{\# of reference slots correctly detected by SLU}}{\text{\# of total slots detected by SLU}} \quad (1.3)$$

$$Recall = \frac{\text{\# of reference slots correctly detected by SLU}}{\text{\# of total reference slots}} \quad (1.4)$$

$$F_1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (1.5)$$

- **End-to-end Evaluation**: The above evaluation metrics compare the SLU output with the reference semantic representation. Since different systems may use different intermediate semantic representations, it is difficult to compare different SLU components even if a common test set is available. In the original DARPA-sponsored ATIS evaluation, the problem is addressed by an end-to-end evaluation metric. Utterances were first recognized and parsed to produce the semantic representations, from which SQL queries were generated and submitted to the backend database engine. The resulting database outputs were then compared with the target entries created by using the manually labeled SQL queries. Here a target entry is a pair of minimum and maximum information, which indicates the columns that have to be returned (minimum information) and the columns that are permitted to be included in the database search results (maximum information). The minimum-maximum restriction penalizes the implementation that always returns the information from all the fields of a database entry without understanding what specific information the user has requested for. The evaluation for SLU can be conducted on both the ASR and the manual transcriptions of the test utterances. The utterance level understanding accuracy is the percentage of the test utterances for which the correct database entries are output with at least the minimum information (database columns) and at most the maximum information.

1.2 Knowledge-based Solutions

1.2.1 Semantically-enhanced Syntactic Grammars

Many advocates of the knowledge-based approach believe that general linguistic knowledge is helpful in modeling domain specific language. This includes the syntactic constraints as well as some optional rudimentary domain-independent semantic knowledge. However, since the ultimate goal is to extract the domain-dependent semantic information, one major question is how to inject the domain specific semantic constraints into a domain-independent grammar.

MIT's TINA system (Seneff 1992) aims at the graceful, seamless interface between syntax and semantics. It uses context free grammar (augmented with a set of features used to

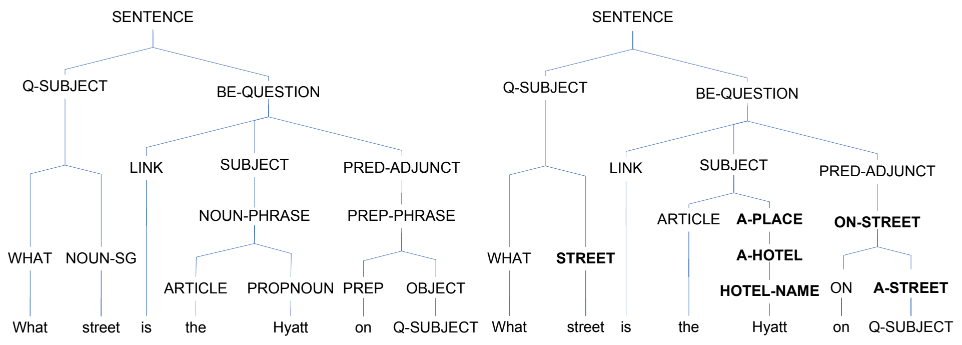


Figure 1.7 TINA parse tree with syntactic rules only (left) and with lower-level syntactic rules replaced by domain-dependent semantic rules (right). The second one was reproduced from (Seneff 1992).

enforce several syntactic and semantic constraints via unification). The injection of the domain-dependent semantics is accomplished by replacing the low level syntactic non-terminals with the semantic non-terminals. In doing so, the top level syntactic rules makes the grammar capable of modeling the domain-independent linguistic constraints, such as the Wh-movement/trace-management. The feature unifications introduce additional syntactic and semantic constraints such as the person and number agreement and the subject/verb selectional restrictions. On the other hand, the domain-specific semantic content of a sentence is completely encoded by the lower level semantic non-terminals (categories) in the parse tree, thus making it feasible to extract the semantic frame representation directly from a parse tree. Figure 1.7 shows two pare trees – one with the syntactic non-terminals only, the other has the low level syntactic categories replaced with the (bold) domain-specific semantic non-terminals.

The context-free rules are automatically converted to a graph representation of shared transition networks, where the transitions can be associated with probabilities by automatic training when training data is available. It is reported in (Seneff 1992) that the introduction of transition probabilities has greatly reduced the perplexity of the grammar – from 368 to 41.4 in the Resource Management domain.

SRI's Gemini system (Dowding et al. 1993) is implemented on top of its Core Language Engine (CLE) (Alshawi 1992), a general natural language understanding system that parses an input sentence and generates its semantic representation in the logical forms. Here the unification grammar is also used to model the general syntactic constraints. Unlike TINA that blends syntax and semantics together with the mixed grammar categories in constituent parsing, it clearly separates the domain independent syntax from the domain dependent semantics. Another specialty of Gemini is its adoption of the logical forms instead of the frame-like representation for semantics. Logical form in CLE is an extension to the first-order (predicate) logic (Alshawi and van Eijck 1989). The example below shows the logical form for "a flight to Boston":

```
exists(A, [and, [flight, A], [to, A 'BOSTON']])
```

which can be read as "there is an variable A, such that A is a flight and A is to 'BOSTON'."

In Gemini, recognized utterances are parsed according to syntactic rules. Each syntactic node in a parse tree invokes a set of semantic rules to construct a set of logical forms for the node. Both domain dependent and domain-independent selectional restrictions on logical forms are enforced through *sortal constraints*. Sortal constrains are used by linguists, logicians and philosophers to explain the oddity of sentences like "the flight to boston is a cat", which cannot be conveniently explained by syntactic theories. Instead, the oddity is explained by sort incompatibility – the semantic interpretation for each constituent in a parse tree is assigned a sort. The sorts of constituents need to be compatible with each other when compositional semantics are constructed with respect to the syntactic rule that combines the constituents. Since the sorts of "flight" and "cat" are not compatible, the reading of the sentence is odd.

Sortal constraints can be used independent of the application domains – they can model the selectional constraints expected by the predicates for their arguments in logical forms, hence they cut down on structural (e.g. attachment) and word sense ambiguities. In Gemini, sortal constraints are also used to model the semantic restrictions imposed by an application domain, such that the logical forms are restricted by the domain constraints. The following is the logical form with sort assignments for the previous example:

```
exists((A; [flight]),
       [and, [flight, (A; [flight])]; [prop],
             [to, (A; [flight]), (‘BOSTON’; [city])]; [prop]
       ];[prop]
      ); [prop]
```

here every expression/sub-expression in the logical form is followed by a sort it has been assigned to, separated by a semicolon. For example, the quantified variable `A` is assigned the sort `[flight]`. The quantifier `exists(A, [P A])` is assigned the sort `[prop]` (proposition), so are the sub-expressions `[flight, (A; [flight])]`, `[to, (A; [flight]), ('BOSTON'; [city])]` and their conjunction (the `and` expression). The literal `‘BOSTON’` is assigned the sort `[city]`.

To construct the logical form in the previous example, a grammar developer needs to introduce the following sortal constraints:

```
sort(‘BOSTON’, [city])
sort(to, [[flight], [city]], [prop])
sort(and, [[prop], [prop]], [prop])
```

Here the first sort rule specifies that the literal 'BOSTON' can be assigned the sort `[city]`. The second rule indicates that the `to` predicate takes two arguments, one must have been assigned the sort `[flight]`, the other assigned `[city]`. The compositional semantics of the predicate is assigned the sort `[prop]`. The third rule states the conjunction operator takes two arguments assigned with the sort `[prop]` and produces the compositional logical form with the sort `[prop]`.

Gemini adopts a two-stage parser. In the first stage of *constituent parsing*, bottom-up chart parser is used, which applies the syntactic and semantic rules to populate the chart with linguistic constituents that include the syntactic and logical form information. In the second stage of *utterance parsing*, a second set of syntactic and semantic rules is applied. The rules are required to span the entire utterance, such that a complete parse for the utterance can be constructed.

To improve the robustness to spontaneous speech with disfluencies, when no complete parse can be constructed during the stage of utterance parsing due to the violation of either syntactic or sortal constraints, a repair component is invoked to detect and correct the disfluencies. To handle speech recognition errors, Gemini operates on the n-best ASR outputs.

1.2.2 Semantic Grammars

While using the semantically-enhanced syntactic grammars saves grammar developers from the effort to model the general language structures, it requires profound knowledge about the general syntactic grammars. In addition, the knowledge-based approach often requires the exact matching of input sentences to the grammar rules, which makes it not robust to ASR errors, extra-grammaticality and disfluencies in spontaneous speech. Often it has to resort to some kind of semantic-based robust parsing as a backup.

The Phoenix spoken language understanding system (Ward 1991) directly models the domain dependent semantics with a semantic grammar. It was used by CMU in the ATIS evaluation, and was one of the top performing SLU systems in the evaluation. As we have discussed previously about semantic representation, it uses semantic frames to represent semantic relations – the basic type of action for the application. Slots in a frame are filled by matching the input strings (sentences) against the slot-nets, the recursive transition networks (RTNs) that specifies the patterns for filler strings. RTNs are finite state transition networks, where the arcs in the networks can include not only terminal words, but also calls to other networks. They are equivalents of the context free grammars in graph representation. During parsing, the system uses the slot-nets to match substrings in the input sentence. When a slot-net matches a substring, it is passed along for incorporation into the frames. While the slot-nets require the exact matches, the system is robust in the phase when a set of matching slots are composed into a semantic frame – beam search is used in frame construction. When a slot matches, it will extend all the active frames that contain the slot, and activate any currently inactive frames that contain the slot. At the end of the search process, the single best parse that covers the most slots discovered by the slot-nets is returned from the beam.

The grammar used by Phoenix for ATIS was very complicated. It consisted of 3.2K non-terminals and 13K grammar rules. Figure 1.8 shows the slot-net for “PriceRange”, together with some of the sub-nets that the slot-net calls.

1.2.3 Knowledge-based Solutions in Commercial Applications

In most commercial spoken dialog systems, the frame-based SLU is tackled via the knowledge-based approach. Domain dependent semantic context free grammars are developed to provide the powerful domain and linguistic constraints to the ASR component (language model), and to provide a mechanism to construct the target semantic representation from an input utterance. The rules in a grammar define the permissible syntactic/semantic expressions. They are also associated with semantic interpretation tags, from which the ASR engine can construct the semantic representation from the parse tree.

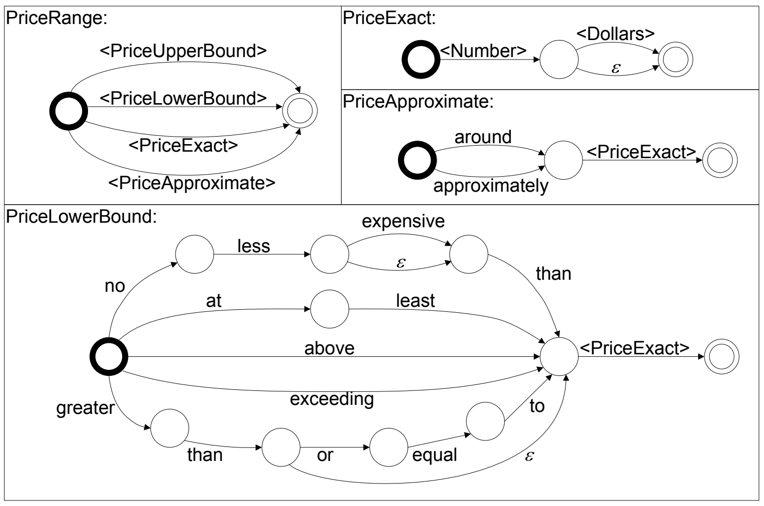


Figure 1.8 Recursive transition network for "PriceRange," together with three sub-nets called by it: "PriceExact", "PriceApproximate" and "PriceLowerBound." The arc labels in angular bracket indicate calls to sub-networks.

To ensure the portability of a speech recognition grammar, W3C has standardized the grammar format SRGS (Speech Recognition Grammar Specification) (Hunt and McGlashan n.d.) and the semantic interpretation tags SISR (Semantic Interpretation for Speech Recognition) (W3C n.d.). Figure 1.9 shows an exemplar grammar with semantic interpretation tags defined in the SRGS XML form (The alternative ABNF form is more compact but less readable). Four rules of the grammar are shown in the figure, which correspond to the RTNs in Figure 1.8. Each rule has a rule variable "`out`", which holds the semantic values of the rule. A rule valuable may have a structured value – it can contains hierarchical properties. For example, `out.amount` represents the "`amount`" property of the rule variable. A rule may not be associated with an explicit semantic interpretation tag. In such a case, the semantic structure (value of the rule variable) for the rule is constructed according to the implicit semantic interpretation tags – If there are no rule references (the <`ruleref`> tag) in the parse, the text covered by the rule is assigned to the rule variable. Otherwise, the value of the rule variable of the last rule reference (non-terminal) in the parse is automatically copied into the rule variable. The semantic of a rule reference can be obtained via the "`rules`" variable. For example, "`rules.PriceExact`" in Figure 1.9 refers to the value of the structured rule variable (semantic structure) for the rule reference "`PriceExact`" in "`PriceApproximate`" and "`PriceLowerBound`", and "`rules.PriceExact.amount`" represents the value of the "`amount`" attribute in the structured rule variable of "`PriceExact`".

Given the grammar in Figure 1.9, the parse tree for the utterance "about 500 dollars," together with the values of the rule variables for the rules referenced in the tree, is illustrated in Figure 1.10. From which the semantic representation for the entire utterance can be constructed as follows:

```
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar"
         xml:lang="en-US" tag-format="semantics/1.0-literals"
         root="PriceRange">
   <rule id="PriceRange" scope="public">
      <one-of>
         <item><ruleref uri="#PriceExact"/></item>
         <item><ruleref uri="#PriceApproximate"/></item>
         <item><ruleref uri="#PriceLowerBound"/></item>
         <item><ruleref uri="#PriceUpperBound"/></item>
      </one-of>
      <tag>out.pricerange=rules.latest()</tag>
   </rule>
   <rule id="PriceExact">
      <item><ruleref uri="#Number"/></item>
      <item repeat="0-1"><ruleref uri="#Dollars"/></item>
      <tag>out.amount=rules.Number</tag>
      <tag>out.match="exact"</tag>
   </rule>
   <rule id="PriceApproximate">
      <one-of><item>about</item>
              <item>approximately</item>
      </one-of>
      <item><ruleref uri="#PriceExact"/></item>
      <tag>out.amount=rules.PriceExact.amount</tag>
      <tag>out.match="approximate"</tag>
   </rule>
   <rule id="PriceLowerBound">
      <one-of><item>exceeding</item>
              <item>above</item>
              <item><token>no less</token>
                    <item repeat="0-1">expensive</item>
                    <token>than</token>
              </item>
              <item><token>greater than</token>
                    <item repeat="0-1">or equal to</item>
              </item>
      </one-of>
      <item><ruleref uri="#PriceExact"/></item>
      <tag>out.amount=rules.PriceExact.amount</tag>
      <tag>out.match="lowerbound"</tag>
   </rule>
   ......
</grammar>
```

Figure 1.9 W3C SRGS grammar for “PriceRange” with SISR semantic interpretation tags.

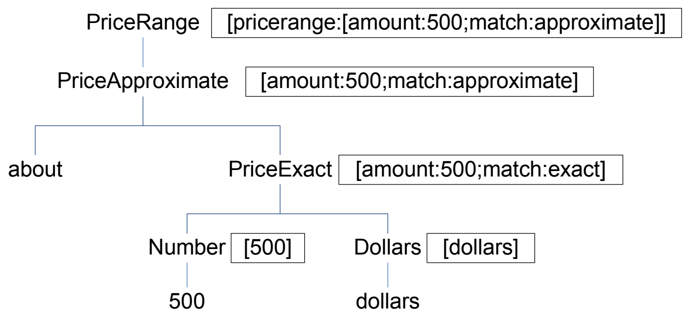


Figure 1.10 The parse tree for the utterance “about 500 dollars” together with the semantic structures for the non-terminal nodes (in boxes), constructed according to the explicit or implicit semantic interpretation tags.

```
<pricerange>
    <amount>500</amount>
    <match>approximate</match>
</pricerange>
```

Readers can find more detailed information about the SRGS grammar format and the semantic interpretation tags in (Hunt and McGlashan n.d.) and (W3C n.d.).

1.3 Data-Driven Approaches

The knowledge-based solution has the advantage of not requiring much labeled data. In addition, almost everyone can start writing a SLU grammar with some basic training. The grammar can be used as both the ASR language model and the SLU model in a single pass speech understanding. However, a knowledge-based system is difficulty and expensive to develop and maintain due to the following reasons:

1. Grammar development is an error-prone process. While it does not take much effort for a common developer to learn the syntax for writing a speech understanding grammar, it requires combined linguistic and engineering expertises, plus the deep knowledge about the application domain, to write a *good* grammar. Grammar authoring is a balancing act between simplicity and coverage. People talk differently, therefore a good grammar has to account for the different expressions for the same concept, action or request. If a grammar is too simple, it is inadequate to model the linguistic diversity. On the other hand, if a grammar is too complicated, it may not only slow down the parser, but also increase the ambiguities, hence confuses the SLU system and degrades its performance. Design the structure of a grammar is an art. It takes much experience to have a good design where frequently used concepts, for example, are modeled by separate rules to be shared by other rules at a higher level.
2. It takes multiple rounds to fine tune a grammar. Grammar authoring can hardly be a one-shot deal – nobody can write a perfect grammar with a single try. Furthermore, grammars need to evolve over time – new features and scenarios may be introduced

to an application after its initial deployment. Ideally, an SLU system should be able to automatically adapt to the real data collected after its deployment. On the contrary, knowledge-based systems require an expert's involvement, sometimes even the involvement of the original system designer, in the adaptation loop.

3. Grammar authoring is difficult to scale up. It is relatively easy to write a grammar to model a single concept as in a system-initiated dialog system, where the user is prompted to provide a single piece of information (e.g., name, account number, social security number, address, etc.) at a dialog turn. However, if we allow users to volunteer multiple pieces of information in a single utterance, the ways to put together these pieces are combinational. As we have shown previously, for a very restricted domain like ATIS, the semantic grammar already contains 3.2k non-terminals and 13k grammar rules.

SLU based on data-driven statistical learning approaches directly addresses many of the problems associated with the knowledge-based solutions. Statistical SLU systems can automatically learn from example sentences with their corresponding semantics annotated. Compared to the manual grammar authoring, the annotations are much easier to create, without the requirement of the specialized knowledge. The statistical approach can adapt to new data, possibly via unsupervised learning. One disadvantage of such an approach, however, is the data-sparseness problem. The requirement of a large amount of labeled training data is not very practical in real-world applications, which are quite different from a few showcase problems studied in research labs. This is the case especially at the early stage of system development.

In this section, we introduce a general framework for the statistical SLU, and review various types of statistical SLU models in the literature. We assume that most readers have been exposed to HMMs as commonly used in speech recognition and language processing.

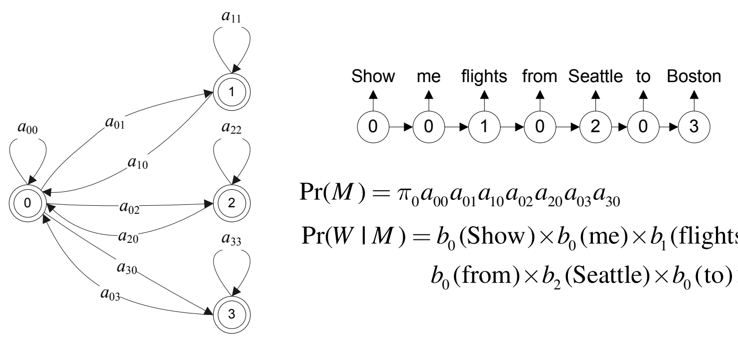
1.3.1 Generative Models

In the statistical frame-based SLU, the task is often 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 $P(M \mid W)$. In the generative model framework, the following decision rule is used:

$$\hat{M} = \arg\max_{M} P(M \mid W) = \arg\max_{M} P(W \mid M)P(M) \tag{1.6}$$

And the objective function of a generative model is to maximize the joint probability $P(W,\, M) = P(W \mid M)P(M)$ given a training sample of W and its semantic annotation M.

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 $P(W \mid M)$, sometimes called *lexical generation* or *realization* model (Miller et al. 1994), 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 implementation of Eq. (1.6), 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.



$$\Pr(M) = \pi_0 a_{00} a_{01} a_{10} a_{02} a_{20} a_{03} a_{30}$$

$$\Pr(W \mid M) = b_0(\text{Show}) \times b_0(\text{me}) \times b_1(\text{flights}) \times b_0(\text{from}) \times b_2(\text{Seattle}) \times b_0(\text{to}) \times b_3(\text{Boston})$$

Figure 1.11 An HMM tagging model for SLU. State 0 represents "command", state 1 represents the "subject" slot, state 2 represents the "DCity" slot, 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 $<$DCity 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 post-processing step.

The alignment between the observations (words) and the states is hidden (Figure 1.11). 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 the problem by allowing a state to emit one or more "segments" of multiple words at a time. In this case, the generative process for the observations is:

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

Here S, A and L are not observed. They are hidden variables that can be marginalized out according to

$$P(W \mid M) = \sum_{S,A,L} P(W, S, A, L \mid M) \tag{1.7}$$

$$= \sum_{S,A,L} P(S \mid M)\, P(A \mid S, M)\, P(L \mid A, S, M)\, P(W \mid L, A, S, M) \tag{1.8}$$

```
<ShowFlight>
        <subject>flights</subject>
        <Flight>
                <DCity>Seattle</DCity>
                <ACity>Boston</ACity>
                <DDate>Christmas Eve</DDate>
        </Flight>
</ShowFlight>
```

Figure 1.12 Simplified semantic representation – a separate post-processing is necessary for slot value normalization. Nevertheless, the representation is easier to annotate.

This type of lexicalization model for correlated word observation sequences is analogous to the segment model in acoustic modeling for ASR (Ostendorf et al. 1996). One main difference, however, is that in SLU the alignment process is more elaborated than that for 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 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 many different ways a fixed meaning may be expressed as a composite of phrases/words in varying orders. The purpose of the alignment in Step 2 above is to account for this type of variability, which is largely absent in ASR problems.

Eq. (1.7) has different implementations in various SLU systems. We will discuss this topic of lexicalization modeling in detail in later in this section after surveying semantic-prior modeling first.

Note that in the semantic representation in Figure 1.3 or Figure 1.4, the frame slots have normalized values that may be different from the original text in the utterance (e.g., the original text "Seattle" is normalized as the city code "SEA"). Such a representation relieves the application developers from handling different expressions for the same meaning. However, training a model to produce such a normalized semantic representation requires the training data be labeled in the same way, which adds extra burden to the annotators and makes the model more complicated – we will show one of such models later in a subsection titled "2+1 SLU Model.". Instead, a majority of statistical models just align the segments of an utterance to the slots and produce a semantic representation like the one in Figure 1.12, and employ a separate post-processing step to generate the normalized representation.

Semantic Priors in Understanding Models

In statistical SLU that models 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 (Pieraccini and Levin 1993) and IBM's fertility model (Della Pietra et al. 1997), the topology of the model is a fully connected network as in Figure 1.13. Unlike the topology in Figure 1.11, there is no self loop on a state because it is already modeling a segment of words with the length information already encoded.

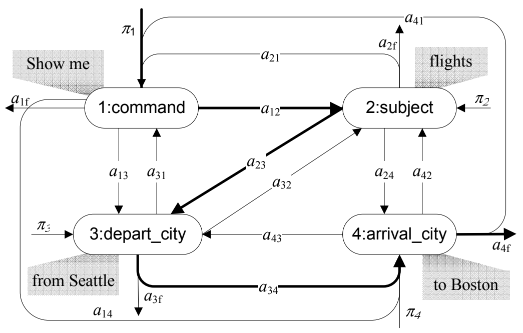


Figure 1.13 The topology of the statistical model that adopts the flat concept semantic representation. Each state represents a concept. States are fully interconnected. The initial state distribution $\pi_i = P(i)$ and the state transition distribution $a_{ij} = P(j \mid i)$ comprise the semantic prior of the model. (The final state f = "</s>" is not shown in the topology.) The thicker lines illustrate the correct state transitions for the sentence "Show me flights from Seattle to Boston." The example sentence is also aligned to the states.

The prior probability of the semantic structure underlying the representation in Figure 1.4, 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 1.13:

$$P(M) = P\left(\text{Command}|\text{<s>}\right) P\left(\text{Subject}|\text{Command}\right) P\left(\text{ACity}|\text{Subject}\right)$$
$$P\left(\text{ACity}|\text{DCity}\right) P\left(\text{</s>}|\text{ACity}\right) = \pi_1 a_{12} a_{23} a_{34} a_{4\text{f}}$$

where <s> represents the start of the semantic concept list and </s> represents the end of the flat-concept list. The model is also used by a MEDIA system that uses attribute-value list to represent the hierarchical semantic information (Lefèvre 2007).

For models that use hierarchical semantic structures, including BBN's Hidden Understanding Model (Miller et al. 1994) and Microsoft Research's HMM/CFG composite model (Wang and Acero 2003a), the semantic prior is a natural extension of Eq. (1.9). Figure 1.14 shows the topology of the underlying states for the semantic frames in Figure 1.2. The left part of the diagram shows the top level network topology, and the right part shows a zoomed-in sub-network for state 2, which represents the embedded "Flight" frame. The initial state probabilities $\pi_1 = P(\text{ShowFlight})$ and $\pi_5 = P(\text{GroundTrans})$ 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 sub-network 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 1.3 is the product of the Markov transitional probabilities across the different levels in the semantic hierarchy (The marked path in Figure 1.14):

$$P(M) = P(\text{ShowFlight}) P(\text{Subject}|\text{<s>}; ShowFlight) P(\text{Flight}|\text{Subject}; ShowFlight)$$

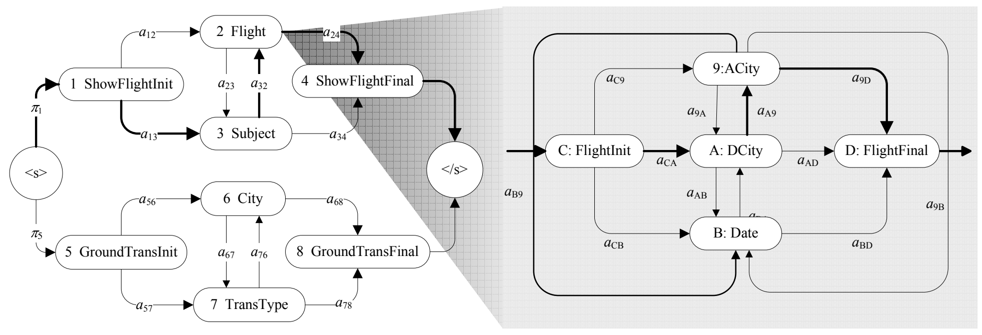


Figure 1.14 Hierarchical model topology for the semantic frames in Figure 1.2. On the left is the top level network structure. On the right is the sub-network of the semantic frame "Flight." Since the semantic frames are defined in a finite state language, the sub-network can substitute for state 2. Substitutions by the same sub-network may share the parameters of the sub-network. The thicker path shows the correct state sequence for the sentence "Show me the flight from Seattle to Boston on Christmas Eve."

$$\begin{aligned}&P(\text{DCity}|\text{<s>};\mathit{Flight})P(\text{ACity}|\text{DCity};\mathit{Flight})P(\text{</s>}|\text{ACity};\mathit{Flight})\\&P(\text{</s>}|\text{Flight};\mathit{ShowFlight})\\=\ &\pi_1 a_{13}a_{32}a_{\text{CA}}a_{\text{A9}}a_{\text{9D}}a_{24}\end{aligned}\qquad(1.9)$$

Generally,

$$P(M)=\prod_{i=1}^{|M|+1}P\left(C_M(i)\mid C_M(i-1)\right)P\left(M(i)\right)\qquad(1.10)$$

Here $|M|$is the number of the instantiated slots in *M*. $C_M(i)$ is the name of the *i*-th slot in M, ($C_M(0)$=<s> and $C_M(|M|+1)$=</s> stand for the beginning and the end of a frame, respectively) and $M(i)$ is the sub-structure that fills the *i*-th slot in M. Eq. (1.10) recursively calculates the prior probabilities of the sub-structures and includes them in the prior probability of the parent semantic structure.

Cambridge University's Hidden Vector State model (He and Young 2003) uses 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 pre-terminal nodes (the nodes immediately above the terminal words) in a semantic tree, as illustrated in Figure 1.15. The hidden vector 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 the states becomes finite, and the prior model becomes the Markov chain in an HMM. The difference from the earlier examples in Figure 1.13 and Figure 1.14 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) that pops the top n elements from the stack; and 2) Push(C) that pushes a new concept C into the stack. For example, the transition from the state represented by the fifth block in Figure 1.15 can be made with two operations: Pop(2) and Push(ON). The pop operation depends on the stack contents of the state from

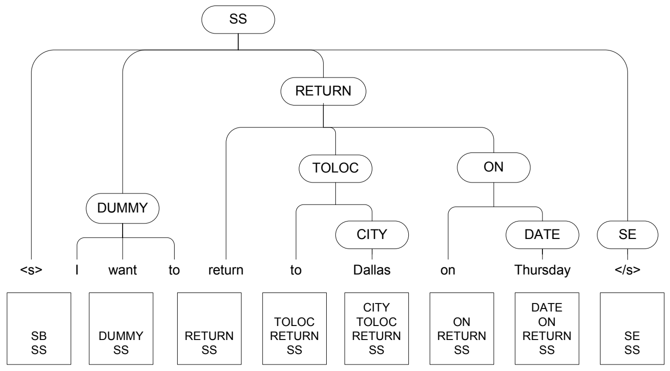


Figure 1.15 In the Hidden Vector State model, the states (illustrated by the blocks) represent the stack status of a pre-terminal nodes (the parent node of the terminal words) in the semantic tree, which is the trace from a pre-terminal (at the top of the stack) to the root of the parse tree (Reproduced from (He and Young 2003)).

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

$$P_{pop}(2\ |[\text{CITY TOLOC RETURN SS}])P_{push}(\text{ON}\ |[\text{RETURN SS}]) \tag{1.11}$$

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} in order to transform S_{t-1} to S_t. Thus the semantic prior is the product of the transition probabilities in the Markov chain:

$$\begin{aligned} P(M) &= \prod_{t=1}^{|M|} P\left(S_t \mid S_{t-1}\right) \\ &= \prod_{t=1}^{|M|} P_{pop}(|S_{t-1}| - |S_t| + 1 \mid S_{t-1}) P_{push}(TOP[S_t] \mid POP[1, S_t]) \\ &\qquad \delta(POP[|S_{t-1}| - |S_t| + 1, S_{t-1}],\ POP[1, S_t]) \end{aligned} \tag{1.12}$$

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. $\text{TOP}[S]$ is the top element of the stack S, and $\text{POP}[n, S]$ is the new stack after the top n elements are popped out of the stack S. $\delta\left(x, y\right) = 1$ if $x = y$, otherwise $\delta\left(x, y\right) = 0$. It guarantees that S_{t-1} to S_t is a legal transition by restricting 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 (Yu et al. 2003).

Lexicalization Models

In this section we review the lexicalization models under the general framework of Eq.(1.7), which captures the dependency and temporal ordering for the observed words within the same state.

N-gram Lexicalization Model

The first lexicalization model, used by both CHRONUS (Pieraccini and Levin 1993) and the Hidden Understanding Model (Miller et al. 1994), 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 Eq. (1.7):

$$\begin{aligned} P\left(W \mid M\right) &= \sum_{L} P\left(W, L \mid q_1, ..., q_m\right) = \sum_{\pi=\varphi_1,...,\varphi_m} P\left(\pi \mid q_1, ..., q_m\right) \\ &\approx \sum_{\pi=\varphi_1,...,\varphi_m} \prod_{i=1}^{m} P\left(\varphi_i \mid q_i\right) \end{aligned} \tag{1.13}$$

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

Both CHRONUS (Pieraccini and Levin 1993) and the Hidden Understanding Model (Miller et al. 1994) exploited state specific n-grams to model $P(\varphi \mid q)$. Let's use the flat-concept semantic structure in Figure 1.13 as an example. It illustrates a segmentation $\pi = \text{“show me}'',\ \text{“flights}'',\ \text{“from Seattle}'',\ \text{“to Boston}''$. The probability of the surface sentence under this segmentation is:

$$\begin{aligned} &P(\pi \mid M = \mathbf{command}, \mathbf{subject}, \mathbf{DCity}, \mathbf{ACity}) = \\ &\quad P(\text{Show}|\text{<s>};\ \mathbf{command})P(\text{me}|\text{Show}; \mathbf{command})P(\text{</s>}|\text{me}; \mathbf{command}) \\ &\quad P(\text{flights}|\text{<s>}; \mathbf{subject})P(\text{</s>}|\text{flights}; \mathbf{subject}) \\ &\quad P(\text{from}|\text{<s>}; \mathbf{DCity})P(\text{Seattle}|\text{from};\ \mathbf{DCity})P(\text{</s>}|\text{Seattle}; \mathbf{DCity}) \\ &\quad P(\text{to}|\text{<s>}; \mathbf{ACity})P(\text{Boston}|\text{to}; \mathbf{ACity})P(\text{</s>}|\text{Boston}; \mathbf{ACity}) \end{aligned} \tag{1.14}$$

In Eq. (1.14), the 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 too. In this case, instead of depending on "<s>", the initial words from a state may depend on the last few words (actual number is determined by the n-gram order) from the previous state.

Fertility Lexicalization Model

IBM's *fertility model* (Della Pietra et al. 1997) is another implementation of Eq. (1.7). It is based on their statistical machine translation work. Similar idea was adopted in (Macherey et al. 2001). The generative process of the model can be illustrated by the following 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 $S = < \mathrm{SEG}_{\mathrm{DISPLAY}}, \mathrm{SEG}_{\mathrm{FLIGHTS}}, \mathrm{SEG}_{\mathrm{TO}}, \mathrm{SEG}_{\mathrm{ARRIVAL\_CITY}}, \mathrm{SEG}_{\mathrm{DISPLAY}} >$. 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, one word "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 non-consecutive word sequences (segments).

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

1. $P\left(S \mid M = q_1, ..., q_m\right) \approx \prod_{i=1}^{m} f(n_i \mid q_i) = \prod_{i=1}^{m} \frac{e^{-\lambda_{q_i}} \lambda_{q_i}^{n_i}}{n_i!}$, 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, $n = \sum_{i=1}^{m} n_i = |S|$ is the total number of segments generated from the m states.

2. The alignment model $P\left(A = a_1, ..., a_n \mid S = s_1, ..., s_n, M = q_1, ..., q_m\right)$ follows a uniform distribution. Here a_j is the index of the state in M that the *j*-th segment is aligned to. In other words, according to A, segment s_j is aligned to the state q_{a_j}.

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

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

Given the above assumptions,

$$P(\, W, L, A, S \mid M)$$

$$= P(S \mid M)\, P(A \mid S, M)\, P(L \mid A, S, M)\, P(W \mid L, A, S, M) \tag{1.15}$$

$$= \left(\prod_{i=1}^{|M|} \frac{e^{-\lambda_{q_i}} \lambda_{q_i}^{n_i}}{n_i!}\right) \left(\frac{1}{n!} \prod_{i=1}^{|M|} n_i!\right) \left(\prod_{j=1}^{n} n\left(l_j \mid q_{a_j}\right) \prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right)\right) \tag{1.16}$$

$$= \frac{1}{n!} \prod_{i=1}^{|M|} e^{-\lambda_{q_i}} \lambda_{q_i}^{n_i} \prod_{j=1}^{n} n\left(l_j \mid q_{a_j}\right) \prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right) \tag{1.17}$$

$$= \frac{1}{n!} \prod_{i=1}^{|M|} e^{-\lambda_{q_i}} \prod_{j=1}^{n} \lambda_{q_{a_j}} n\left(l_j \mid q_{a_j}\right) \prod_{k=1}^{l_j} p\left(w_{jk} \mid q_{a_j}\right) \tag{1.18}$$

$$= \frac{1}{n!} \prod_{i=1}^{|M|} e^{-\lambda_{q_i}} \prod_{j=1}^{n} r\left(l_j, s_j | q_{a_j}\right) \tag{1.19}$$

In Eq. (1.16), the second factor is the permutation (alignment) probability, which is the inverse of the multinomial coefficient. Eq. (1.18) distributes the λ's inside the second product[1]. In Eq. (1.19), $r(l, s \mid q) \triangleq \lambda_q n\,(l \mid q) \prod_{k=1}^{l} p(w_{jk} \mid q)$ is proportional to the probability that a segment $s = w_{j1}, ..., w_{jl}$ is generated from the state q. In this form, one can sum over all possible alignments A in polynomial time:

$$\begin{aligned} P(W, L, S \mid M) &= \sum_A P(W, L, A, S \mid M) \\ &= \sum_A \frac{1}{n!} \prod_{i=1}^{|M|} e^{-\lambda_{qi}} \prod_{j=1}^{n} r(l_j, s_j \mid q_{a_j}) \end{aligned} \tag{1.20}$$

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

$$\hat{M} = \arg\max_M P(W \mid M) P(M) = \arg\max_M \sum_{S,L} P(W, L, S \mid M) P(M) \tag{1.21}$$

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

2+1 Level SLU Model

In the lexicalization model we have discusses so far, a state (attribute) is assigned to each word in an utterance, such that the utterance can be segmented into a sequence of attributes. From this a semantic representation as in Figure 1.12 can be constructed, where the original text of a segment specifies the value of the slots in the semantic frame. The text is often not in the canonical form as in Figure 1.3. It often needs a separate rule-based post-processing process to derive the normalized slot values from the text.

[1]Note that $\sum_{i=1}^{|m|} n_i = n$.

Lefèvre (2007) introduces a 2+1 SLU model that integrates the normalization process in the lexicalization model. Here the number “2” stands for the semantic prior model and the concept model, which is the traditional lexicalization model that generates the lexical string from the concept. “1” stands for the additional model that treat the normalized attribute values as a hidden variables. Ideally, a decoder can jointly find the concept sequence and the normalized value sequence according to the following decision rule:

$$\hat{C}, \hat{V} = \arg\max_{C.V} P(C, V \mid W) = \arg\max_{C.V} P(W \mid C, V) P(V \mid C) P(C) \tag{1.22}$$

Here the lexicalization model $P(W \mid C, V)$ depends not only on the attribute C, but also on the normalized value of C, V. However, the complexity of this decoding process is too high to allow an efficient algorithm, a two step “2+1” decoding process is adopted instead. In the first step, like the other frame-based SLU models we have reviewed, the semantic prior model and the concept model is applied to obtain the concept segmentation according to

$$\hat{C} = \arg\max_{C} P(C \mid W) = \arg\max_{C} P(W \mid C) P(C) \tag{1.23}$$

where $P(C)$ is the flat attribute n-gram semantic prior model, and $P(W \mid C)$ is the attribute dependent n-gram lexicalization model. Both of them are reviewed earlier. In the second, both W and the attribute segment sequences C are observed, and $\hat{V}$ can be found according to

$$\hat{V} = \arg\max_{v} P(W \mid \hat{C}, V) P(V \mid \hat{C}) P(\hat{C}) \tag{1.24}$$

To further simplify the decoding process, the attribute names in C in Eq. (1.23) and (1.24) do not include the specifiers in the MEDIA semantic representation. The specifiers are assigned to the concept sequence in a later stage with a conditional sequential labeling model, which we will review in Section 1.3.4.

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 (Pieraccini and Levin 1993) and the Hidden Understanding Model in (Miller et al. 1994), 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. However, fully annotated training data are expensive to obtain. It is desirable to label only the the words that carry the important semantic information, for example, only the slot filler words. In such a case, the state labels for many words are hidden. The Expectation-Maximization algorithm can be applied to estimate the parameters. Section 1.3.2 presents a case study of the statistical learning with the HMM/CFG composite model, a generative model assisted by the domain/linguistic knowledge.

Implementation of the Generative Models

Many generative models are implemented with standard toolkits like the stochastic finite state transducers (SFST) (Raymond and Riccardi 2007) or the graphic models (Bilmes and

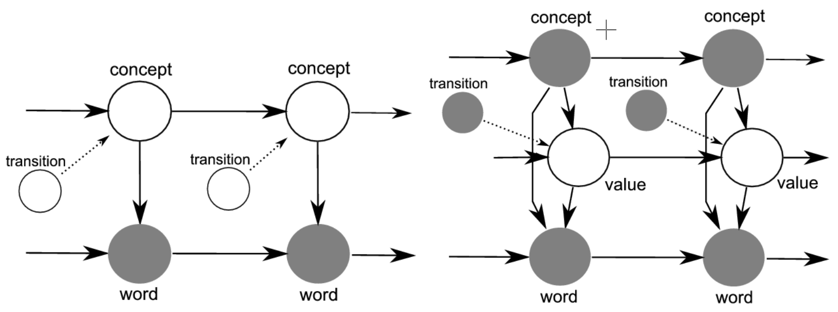


Figure 1.16 Graphic model representation of the DBNs for SLU. Left: the DBN used in first step decoding, in which the words are aligned to the concepts. Right: the DBN for second step decoding, in which the concepts and words are observed, and the normalized values for the concepts are identified. (Courtesy of Fabrice Lefèvre)

Zweig 2002). For example, each SLU component can be implemented as a SFST, and the SLU system can be built by composing the component SFSTs. Raymond and Riccardi (2007) shows a SFST implementation of a generative model. The lattice from an ASR is represented by a stochastic finite state machine (SFSM) λ_W. It uses an n-gram as the lexicalization model for "concepts", which are slots or a null attribute that models the carrier phrases connecting the slots. N-gram can be represented by a SFSM as well, as described in (Riccardi et al. 1996). An n-gram SFSM for a concept can be turned into a SFST by outputting the accepted words together with the concept's name. The union of all the SFSTs for all concepts forms the lexicalization model λ_{w2c} that maps words to concepts. Finally, a statistical conceptual language model is used as the semantic prior model. The SFST model λ_{CLM} is flexible enough to implemented different semantic prior models. In the case in (Raymond and Riccardi 2007), the conceptual language model is used to compute the joint probability $P(W, M)$ instead of $P(M)$:

$$P(W, M) = \prod_{i=1}^{k} P(w_i,\ c_i \mid h_i) \tag{1.25}$$

where $M = c_1, c_2, \ldots, c_k$ is a sequence of concepts, $W = w_1, w_2, \ldots, w_k$ is a sequence of words, and $h_i = w_{i-1}c_{i-1}, w_{i-2}c_{i-2}$ is the trigram history of word/concept pairs. The SLU model is thus a SFST composition:

$$\lambda_{SLU} = \lambda_W \circ \lambda_{w2c} \circ \lambda_{CLM} \tag{1.26}$$

The generative models can also be easily implemented with the general purpose graphic model toolkit. For example, researchers from Université d'Avignon used Dynamic Bayesian Networks (DBNs) (Lefèvre 2007; Meurs et al. 2009) to implement generative SLU models. The aforementioned "2+1 SLU" model can be represented with the graphic models in Figure 1.16, which is implemented with the generic graphic model toolkit GMTK (Bilmes and Zweig 2002) and the language model toolkit SRILM (Stolcke 2002).

1.3.2 Integrating Knowledge in Statistical Models – A Case Study of the Generative HMM/CFG Composite Model

One disadvantage of a purely data-driven, statistical SLU approach 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 (a) replacing the words in a semantic class with the class name; e.g., "Seattle" and "Boston" are replaced with the superword "_cityname"; and (b) 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:

1. As it will be discussed later, it is beneficial to use the SLU model as the language model for speech recognition. However, with actual words being replaced by the superwords, many of these are modeled with complicated CFGs instead of a word list as in class-based language models (Brown et al. 1992; Kneser and Ney 1993), the model can no longer be used for speech recognition.
2. 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 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 toward the first interpretation.

This section presents a case study of the HMM/CFG composite model. The case study serves two purposes. First, it shows an effective way to seamlessly integrate the domain-dependent knowledge in a data-driven, statistical learning framework to improve the SLU performance. Second, it describes the application of important optimization algorithms for the generative models in SLU, in particular, the EM algorithm, with rigorous mathematic derivations – until now, we have mentioned the EM algorithm several times without detailed descriptions.

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 predict the words for speech recognition, and it 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 lexicalization model has been introduced in (Wang and Acero 2003a), which we review here, to aim at solving these problems. This model uses the same semantic prior for hierarchical Markov topology as in Eq. (1.10). The underlying state corresponding to a slot in the semantic frame is expanded into a preamble-filler-postamble three state sequence

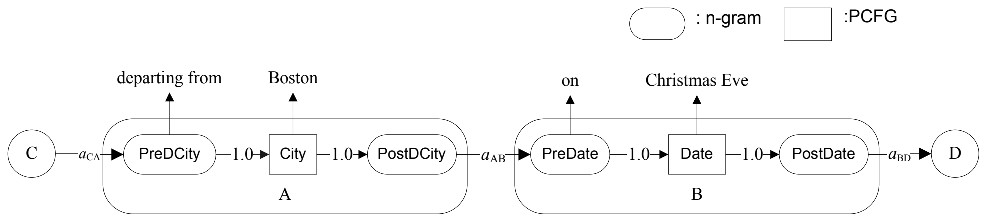


Figure 1.17 State alignment for the phrase “departing from Boston on Christmas Eve” according to the topology in Figure 1.14. The original states A and B are expanded to three states.

(Figure 1.17). 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 Eq. (1.13), similar to CHRONUS and the Hidden Understanding model. The differences here include:

1. The HMM/CFG composite model 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 the domain-specific and domain-independent knowledge, like a city-name list and a date grammar. The CFG rules can be populated with database entries or pre-built in a grammar library for domain-independent concepts, for example, CFG rules for date and time. The lexicalization of other states is modeled with the n-grams. Figure 1.17 shows the state alignment for an example phrase according to the network topology in Figure 1.14. 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.
2. The training data for the composite model is not required to be fully annotated – not every word is associated with a state. Instead, only a high level semantic annotation is required, which labels the slots of semantic frames. For example, the utterance “show flights departing from Seattle flying to Boston on Christmas Eve” is labeled as in Figure 1.12. The words without explicit labels, for example, “on”, may be associates with different states in the topology, for example, either “PostDCity” or “PreDDate” in Figure 1.17. Because the alignments are hidden for those unmarked words, EM algorithm (Dempster et al. 1977) needs to be applied for model training.

Formally, the lexicalization probability in this composite model is

$$P(W \mid M) = \sum_{\pi=\varphi_1,\ldots,\varphi_m} P(\pi \mid q_1, \ldots, q_m) = \sum_{\pi=\varphi_1,\ldots,\varphi_m} \prod_{i=1}^{m} P_{q_i}(\varphi_i \mid q_i) \tag{1.27}$$

Here P_q is an n-gram model if q is a preamble or a postamble state, or 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.

Modeling slot fillers with PCFG has another advantage – since the slot fillers are the bearers of semantic information, the capability of semantic interpretation of PCFG introduced by the semantic interpretation tags allows compositional construction of normalized semantic representation. For example, both “two fifteen PM” and “quarter past two in the afternoon” may have the same semantic representation like

```
<PreciseTime>
    <TimePoint>2:15</TimePoint>
    <TimeOfDay>PM</TimeOfDay>
</PreciseTime>
```

such that application developers do not have to design an extra post-processing step to specifically process the two literally different phrases.

Parameter Estimation

The composite model can be formalized as (S, A, G), where S 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. S is determined by the semantic frame. Parameters of A and the n-gram parts of G have to be estimated from the manually annotated training data as shown in Figure 1.12, in which only the semantic salient information, i.e., frames and slot fillers, is marked. The annotation determines the state sequence in the model topology[2], as illustrated by Figure 1.17. Therefore, the counts of state transitions can be collected directly, and the ML estimation is applied for the state transition probabilities. Here a prior count c is used to smooth the transition distributions, which 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 (Rabiner and Juang 1993). Note that in discrete HMM training, the posterior

$$\gamma_t(q) = P_\phi(q_t = q \mid W) = \frac{\alpha_t\,(q)\,\beta_t\,(q)}{\sum_{q'} \alpha_t\,(q')\,\beta_t\,(q')} \tag{1.28}$$

can be calculated with the Forward-Backward algorithm. And the emission probability can be then estimated by:

$$b_q(w) = \sum_{t:w_t=w} \gamma_t(q) \bigg/ \sum_{t=1}^{T} \gamma_t(q) \tag{1.29}$$

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 $\xi_t(q,r) = P(q_{t-1} = q, q_t = r \mid W)$ can also be calculated with the Forward-Backward algorithm:

$$\xi_t(q,r) = P(q_{t-1} = q, q_t = r \mid W)$$

[2]Note that some states may be a dummy state that emits an empty string.

$$= \frac{\alpha_{t-1}(q)\, t(q,r,w_{t-1}) P_r(w_t \mid h(q,r,w_{t-1}))\, \beta_t(r)}{\sum_{q'} \sum_{r'} \alpha_{t-1}(q')\, t(q',r',w_{t-1}) P_{r'}(w_t \mid h(q',r',w_{t-1}))\, \beta_t(r')} \tag{1.30}$$

$$h(q,r,w) = \begin{cases} \text{<s>} & \text{when } q \neq r \\ w & \text{when } q = r \end{cases} \tag{1.31}$$

$$t(q,r,w) = \begin{cases} P_q(\text{</s>} \mid w) a_{qr} & \text{when } q \neq r \\ 1 & \text{when } q = r \end{cases} \tag{1.32}$$

There are several differences from the standard HMM training here. 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

$$P_q(v \mid w) = \frac{\sum_{t:w_{t-1}=w, w_t=v} \xi_t(q,q)}{\sum_{t:w_{t-1}=w} \xi_t(q,q)} \tag{1.33}$$

for $w \neq$ <s>, and

$$P_q(v \mid \text{<s>}) = \frac{\gamma_1(q)\delta(w_1,v) + \sum_{r\neq q} \sum_{t:w_t=v} \xi_t(r,q)}{\gamma_1(q) + \sum_{r\neq q} \sum_t \xi_t(r,q)} \tag{1.34}$$

where $\delta(w,v) = 1$ if $w = v$ and 0 otherwise.

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:

$$P_q(w_k \mid w_{k-n+1},...,w_{k-1}) = \frac{\sum_\varphi N(q,\varphi) C(w_{k-n+1},...,w_{k};\varphi)}{\sum_\varphi N(q,\varphi) C(w_{k-n+1},...,w_{k-1};\varphi)} \tag{1.35}$$

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 Figure 1.12, $N(q,\varphi)$ can be viewed as the expected fractional count the state q generate the string φ. Notice that the annotation pegs the slot fillers to the slot states, which restricts the alignments of the remaining words and states, so a subsequence of the observation W (e.g., W ="departing from") can only align to a state subsequence Q (e.g., Q ="PostSubject PreFlight PreACity" in Figure 1.12). The counts of these subsequence alignments, $c(Q,W)$, can be empirically collected from all the annotated training examples. Then the state-specific n-gram parameters are estimated with the EM algorithm (Dempster et al. 1977). In the E-step, $N(q,\varphi)$, the fractional count that state q is aligned to the substring φ, is computed with the current model parameters (initially based on the uniform distribution). In the M-step, Eq. (1.35) is applied to estimate the n-gram probability (with proper smoothing such as deleted interpolation (Jelinek and Mercer 1980)]).

To compute $N(q, \varphi)$, the expected count for state q to generate segment φ, note that

$$N(q, \varphi) = \sum_{Q,W} c(Q, W) \sum_{\pi} P_\phi(\pi \mid Q, W) c(q \uparrow \varphi; \pi, Q, W) \tag{1.36}$$

here $c(Q, W)$ is the number of occurrences that the state sequence $Q = q_1, q_2, ..., q_m$ co-occurs with the word sequence $W = w_1, ..., w_k$ according to the partially labeled training data. π is a segmentation that breaks W into m non-overlapping 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 segmentation π.

The summation over all possible segmentation in Eq. (1.36) makes it difficult to compute $N(q, \varphi)$ efficiently. To remove the summation, note that because $P_\phi(\pi, Q, W) = \prod_{q,\varphi} P_\phi(\varphi \mid q)^{c(q\uparrow\varphi;\pi,Q,W)}$,

$$\begin{aligned}
\frac{\partial P_\phi(Q, W)}{\partial P_\phi(\varphi \mid q)} &= \frac{\partial \sum_\pi P_\phi(\pi, Q, W)}{\partial P_\phi(\varphi \mid q)} \\
&= \frac{\partial \sum_\pi \prod_{(q,\varphi)\in\pi} P_\phi(\varphi \mid q)^{c(q\uparrow\varphi;\pi,Q,W)}}{\partial P_\phi(\varphi \mid q)} \\
&= \sum_\pi \frac{c(q \uparrow \varphi; \pi, Q, W) \prod_{q',\varphi'} P_\phi(\varphi' | q')^{c(q'\uparrow\varphi';\pi,Q,W)}}{P_\phi(\varphi \mid q)}
\end{aligned} \tag{1.37}$$

Rearranging Eq. (1.37), we obtain

$$\sum_\pi P_\phi(\pi \mid Q, W) c(q \uparrow \varphi; \pi, Q, W) = \frac{P_\phi(\varphi \mid q)}{P_\phi(Q, W)} \frac{\partial P_\phi(Q, W)}{\partial P_\phi(\varphi \mid q)}. \tag{1.38}$$

On the left of (1.38) is the summation that occurs in the expected count. On the right there is no summation at all. The problem now becomes efficiently computing $P_\phi(Q, W)$ and $\frac{\partial P_\phi(Q,W)}{\partial P_\phi(\varphi \mid q)}$. For that purpose, define $\alpha_k(i) = \Pr(\pi = \varphi_1, ..., \varphi_m \ \wedge \varphi_k = ..., w_i; W \mid Q)$ to be the probability of all the segmentations π that align the end of q_k in Q to the i-th word in W, and $\beta_k(i) = P(\pi = \varphi_1, ..., \varphi_m \ \wedge \varphi_k = w_i ...; W \mid Q)$ to be the probability of all the segmentations that align the beginning of q_k to the i-th word in W. Then

$$\begin{aligned}
P_\phi(Q, W) &= \alpha_m(n) P_\phi(Q) \\
&= P_\phi(Q) \sum_{ij} \alpha_{k-1}(i-1) P_\phi(w_i, ..., w_j \mid q_k) \beta_{k+1}(j+1), \forall k.
\end{aligned} \tag{1.39}$$

According to (1.39),

$$\frac{\partial P_\phi(Q, W)}{\partial P_\phi(\varphi \mid q)} = P_\phi(Q) \sum_{\substack{k \,:\, q_k = q \\ ij \,:\, \varphi = w_i \ldots w_j}} \alpha_{k-1}(i-1) \beta_{k+1}(j+1) \tag{1.40}$$

Combining Eqs. (1.38), (1.39) and (1.40), the expected count

$$N(q,\varphi) = P_\phi(\varphi \mid q) \sum_{Q,W} \frac{c(Q,W)}{\alpha_{|Q|}(|W|)} \sum_{\substack{k\,:\,q_k = q \\ ij\,:\,\varphi = w_i \ldots w_j}} \alpha_{k-1}(i-1)\beta_{k+1}(j+1) \tag{1.41}$$

$\alpha_k(i)$ and $\beta_k(i)$ can be computed efficiently with dynamic programming according to Eq. (1.42):

$$\begin{aligned} \alpha_0(i) &= P_\phi(w_1,...,w_i \mid q_0); \\ \alpha_k(i) &= \sum_{r \le i} \alpha_{k-1}(r) P_\phi(w_{r+1},...,w_i \mid q_k); \\ \beta_m(i) &= P_\phi(w_i,...,w_n \mid q_m); \\ \beta_k(i) &= \sum_{r \ge i} \beta_{k+1}(r) P_\phi(w_i,...,w_{r-1} \mid q_k) \end{aligned} \tag{1.42}$$

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

The performance of the HMM/CFG Composite Model

The HMM/CFG composite model balances the trade-off between robustness and the constraints on over-generalizations/ambiguities with the different models for the preambles/postambles and the 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, like the city names and airport names in the ATIS domain; or they are common domain-independent concepts like phone number, date, time, which are already modeled in a grammar library; or they can be automatically generated according to some high level description like a regular expression for an alphanumeric concept (Wang and Acero 2006). The non-slot states serve as the "glue" that sticks different slot fillers together. This type of inter-concept language is normally domain dependent, hard to pre-build 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 sub-language. 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 context in the training data).

Figure 1.18 plots the slot error rate of the HMM/CFG composite model with the ATIS 93 category A test set, 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 (˜1700 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 model that used all the ATIS2 and ATIS3 training data

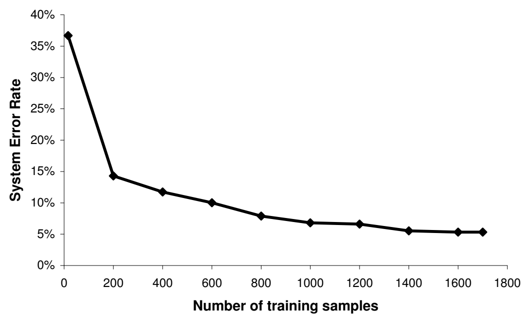


Figure 1.18 ATIS end-to-end system error rate on text input vs. training set size for the HMM/CFG composite model.

(over 6000 sentences). The inclusion of domain knowledge in the statistical model reduced requirement on training data.

1.3.3 Use of Generative Understanding Models in Speech Recognition

Sub-optimality of Two-pass SLU

Eq. (1.6) assumes that the sentence or word sequence *W* is the observation. This is only true for typed-in language. However, for spoken language understanding, the observation is the acoustic observation sequence *O*. Hence the optimal meaning for a speech utterance *O* is

$$\hat{M} = \arg\max_M P(M \mid O) = \arg\max_M P(O \mid M)P(M) \tag{1.43}$$

Eq. (1.43) is often 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 the recognizer. In the second pass, the corresponding meaning $\hat{M}$ is extracted from $\hat{W}$ by plugging $\hat{W}$ into Eq. (1.6):

$$\begin{aligned}\hat{M} &= \arg\max_M P(M \mid \hat{W}) = \arg\max_M P\left(M \mid \arg\max_w \left(P(O \mid W)P(W)\right)\right)\\ &= \arg\max_M P(M)P\left(\arg\max_w \left(P(O \mid W)P(W)\right)|M\right)\end{aligned} \tag{1.44}$$

An alternative to this two-pass solution is shown in Eq. (1.45):

$$\begin{aligned}\hat{M} &= \arg\max_M \Pr(M \mid O) = \arg\max_M P(O \mid M)P(M)\\ &= \arg\max_M \sum_W P(O, W \mid M)P(M)\end{aligned} \tag{1.45}$$

Here the understanding model $P(W \mid M)P(M)$, which can be considered as meaning-specific "language model", is directly used in place of the generic language model in ASR.

The two-pass solution is simpler from the 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 sub-optimal 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 to adapt to changes. If new cities are added into a database, or a new type of services is introduced, the language model in a two-pass system has to be retrained unless a class-based language is used.

Using an 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 unification grammar, they can be converted to a CFG for ASR (Moore 1998; Rayner et al. 2001). However, 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 the speech recognizers.

In statistical SLU, robustness is built into the model itself through proper smoothing. The remaining problems have more of an engineering nature – how can a statistical model like the HMM/CFG composite model be converted into a format that a recognizer can take as a language model. In (Wang and Acero 2003b) the HMM/CFG composite model is converted to a CFG as follows: the backbone Markov chain is basically a statistical finite state machine, which is a sub-class of the PCFG. The efficient conversion of the n-gram observation model follows the work in (Riccardi et al. 1996), and the CFG observation model is used directly. The composite model, in the format of PCFG, was applied under the framework of Eq. (1.45), and the results were compared with the two-pass recognition/understanding paradigm under the framework of Eq. (1.44), 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 1.1 shows the findings with a commercial decoder and a research decoder. For the commercial decoder, even though the composite model's word error rate is over 46% higher than the trigram model, its SLU error rate (again measured as the slot insertion-deletion-substitution rate) is 17% lower. With the research decoder that is less aggressive in pruning, the word error rate 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 sub-optimality 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 word error rate. 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

Table 1.1 The ASR word error rate and the SLU error rate (slot ins-del-sub) of the trigram model (2-passes) and the HMM/CFG composite mode (1-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	WER	8.2%	12.0%	
Decoder	SLUER	11.6%	9.8%	5.1%
Research	WER	6.0%	7.6%	
Decoder	SLUER	9.0%	8.8%	5.1%

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 still meaningful because unannotated training data is much easier to obtain than the annotated data. When only the 1700 training samples were used for LM training, the two-pass system had 10.4% WER and 13.1% SLUER with the commercial recognizer, and 7.5% WER and 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 the acoustics and the semantics. In (Riccardi and Gorin 1998), 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 (Chelba et al. 2003), a single pass ASR/ACR system, in which the ACR statistical model was used as the language model for ASR as well, resulted in worse word error rate but better call classification accuracy. In (Estève et al. 2003), a concept decoder that adopted a model similar to the HMM/CFG model also yielded better understanding results.

1.3.4 Conditional Models

The statistical models for SLU we have introduced so far are all generative models – the semantic structure M is first generated according to the semantic prior model $P(M)$, from which the observation W is generated according to $P(W \mid M)$, which an be modeled by the different lexicalization processes we just described.

Conditional models are non-generative. In a conditional model, the states (which encode the meaning M) are directly conditioned on the observation. For SLU, Conditional Random Fields (CRFs) or Hidden State Conditional Random Fields (HCRFs) are commonly used conditional models. With CRFs or HCRFs, the conditional probability of the entire state (label) sequence **y** given the observation sequence is modeled as an exponential (log-linear) distribution, with respect to a set of features $f_k(\mathbf{y}, \mathbf{x})$. The features are functions of the observation sequence **x** and the associated label sequence **y**:

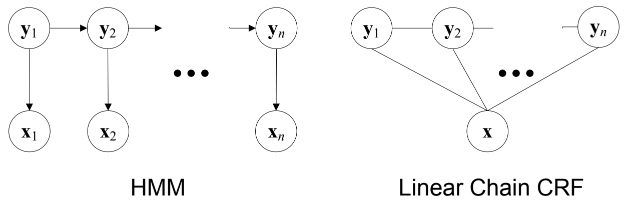


Figure 1.19 Graphic model representations for HMMs (generative model) and CRFs (conditional model). A generative model is a directed graph that points to the direction of the generative process, and observations are generated frame by frame in a uniformed way. Conditional models are represented by undirected models, where the entire observation sequence $\mathbf{o}$ is observable for every states in the model.

$$P(\mathbf{y} \mid \mathbf{x}; \Lambda) = \frac{1}{Z(\mathbf{x}; \Lambda)} \exp\left\{\sum_k \lambda_k f_k\left(\mathbf{y}, \mathbf{x}\right)\right\} \tag{1.46}$$

Here $\Lambda = \{\lambda_k\}$ is a set of parameters. The value of λ_k determines the impact of the feature $f_k(\mathbf{y}, \mathbf{x})$ on the conditional probability. $Z(\mathbf{x}; \Lambda) = \sum_{\mathbf{y}} \exp\{\sum_k \lambda_k f_k\left(\mathbf{y}, \mathbf{x}\right)\}$ is a partition function that normalizes the distribution. Given a set of m labeled training examples ($\mathbf{x}_1$, $\mathbf{y}_1$) $\ldots$($\mathbf{x}_m$, $\mathbf{y}_m$), the model is trained to optimize the following objective function:

$$\begin{aligned} L(\Lambda) &= \frac{1}{m}\sum_{i=1}^{m} \log P(\mathbf{y}_i \mid \mathbf{x}_i; \Lambda) - \frac{1}{2\sigma^2}\left\|\Lambda\right\|^2 \\ &= \mathbb{E}_{\hat{P}(\mathbf{x},\mathbf{y})} \log P(\mathbf{y} \mid \mathbf{x}; \Lambda) - \frac{1}{2\sigma^2}\left\|\Lambda\right\|^2 \end{aligned} \tag{1.47}$$

where $\hat{P}(\mathbf{x}, \mathbf{y})$ stands for the empirical distribution of the labeled training samples.

The second term in Eq. (1.47) regularizes the parameters to keep them from taking extreme values, thus prevents the model from over-fitting the training data. Note that the objective function is a convex function, so a single global optimum exists.

The CRF in Eq. (1.46) is unconstrained in the sense that the feature functions are defined on the entire label sequence **y**. Because the number of all possible label sequences is combinatorial, the model training and inference of an unconstrained CRF is very inefficient. Because of that, it is common to restrict attention to the linear-chain CRFs (Lafferty et al. 2001). The linear chain CRFs impose a Markov constraint on the model topology, and as a consequence, restrict the feature functions to depend only on the labels assigned to the current and the immediately previous states, in the form $f(y_{t-1}, y_t, \mathbf{x}, t)$. The restrictions enable the application of efficient dynamic programming algorithms in model training and inference – yet still support potentially interdependent features defined on the entire observation sequence **x**.

Figure 1.19 shows the difference between the HMMs and the linear chain CRFs in graphic model representations. First, since HMMs are generative models, the graph is directed, which points to the direction of generative process. States are chosen depending on the previous states, and observations are generated at the states. CRFs, on the other hand, are represented by undirected graphs since they are direct, non-generative models. Second, observations in

HMMs are generated one at a time in a uniformed way, while in CRFs the entire observation sequence is given, so at each state it can employs features that depends on the entire observation sequence.

When the state sequence **y** is not fully observable, as in the previous example where only slot fillers are manually labeled and the remaining word can be aligned to different preamble/postambles, HCRFs can be used. They treat the unknown state assignment as hidden variables, and their features can be defined on these hidden variables. In this case, the following conditional probability is used instead in the objective function in Eq. (1.47):

$$P(\mathbf{l} \mid \mathbf{x}; \Lambda) = \frac{1}{Z(\mathbf{x}; \Lambda)} \sum_{\mathbf{y}:\mathbf{y} \text{ is consistent with } \mathbf{l}} \exp\left\{ \sum_k \lambda_k f_k(\mathbf{y}, \mathbf{x}) \right\} \tag{1.48}$$

where **l** is the partial label marked for **x** that only partially determines the state sequence for **x**. The objective function sums over all legal state sequences **y** that are consistent with **l**. Because of this additional summation, the objective function is no longer convex in Λ.

The parameters Λ in CRFs or HCRFs can be optimized according to the objective function with numeric algorithms like stochastic gradient decent or L-BFGS. Both of them use the gradient of the objective function, which can be easily derived as

$$\frac{\partial L(\Lambda)}{\partial \lambda_k} = \mathbb{E}_{\hat{P}(\mathbf{x},\mathbf{y})}\left[f_k(\mathbf{x}, \mathbf{y})\right] - \mathbb{E}_{\hat{P}(\mathbf{x})P(\mathbf{y}|\mathbf{x})}\left[f_k(\mathbf{x}, \mathbf{y})\right] \tag{1.49}$$

for CRFs and

$$\frac{\partial L(\Lambda)}{\partial \lambda_k} = \mathbb{E}_{\hat{P}(\mathbf{x},\mathbf{l})P(\mathbf{y}|\mathbf{x},\mathbf{l})}\left[f_k(\mathbf{x}, \mathbf{y})\right] - \mathbb{E}_{\hat{P}(\mathbf{x})P(\mathbf{y}|\mathbf{x})}\left[f_k(\mathbf{x}, \mathbf{y})\right] \tag{1.50}$$

for HCRFs. In other words, the gradient with respect to a feature weight is the difference between the expected value of the feature given both the observation and the label sequence and the expected value of the feature given only the observation and a model $P(\mathbf{y} \mid \mathbf{x})$ trained so far. These expected value can be collected with the forward-backward algorithm when the linear-chain CRFs or HCRFs are used.

Compared with the generative models, conditional models have the following advantages:

1. A generative model wastes its capacity by modeling the generation of observations, which in practice are always known. Because of that, several discriminative training algorithms like maximum mutual information (MMI) (Bahl et al. 1986) and minimum classification error (MCE) (Juang et al. 1997) have been proposed. Conditional models, on the other hand, are discriminative in nature – its objective function can only be optimized by maximizing the posterior probability of the correct state sequence while minimizing the posterior of the competing hypotheses.
2. The components of a generative model, such as the transition and emission distributions, need to be a properly normalized probability density functions, even though the only probability distribution we are interested in for a SLU task is $P(\mathbf{y} \mid \mathbf{x})$. This makes the model too restrictive. In contrast, CRFs models the entire state sequence with the exponential model and the only restriction is that $P(\mathbf{y} \mid \mathbf{x})$ forms a proper probabilistic distribution. In the specific case of HMM and CRF, if the same features (transition and emission) are used, then the space of the HMM parameters

list	**twa**	**flights**	**from**	**washington**	**to**	**philadelphia**
|	|	|	|	|	|	|
null	*AirlineCode*	*null*	*null*	*ACity*	*null*	*DCity*

Figure 1.20 Frame-based SLU as a sequential labeling problem: words are assigned labels representing their meaning.

values is limited to a subset of that of the CRF due to the constraints that is not relevant to the objective function. This extra constraints may limit the model's discriminative capability.

3. A generative model has uniform observation space for each state. It is very expensive to expand the observation context at a state due to the curse of dimensionality – since it needs to model the generative process for all possible such expanded context. In contrast, since the observation are given in a conditional model, it can incorporate interdependent, overlapping features only observed in the training data. The features can be defined on the entire observation, as illustrated by Figure 1.19.

In a straightforward application of CRFs for the frame-based SLU, **x** consists of a transcribed utterance, and **y** is the semantic label sequence assigned to **x**. Non-slot filler words are assigned a null state, as illustrated by Figure 1.20.

Typical features used by the CRFs for SLU include the transition features, the n-gram features, and the class membership (a.k.a. word list) features. The transition features model the dependency among adjacent states, which capture the influence of context on the tag assigned to a given word:

$$f_{i,j}^{TR}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(y_{t-1} = i)\delta(y_t = j) \tag{1.51}$$

where i, j are states (labels) of the model. $\delta(e) = 1$ when $e =$ **true** and $\delta(e) = 0$ when $e =$ **false**.

The n-gram features often include the unigram $f_{w,j}^{UG}$ and the bigram $f_{w,w',j}^{BG}$ features to capture the relation between the current state and the identity of the current (and previous) words

$$f_{w,j}^{UG}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_t = w)\delta(y_t = j) \tag{1.52}$$
$$f_{w,w',j}^{BG}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_{t-1} = w)\delta(x_t = w')\delta(y_t = j)$$

where w, w' are words and j is a model state (label).

The class member features check if the current word is part of an entry in the lexicon of a syntactic or semantic class:

$$f_{C,j}^{SC}\left(y_{t-1}, y_t, \mathbf{x}, t\right) = \delta\left(C \triangleright [\mathbf{x}, t]\right)\delta\left(y_t = j\right) \tag{1.53}$$

where C is a class, $C \triangleright [\mathbf{x}, t]$ indicates that a member of C is a substring of **x** that covers x_t. j is a model state. Here the member of C can be defined by either a phrase list, a finite state machine (equivalently a regular expression) or a context free grammar. For example, $f_{\text{TIME,DTIME}}^{SC}\left(y_{t-1}, y_t, \mathbf{x}, t\right) = 1$ when **x** = "I need to fly to Boston three o'clock in the

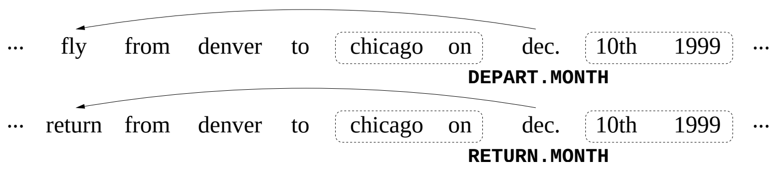

··· fly from denver to chicago on dec. 10th 1999 ···
DEPART.MONTH
··· return from denver to chicago on dec. 10th 1999 ···
RETURN.MONTH

Figure 1.21 Example of long distance dependency in Air Travel data: the interpretation of "dec." depends on the word (pointed to by the arrows) beyond the range covered by the local features. (Courtesy of Minwoo Jeong & Gary Geunbae Lee).

afternoon," "three o'clock in the afternoon" is recognized as a TIME expression, $t \in [7, 11]$ and the model is assigning the state DTIME to position t.

In (Raymond and Riccardi 2007), this approach is compared with the SFST generative model, using the same feature set – in this case, the transition feature depends not only on the adjacent states but also on the words that the states are assigned to. It is found that SFST is more robust when data is sparse and inconsistently labeled, while CRFs surpassed SFST when more labeled data are available. However, directly porting the topology and features of a generative SFST to a conditional model may not be the most useful thing to try. Although it takes advantage of the discriminative nature of the model, the model's capability to incorporate interdependent, overlapping features are not leveraged. Jeong and Lee (2008) shows a good example of such features introduced to improve the SLU accuracy. It noticed that long distance dependency is an important issue in SLU, as illustrated by Figure 1.21.

To overcome this problem, Jeong and Lee (2008) introduces trigger features[3]. An element a is a *trigger* for another element b ($a \rightarrow b$) if a is significantly correlated to b, and b is called the *trigged element*. Here an element can be a word, an attribute associated with a word (e.g., the part-of-speech tag of a word), or a model state. a and b forms a *trigger pair*. For example, the trigger pair "return $\rightarrow$ dec." indicates that the currently observed word is "dec." and there is a word "return" in the observation that is at least two words away (long distance) before the current word. The trigged element can be null (ϵ) when the identity of the current word is not important. For example, "return $\rightarrow \epsilon$" states that there is a word "return" that is long distance away from the current word.

Given a trigger pair $a \rightarrow b$, a CRF feature function can be define on the pair:

$$f^{TRI}_{a \rightarrow b,j,k}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_t \sim b)\delta(\exists_{t'<t-2} x_{t'} \sim a)\delta(y_{t-1} = j)\delta(y_t = k) \qquad (1.54)$$

where $x \sim a$ means that the x is consistent with the element a – if a is a word, it means that $w = a$. If a is an attribute like a POS tag, it means that w is assigned the part-of-speech a. $x \sim \epsilon$ is always true. For the state variables, $y_i = \epsilon$ is also always true – it means that the feature does not care about the identity of a state.

Note that although by definition a trigger pair can be a pair of model states separated by a long distance, they can not be incorporated in a linear chain CRF because it breaks the Markov assumption. Since the entire observation is known in a conditional model, this is not a problem when the trigger pair is defined on the observation.

[3]Similar idea has been investigated in language modeling (Rosenfeld 1996).

An example can illustrate the power of the trigger features. The feature

$$f^{TRI}_{\text{return}\to\epsilon,\epsilon,\text{RETURN.MONTH}}(y_{t-1}, y_t, \mathbf{x}, t) \tag{1.55}$$

is activated (has value 1) when t is at the position of the word "dec." in Figure 1.21 and the hypothesized state for the position is "RETURN.MONTH." If this feature has occurred in the training data, namely there is at least one training sample where a word labeled as RETURN.MONTH is preceded by "return" long distance away, this feature will likely be associated with a positive weight and help generalizing into other cases where the word "return" precedes a month name, as in the example in Figure 1.21.

However, the use of the trigger features introduces another problem. There are too many potential trigger pairs that are observed in the training data – most of them may not satisfy the requirement of strong correlation. If all those potential pairs are introduced into the model, it will not only slow down the training speed, but also make the model easy to over-fit the training data. Therefore a feature selection mechanism is necessary in this case. Jeong and Lee (2008) uses a feature selection algorithm adopted and modified from the one in (McCallum 2003). It is basically a greedy hill-climbing algorithm that iteratively tries each possible individual feature, check how much gain it brings to the objective function of the CRF when it is added to the model, and select the one that results in the most gain. This is, again, a very expensive procedure since it needs to retrain the CRF for each addition of features. However, the procedure can be approximated in several different ways to speed up significantly. Detailed descriptions of the feature selection algorithms is beyond the scope of this introductory book. Interested readers can find the relevant information in (McCallum 2003) and (Jeong and Lee 2008).

Figure 1.22 compares the recall-precision curve of the models with or without long distance features with different types of inputs to the SLU components. It demonstrates significant improvement on both text and speech (one-best ASR and n-best ASR) inputs. The paper also compared the conditional models with the HVS generative model. Both CRFs models with or without the long distance features outperformed the generative model.

Motivated by the success of the HMM/CFG composite model, Wang et al. (2006) investigated the unified CFG/CRF model. To model the long distance dependency, it introduces links between non-adjacent nodes in the graphical model, as illustrated by Figure 1.23.

Note that the nodes are fully connected to each other in Figure 1.23. Exact model inference via dynamic programming will no longer be possible if there is no additional constraint. CFG comes to the rescue by introducing the additional constraint that such a long distance link is only allowed between the random variables $\mathbf{y}_j$ and $\mathbf{y}_k$ when they have the values representing the same slot, and there is a CFG rule $R \in R(y)$, such that $R \overset{*}{\Rightarrow} \mathbf{x}_j, \ldots \mathbf{x}_k$. Here $R(y)$ stands for the set of grammar rules that is compatible to the label y (e.g., "PreciseTime" is compatible to the label "StartTime"). This is equivalent to introducing a feature of "no compatible CFG rules (NCR)" to the model in Figure 1.23 which always has the weight $-\infty$:

$$f^{NCR}_{R,j}(y, y', \mathbf{x}, j, k) = \delta\left(S(y) \neq S(y') \vee \nexists R \in R(y) : R \overset{*}{\Rightarrow} \mathbf{x}_j, \ldots, \mathbf{x}_k\right) \tag{1.56}$$

where $S(y)$ stands for the slot that the label y represents. This constraint of CFG rule matching enables dynamic programming for model training/inference when the observation

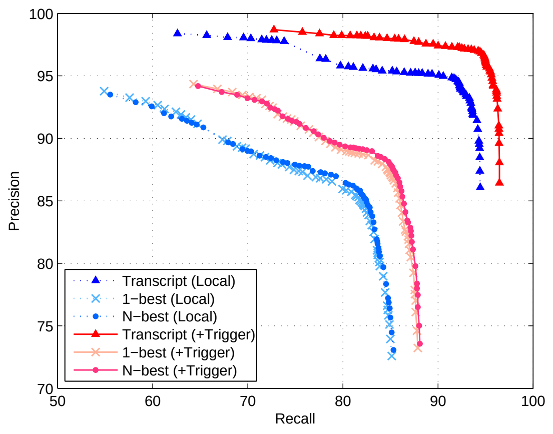


Figure 1.22 The recall-precision curves on Air Travel data used by the Communicator system (Courtesy of Minwoo Jeong & Gary Geunbae Lee).

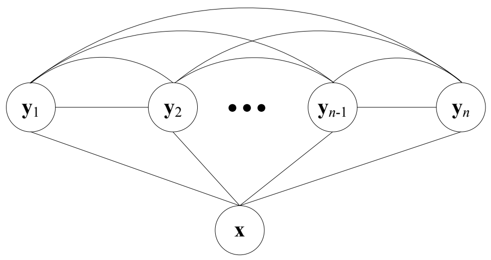


Figure 1.23 The graphic model representation of the unified CFG/CRF model. The nodes are interconnected to each other according to the spans of CFG rule coverage.

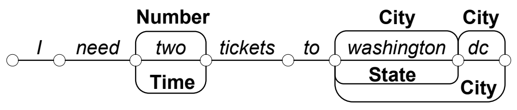


Figure 1.24 The observation includes a word sequence and the subsequences covered by CFG non-terminals.

is chunk parsed into a lattice like the one in Figure 1.24, which consists of not only a word sequence w_1^τ but also a list of CFG non-terminals (NT) that span different segments of w_1^τ.

The task of SLU becomes to select a path from the lattice and assign a semantic label to a segment – the label has to be consistent with a CFG nonterminal covering that segment. To do so, the model must be able to resolve several kinds of ambiguities:

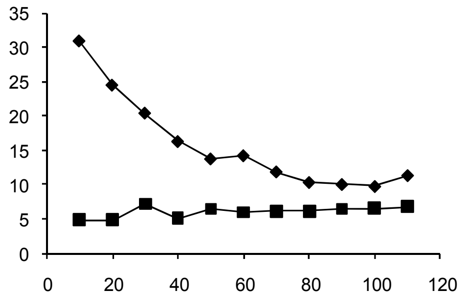


Figure 1.25 Test set slot error rates (in %) at different training iterations. The top curve is for the flat start initialization, the bottom for the generative model initialization.

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 ACity slot.
2. CFG ambiguity, e.g., “Washington” can be either a City or a State.
3. Segmentation ambiguity, e.g., [Washington] [D.C.] vs. [Washington D.C].
4. Semantic label ambiguity, e.g., “Washington D.C.” can be either an ACity or a DCity.

In one setting, the same model topology and features of the HMM/CFG composite model were directly ported into the CFG/CRF model. Because the state assignment for the non-slot filler words were unknown (e.g., as shown in Figure 1.17, the word “on” could be assigned either the PostDCity state (postamble of the departure city slot) or PreDate state (preamble of the Date slot)), HCRF was used for this model.

To model the popularity of different top level semantic frames in the air travel domain like “ShowFlight,”, “GroundTransportation,” etc., the model also included the *Frame prior* feature type f^{PR} in addition to the commonly used transition and n-gram features introduced earlier:

$$f_c^{PR}(y_{t-1}, y_{(t)}, \mathbf{x}, t) = \delta(t = 0)\delta(Frame(y_0) = c) \tag{1.57}$$

here $Frame(y)$ stands for the frame of which the state y is a part. The condition $\delta(t = 0)$ ensures that the frame prior feature is only activated once for each utterance.

Figure 1.25 shows the test set slot error rates (SER) at different training iterations. With the flat start initialization that set every parameter to 0 initially (top curve), the error rate never comes close to the 5% baseline error rate of the HMM/CFG model. With the generative model initialization that imported the model parameters of a generative model to initialize the conditional model parameters, the error rate is reduced to 4.8% at the second iteration, but the model quickly gets over-trained afterward.

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

1. 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.

departing	**from**	**boston**	**on**	**christmas**	**eve**
|	|	|	|	|	|
PreDCity	*PreDCity*	*DCity*	*PreDate*	*Date*	*Date*

Figure 1.26 The preamble-only labeling scheme: once the slots are marked in the simplified model topology, the state sequence is fully marked, leaving no hidden variables.

2. In order to account for the 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.
3. The hidden variables make parameter estimation less reliable, given only a small amount of training data.

An important lesson here is that we should never think generatively when applying the 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 from different states in a sentence. Language model smoothing may not be necessary since the task of the model is no long to assign a likelihood to an observation – it is unnecessary to waste the model's capacity to predict unseen observation events. The worst consequence of an unseen test n-gram is the n-gram feature will not be used, while the model may still be able to make the right decision based on the other available features. The distinction of preambles and postambles, which was designed to sharpen the distribution of the generative model to improve the likelihood (perplexity) of the observation, may also be unnecessary. Every word that appears between two slots can be labeled as the preamble state of the second slot, as illustrated by Figure 1.26, or all the null state, as illustrated in Figure 1.20. This labeling scheme effectively removes the hidden variables and simplifies the model. It not only expedites model training, but also prevents parameters from settling at a local optimum, because the objective function is convex now.

To fully take advantage of the conditional models' capability in discriminative training and in incorporating interdependent overlapping features, the following types of features are included in the model:

1. The first feature type, *chunk coverage for preamble words*, aims at correcting the confusion between a slot filler state and a slot preamble state – it may be 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* in the training data, and a word that is part of the string covered by the CFG rule for the filler of s is unseen 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. The chunk coverage features detect the potential occurrence of this confusion and informing the model that x_t, which is a potential slot filler, has been labeled as a preamble.

$$f^{CC}_{NT}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(NT \triangleright [\mathbf{x}, t])\delta(\text{IsPreamble}(y_t)) \tag{1.58}$$

2. The *previous-slot context* feature type tries to capture some of the long distance dependency. This is based on the observation that the phrase "*at two PM*" in "*flight from Seattle to Boston at two PM*" and in "*flight departing from Seattle arriving in Boston at two PM.*" has different interpretations, which depends on the identities of the words ("*to*" versus "*arriving at*") in the preamble of the ACity slots ("*Boston*") preceding the time expression[4]. Since the starting position of the previous slot is unknown given then Markov assumption of the linear chain CRF, its preamble words are approximated by the words in a window preceding the longest segment covered by the CFG rules modeling the filler of the previous slot:

$$
\begin{aligned}
f^{PC}_{s_1,s_2,w}\ (y_{t-1}, y_t, \mathbf{x}, t) = \\
\delta(y_{t-1} = s_1)\delta(y_t = s_2)\delta(w \in x^{\arg\min_i(NT(s_1)\to x_i^{t-1})-1}_{\arg\min_i(NT(s_1)\to x_i^{t-1})-K})
\end{aligned}
\quad (1.59)
$$

here K is a window size, $NT(s)$ stands for the CFG rule name for the filler of the slot represented by the state s, $x_i^j = x_i, \ldots, x_j$, and $\arg\min_i(NT \to x_i^j)$ represents the minimum index (the leftmost position) such that the substring from this position to j is covered by the CFG rule NT.

Compared to the trigger features, this feature's capability of modeling long distance dependency is limited to the previous slot's context only. However, since there are only a small set of words for the previous slots' preambles from the training data, there is no need for the expensive feature selection procedure.

3. The *chunk coverage for slot boundary* feature type is introduced to penalize erroneous segmentation, such as segmenting "*Washington D.C.*" into two separate "City" slots (or a "State" followed by a "City"). It 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 CFG nonterminal:

$$
\begin{aligned}
f^{SB}_{NT}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(NT \to \ldots x_{t-1}x_t \ldots)\delta(y_{t-1} \text{ is filler end}) \\
\delta(y_t \text{ is filler start})
\end{aligned}
\quad (1.60)
$$

In Wang et al. (2006), this feature type is tired with the *chunk coverage features for preamble words*, and does not introduce any new parameters.

Table 1.2 1 shows the number of new parameters and the slot error rate (SER) on the test data, after each new feature type has been cumulatively added to the model. The new features improve the prediction of slot identities and reduce the SER by 21%, relative to the generative HMM/CFG composite model. The additional features, again, are critical in reducing the slot error rate.

While CRF is often characterized as a sequential labeling model, it is important to note that such a model can perform semantic frame classification (picking the correct frame for an utterance) and slot filling simultanoutly. In the unified CFG/CRF model, this is achieved by

[4]Note that the preceding slot is ACity in both cases, hence the transition feature is not able to differentiate these two cases.

Table 1.2 Number of additional parameters and the slot error rate after each new feature type has been added to the model cumulatively.

Feature Type	Number of Parameters	SER
Frame Prior	6	
+State Transition	+1377	18.68%
+Unigrams	+14433	7.29%
+Bigrams	+58191	7.23%
+Chunk Coverage of Preamble Word	+156	6.87%
+Previous-Slot Context	+290	5.46%
+Chunk Coverage of Slot Boundaries	+0	3.94%

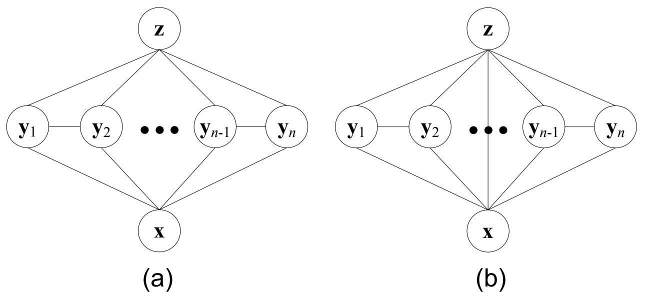


Figure 1.27 Graphical model representation of the joint model for frame classification and slot filling with CRFs. (a) The model used in the unified CFG/CRF model and (b) a general "triangular chain CRF" model.

using frame-specfic slot labels and the introduction of the frame prior feature in Eq. (1.57), which effectively results in the graphic model illustrated by Figure 1.27(a)[5], which can be viewed as a special factorization of the more general model depicted by Figure 1.27(b). Jeong and Geunbae Lee (2008) named Figure 1.27(b) the "triangular chain CRF", and introduced a couple of other factorizations of it for the frame-based SLU.

In fact, this type of conditional model for joint classification/sequential labeling is a common practice in many other speech processing applications. For example, Gunawardanaand et al. (2005) used HCRF for phone classification, where hidden states are aligned (tagged) to acoustic frames, while a class label is assigned to the entire input in the mean time.

1.3.5 Frame-based SLU in Context

Up till now we have only discussed the understanding of an utterance out of its context. In practical speech applications, however, users seldom specify all the important information in a single utterance. They are often engaged in a dialog with the system such that important pieces of information (slots) can be provided to the system at different dialog turns. The

[5]For the sake of simplicity, the long distance dependency introduced by the unified CFG/CRF model is not shown here

USER1:	Show me the flights from Seattle to San Francisco next Wednesday.
SYS1:	<Displays a list of flights, including flight AS 220>
USER2:	I also need a return flight from San Francisco to Seattle.
SYS2:	What's the date for the flight?
USER3:	The Monday after.
SYS3:	<Displays a list of flights>
USER4:	Display details of AS 220.
SYS4:	<Displays the detailed information of the outbounding flight AS 220>

Figure 1.28 The effect of constraints on inheriting semantic information from the discourse.

pieces need to be assembled incrementally to form the semantic representation, such that the dialog system can take appropriate actions according to the information gathered up to the current dialog turn. Without the capability of understanding in context, a dialog system cannot interact with users proficiently. In the specific example of the ATIS corpora, there are class D utterances that cannot be correctly interpreted without taking into consideration the context inforamtion.

For that purpose, a discourse structure D is used to record the information till the current dialog turn when the user utters U. D may contain multiple partially instantiated semantic frames. The context agnostic SLU algorithms/models that we have introduced can be applied to obtain the context independent meaning M_u of u. From D and M_u, a context dependent meaning M_D can be constructed.

Constructing M_D from D and M_u is much more complicated than simply adding together the pieces of information in D and M_u. The information in D often cannot be inherited by M_D. Using ATIS as an example, the mention of a new departure and arrival city often implies that the infomation in D cannot be carried over to M_D because it signals that the user has switched to another task of finding a different flight. Similarly, if the user mentions a specific flight number, the focus has been reset and all the slot information need to be replaced by the corresponding information of that particular flight. The exemplar dialog in Figure 1.28 shows the effect of these two constraints. There are many constraints like these two in the ATIS domain, which were often modeled by handcrafted rules (Seneff et al. 1991), with the exception of the statistical discourse model in (Miller et al. 1996), for which we discuss now.

The statistical discourse model represents M_D with a vector Y, where each element contains the value of a specific slot. Another vector X of the same dimension as Y is used to represent a combination of M_u and M_p, the latest instantiation of the same frame as M_u in D. The elements in X specify one of the five relations between the fillers of the corresponding slots in M_u and M_p (excerpted from (Miller et al. 1996)):

INITIAL	Slot filled in M_u but not in M_p
TACIT	Slot filled in M_p but not in M_u
REITERATE	Slot filled in both M_p and M_u, with the same value
CHANGE	Slot filled in both M_p and M_u, with different values
IRRELEVANT	Slot not filled in either M_p or M_u

Then

$$P(M_D \mid D, M_u) \approx P(Y \mid X) = \prod_i P_{slot}(Y_i \mid X; Y_1, \ldots, Y_{i-1}) \tag{1.61}$$

When $X_i \neq$ TACIT, $P_{slot}(Y_i \mid X; Y_1, \ldots, Y_{i-1}) = 1$ if $Y_i = M_{ui}$, the ith slot of M_u. When $X_i =$ TACIT, a binary statistical classifier is trained for each slot to determine if M_{pi} should be copied to M_D. Miller et al. (1996) employed the decision trees for classification.

Putting it in a broad picture, Eq. (1.61) can be embeded in the following decision rule for utterance understanding in context:

$$\begin{aligned}\hat{M} &= \arg\max_{M_D} P(M_D \mid U, D) = \arg\max_{M_D} \sum_{M_u} P(M_D \mid M_u, U, D) P(M_u \mid U, D) \\ &\approx \arg\max_{M_D} \max_{M_u} P(M_D \mid M_u, D) P(M_u \mid U) \\ &= \arg\max_{M_D} \max_{M_u} P(M_D \mid M_u, D) P(U \mid M_u) P(M_u) \qquad (1.62)\end{aligned}$$

where D represents the context, $P(M_D \mid M_u, D)$ is the discourse model, $P(U \mid M_u)$ is the lexicalization model, and $P(M_u)$ is the semantic prior model.

1.4 Summary

This chapter has introduced the problem of the semantic frame based spoken language understanding, a common SLU problem that has been faced in many research and commercial applications. It has covered some important solutions to the problem, ranging from knowledge-based to data-driven, statistical learning approaches. In the knowledge based solutions, manually developed grammars are coupled with robust parsing technologies. There are two major camps in grammar design, one favors the reuse of domain-independent syntactic grammars across different domains and augmenting the grammars with domain-specific semantic information; the other advocates the direct modeling of semantics with a domain dependent semantic grammar. For the data-driven approaches, both generative models and conditional models for the frame-based SLU are reviewed. The generative model framework has two major component, the *semantic prior model* and the *lexicalization model.* Several seminal generative models are reviewed in details in terms of these two component models. The conditional models have the advantages of discriminative learning and the capability of incorporating many interdependent, overlapping features that are difficult to be included in a generative framework. Hence it is powerful in designing specific features to address some difficult problems, such as long distance dependency. It has also been shown not a good practice to directly port a generative model's topology and features into a conditional model. Rather the design of a conditional model should focus on taking full advantage of its capabilities.

While it is necessary to have some real usage data for reference in the process of grammar development, the knowledge-based approach in general requires less training data to build a model (grammar), and the data do not have to be labeled. With expert's involvement, it is agile to quickly adapt to the topic/trend shifts that may happen frequently in practical spoken dialog applications. However, it needs a combination of application and linguistic expertise, and the robustness does not come from the model directly. Instead, it has to depend on a robust parsing mechanism. The statistical models, on the other hand, are robust to noisy, error-prone inputs by design. It does not need linguistic expertise for grammar development. However, its performance is greatly affected by the availability of the labeled training data. In this chapter, we have also shown the advantage of combining statistical models with the knowledge-based solutions by integrating CFG rules in both generative and conditional models. This has greatly reduce the dependency on a large amount of labeled training data.

References

Allen JF, Miller BW, Ringger EK and Sikorshi T 1996a Robust understanding in a dialogue system. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, CA.

Allen JF, Miller BW, Ringger EK and Sikorski T 1996b A robust system for natural spoken dialogue. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 62–70.

Alshawi H 1992 *The Core Language Engine.*. MIT Press, Cambridge, MA., USA.

Alshawi H and van Eijck J 1989 Logical forms in the core language engine. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pp. 25–32. ACL.

Bahl L, Brown P, de Souza P and Mercer R 1986 Maximum mutual information estimation of Hidden Markov Model parameters for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 49–52. IEEE, Tokyo, Japan.

Bilmes J and Zweig G 2002 The graphical models toolkit: An open source software system for speech and time-series processing. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3916–3919.

Bonneau-Maynard H, Rosset S, Ayache C, Kuhn A and Mostefa D 2005 Semantic annotation of the French MEDIA dialog corpus. *Proceedings of INTERSPEECH*, pp. 3457–3460.

Brown PF, Della Pietra VJ, deSouza PV, Lai JC and Mercer RL 1992 Class-based n-gram models of natural language.. *Computational Linguistics* **18**(4), 467–479.

Chelba C, Mahajan M and Acero A 2003 Speech utterance classification. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE.

Dahl DA, Bates M, Brown M, Fisher W, Hunicke-Smith K, Pallett D, Pao C, Rudnicky A and Shriberg E 1994 Expanding the scope of the ATIS task: the ATIS-3 corpus. *Proceedings of the Human Language Technology Workshop*. Morgan Kaufmann.

Della Pietra S, Epstein M, Roukos S and Ward T 1997 Fertility models for statistical natural language understanding. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 168–173, Madrid, Spain.

Dempster AP, Laird N and Rubin DB 1977 Maximum likelihood from incomplete data via the EM algorithm.. *Journal of the Royal Statistical Society B* **39**, 1–38.

Dowding J, Gawron JM, Appelt D, Bear J, Cherny L, Moore R and Moran D 1993 Gemini: A natural language system for spoken-language understanding. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 54–61, Columbus, Ohio.

ELRA n.d. Elra – european language resources association.

Estève Y, Raymond C, Béchét F and de Mori R 2003 Conceptual decoding for spoken dialog systems. *Proceedings of the Eurospeech Conference*, Geneva, Switzerland.

Gunawardanaand A, Mahajanand M, Acero A and Platt JC 2005 Hidden conditional random fields for phone classification *Proceedings of INTERSPEECH*, pp. 1117–1120. ISCA.

He Y and Young S 2003 Hidden vector state model for hierarchical semantic parsing. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, China.

Hemphill CT, Godfrey JJ and Doddington GR 1990 The ATIS spoken language systems pilot corpus. *Proceedings of the workshop on Speech and Natural Language, HLT'90*, pp. 96–101. Association for Computational Linguistics, Morristown, NJ, USA.

Hunt A and McGlashan S n.d. Speech recognition grammar specification version 1.0.

Jelinek F and Mercer EL 1980 Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice* (ed. Gelsema D and Kanal L) North-Holland.

Jeong M and Geunbae Lee G 2008 Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing* **16**(7), 1287–1302.

Jeong M and Lee GG 2008 Practical use of non-local features for statistical spoken language understanding.. *Computer Speech & Language* **22**(2), 148–170.

Juang BH, Chou W and Lee CH 1997 Minimum classification error rate methods for speech recognition.. *IEEE Transactions on Speech and Audio Processing* **5**(3), 257–265.

Kneser R and Ney H 1993 Improved clustering techniques for class-based statistical language modelling. *Proceedings of the Eurospeech Conference*, pp. 973–976, Berlin, Germany.

Lafferty J, McCallum A and Pereira F 2001 Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML*, pp. 282–289.

LDC n.d. Ldc - linguistic data consortium.

Lefèvre F 2007 Dynamic Bayesian networks and discriminative classifiers for multi-stage semantic interpretation. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 13–16. IEEE.

Macherey K, Och FJ and Ney H 2001 Natural language understanding using statistical machine translation. *Proceedings of the Eurospeech Conference*.

McCallum A 2003 Efficiently inducing features of conditional random fields. *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI03)*, Acapulco, Mexico.

Meurs MJ, Lefèvre F and de Mori R 2009 Learning Bayesian networks for semantic frame composition in a spoken dialog system. *Proceedings of HLT-NAACL*, Boulder, Colorado, USA.

Miller S, Bobrow R, Ingria R and Schwartz R 1994 Hidden understanding models of natural language. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, New Mexico State University.

Miller S, Stallard D, Bobrow R and Schwartz R 1996 A fully statistical approach to natural language interfaces *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pp. 55–61. Association for Computational Linguistics, Morristown, NJ, USA.

Moore R 1998 Using natural language knowledge sources in speech recognition. *NATO Advanced Studies Institute*.

Ostendorf M, Digalakis V and Kimball O 1996 From HMMs to segment models: A unified view of stochastic modeling for speech recognition.. *IEEE Transactions on Speech and Audio Processing* **4**(5), 360–378.

Pieraccini R and Levin E 1993 A learning approach to natural language understanding. *1993 NATO ASI Summer School* New Advances and Trends in Speech Recognition and Coding. Springer-Verlag, Bubion, Spain.

Rabiner L and Juang BH 1993 *Fundamentals of Speech Recognition.*. Prentice Hall P T R, Englewood Cliffs, New Jersey.

Raymond C and Riccardi G 2007 Generative and discriminative algorithms for spoken language understanding. *Proceedings of INTERSPEECH*, pp. 1605–1608.

Rayner M, Dowding J and Hockey BA 2001 A baseline method for compiling typed unification grammars into context free language models. *Proceedings of the Eurospeech Conference*, Aalborg, Denmark.

Riccardi G and Gorin AL 1998 Stochastic language models for speech recognition and understanding. *Proceedings of the International Conference on Spoken Language Processing*, Sidney, Australia.

Riccardi G, Pieraccini R and Bocchieri E 1996 Stochastic automata for language modeling.. *Computer Speech and Language* **10**, 265–293.

Rosenfeld R 1996 A maximum entropy approach to adaptive statistical language modelling.. *Computer Speech & Language* **10**, 187–228.

Seneff S 1992 Tina: A natural language system for spoken language applications.. *Computational Linguistics* **18**(1), 61–86.

Seneff S, Hirschman L and Zue VW 1991 Interactive problem solving and dialogue in the ATIS domain *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pp. 354–359.

Stolcke A 2002 SRILM - an extensible language modeling toolkit. *Proceedings of the IEEE International Conference on Spoken Language Processing*, Denver, Colorado.

W3C n.d. Semantic interpretation for speech recognition (SISR) version 1.0.

Walker M, Aberdeen J, Boland J, Bratt E, Garofolo J, Hirschman L, Le A, Lee S, Narayanan S, Papineni K, Pellom B, Polifroni J, Potamianos A, Prabhu P, Rudnicky A and Sanders G 2001 DARPA Communicator dialog travel planning systems: The June 2000 data collection. *Proceedings of the Eurospeech Conference*.

Walker MA, Rudnicky A, Prasad R, Aberdeen J, Bratt EO, Garofolo J, Hastie H, Le A, Pellom B, Potamianos A, Passonneau R, Roukos S, S G, Seneff S and Stallard D 2002 DARPA Communicator: cross-system results for the 2001 evaluation. *Proceedings of the International Conference on Spoken Language Processing*, pp. 269–272.

Wang YY and Acero A 2003a Combination of CFG and n-gram modeling in semantic grammar learning. *Proceedings of the Eurospeech Conference*, pp. 2809–2812. ISCA.

Wang YY and Acero A 2003b Is word error rate a good indicator for spoken language understanding accuracy. *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 577–582.

Wang YY and Acero A 2006 Rapid development of spoken language understanding grammars.. *Speech Communication* **48**(3-4), 390–416.

Wang YY, Lee J, Mahajan M and Acero A 2006 Combining statistical and knowledge-based spoken language understanding in conditional models. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 882–889. Association for Computational Linguistics.

Ward W 1991 Understanding spontaneous speech: the Phoenix system. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 365–367, Toronto, Canada.

Woods WA 1983 Language processing for speech understanding. *Computer Speech Processing* Prentice-Hall International Englewood Cliffs, NJ.

Yu D, Wang K, Mahajan M, Mau P and Acero A 2003 Improved name recognition with user modeling. *Proceedings of the Eurospeech Conference*, Geneva, Switzerland.

2

Voice Search

Ye-Yi Wang, Dong Yu, Yun-Cheng Ju and Alex Acero

Microsoft Research

Voice search is one of the most actively investigated speech understanding technologies currently. It is the technology underlying many commercial spoken dialog systems (SDSs) that provide users with the information they request with a spoken query. The information normally exists in a large database, and the query has to be compared with a field in the database to obtain the relevant information. The contents of the field, such as business or product names, are often unstructured text. For example, directory assistance (DA) (Bacchiani et al. 2008; Yu et al. 2007) is one of the most popular voice search applications, in which users issue a spoken query and an automated system returns the phone number and address information of a business or an individual. The applications include both telephone only services and multi-modal services on mobile devices. Other voice search applications include music/video management (Mann et al. 2007; Song et al. 2009), business and product reviews (Zweig et al. 2007), stock price quote, and conference information systems (Andreani et al. 2006; Bohus et al. 2007). Recently the task has been extended to using voice queries for Web search from mobile devices, as manifested by the commercial systems from Google and Yahoo.

2.1 Background

Figure 2.1 shows the typical architecture of a voice search system, where a user's utterance is first recognized with an automatic speech recognizer (ASR) that utilizes an acoustic model (AM), a pronunciation model (PM) and a language model (LM). The m-best results from the ASR are passed to a search component to obtain the n-best semantic interpretations, i.e., a list of up to n entries in the database. The interpretations are passed to a dialog manager (DM) subsequently. The DM utilizes confidence measures, which indicate the certainty of the interpretations, to decide how to present the n-best results. If the system has high confidence on a few entries, it directly presents them to the user. Otherwise, a disambiguation module is exploited to interact with the user to understand what he actually needs. their true intent.

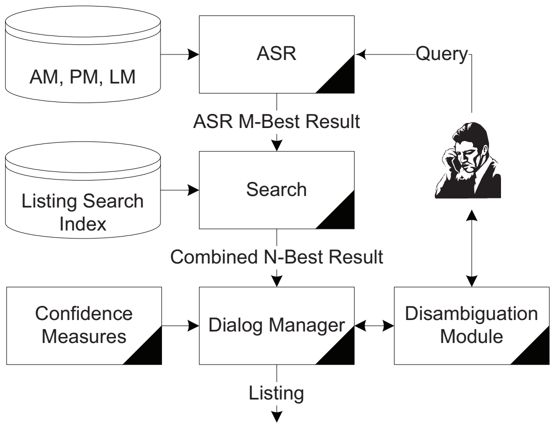


Figure 2.1 Architecture of a typical voice search system.

2.1.1 *Voice Search Compared to the Other Spoken Dialog Technologies*

For automated human-machine interaction, we have reviewed the problem of the frame-based SLU and call-routing. Compared to them, the problem of voice search has its unique characteristics and needs to be addressed separately. The frame-based SLU are used to gather the attribute values of an entity that users are interested in, like the originating and destination cities of a flight. In such systems, users often have to use canned expressions within a small domain. In a directed dialog system, a user's utterances are limited to answers to what the system has prompted for, which often contain a single piece of semantic information; while in a mixed-initiative system, users are allowed to volunteer more semantic information in a single utterance – we call this type of semantic understanding *high-resolution* in the sense that multiple slots need to be identified. The call-routing applications remove the constraints on what a user can say, so one can speak naturally. This is accomplished at the expense of limiting the target semantic space: the understanding of natural language inputs is often achieved with statistical classifiers, which map a user's input to a destination semantic class (intent) in a predefined set of limited size. The classifiers can hardly perform high resolution understanding with many slots, or scale up with a huge number (e.g., thousands to millions) of destination classes. Voice search applications differ from the frame-based applications in their lack of detailed, high resolution semantic analysis. They are similar to the call-routing applications with respect to the naturalness of user inputs and the huge input space. However, they differ from the call-routing applications in the sense that their semantic space, or in the terminology of the call-routing applications, the inventory of the "destination classes," is enormous – sometimes in the range of millions of entries. Data are seldom sufficient to train a statistical classifier. Instead, information retrieval techniques based on some similarity measures between a query and a candidate entry are used to select the entry that best matches the query. Hence in general there is no need of labeled training data. Table 2.1 compares the three types of technologies.

Table 2.1 Comparison of the frame-based SLU, call-routing and voice search spoken language understanding with respect to the characteristics of the input utterance space and the output semantic space.

SLU Task	**User input utterances**		**Target semantic representation**	
	Naturalness	Input space	Resolution	Semantic space
Frame-based/directed dialog	low	small	low	small
Frame-based/mixed-initiative	Low-medium	small	high	small
Call routing	high	large	low	small
Voice search	Medium-high	large	low	medium-large

2.1.2 History of Voice Search

Early work on voice search focused on directory assistance (DA). Institutions on both sides of the Atlantic have deployed experimental systems during mid-late 90's. The early studies focused mainly on residential DA (Billi et al. 1998; Kamm et al. 1995; Lennig et al. 1994), and speech recognition was the major topic of research – as long as personal names were correctly recognized, the search could be a simple database lookup. As a result, the dialog strategies centered on limiting the scope (hence perplexity) of the target listing space for ASR and the confidence measures mostly relied on features from the ASR. Related work includes enterprise level auto-attendant (a.k.a. name dialing) services from Phonetic Systems (acquired by ScanSoft, then merged with Nuance), AT&T (Buntschuh et al. 1998), IBM (Gao et al. 2001), and Microsoft (Ollason et al. 2004). While automating residential DA is important in reducing the operational cost, it is only a small portion (19%) of the total received calls compared to the 61% of business DA calls (Billi et al. 1998). Therefore there have been increasing interests in business DA recently, with the commercial deployments from Tellme (acquired by Microsoft), Jingle Networks, AT&T, Google (Bacchiani et al. 2008), Verizon and Cingular (merged with AT&T Wireless now) and Microsoft (Yu et al. 2007). Because the level of linguistic variance is much higher in business DA queries, SLU/search aiming at correctly interpreting user's intent becomes an important research topic. The linguistic variance increases the ambiguity and uncertainty in the interpretation of a user's intent. As a result, dialog research focuses on the disambiguation strategy, as well as the confidence measures that look into features from different system components to accurately predict the end-to-end performance in interpreting a spoken query.

Other voice search applications include the stock quote system from Tellme and a product/business rating system from Microsoft (Zweig et al. 2007). Separate efforts have been made on conference information systems by Carnegie Mellon University (Bohus et al. 2007) and by the collaboration among AT&T, ICSI, Edinburgh University and Speech Village (Andreani et al. 2006), where users can request information about thousands of papers published in a conference. In entertainment, Daimler has investigated digital music management in automobiles (Mann et al. 2007), and Ford and Microsoft have introduced the commercial dashboard device Ford SYNC that allows in-car music search. Like the business DA applications, many new voice search applications call for research activities in search/SLU and dialog management in addition to speech recognition.

With the broad adoption of mobile devices and the availability of wireless access to the internet, many companies are actively engaged in the space of voice search on mobile or in-car devices (Mann et al. 2007). Google has introduced speech recognition for Google Apps on iPhone, which allows users to use speech input for business or general Web search. Yahoo has similar offering with Yahoo! Search App. Microsoft has just released a voice-enabled Bing Mobile Search. AT&T is currently working on a voice enabled local search for iPhone (Feng et al. 2009). In additional to the industrial efforts, academia is also studying the problem of voice-enabled general Web search (Vertanen and Kristensson 2009) Ford SYNC can connect to a user's mobile device (mobile phones, mp3 players, etc.) and use voice to control the device, including a media search with voice input. New research challenges include multi-modal (GUI with touch screen and speech) user interfaces (Acero et al. 2008; Mann et al. 2007) and efficient and scalable client-server architectures.

2.1.3 Technical Challenges

Voice search poses new challenges to the spoken dialog technology in the following areas:

1. **Speech Recognition:** The state-of-the-art ASR systems have high error rates on voice search tasks. The vocabulary size of a voice search system can be much larger than a typical frame-based or a call routing application – sometimes reaching millions of lexical entries. Many lexical entries in international individual or business names are out of vocabulary and lack the reliable pronunciation information. Calls are often made from different noisy environments. In addition, the constraints from language models are often weaker than other ASR tasks – the perplexity of a language model is often high (e.g., 400˜500 bits for business DA) for voice search.
2. **Spoken Language Understanding (SLU)/Search:** One big problem in SLU is the enormous semantic space – a DA system can easily contain hundreds of thousands (if not millions) of listings in a city. There is also a high level of linguistic variance in the input space. For example, users may not use the official name of a business in a DA or a business rating system. They would typically say, for instance, "Sears" instead of the listed official name, "Sears Roebuck & Co." In addition, the SLU/search component must be robust to ASR errors.
3. **Dialog Management:** The difficulties in ASR and SLU cause much confusability and uncertainty. Dialog manager has to effectively narrow down the scope of what a user may say to reduce the confusability and uncertainty. Search results often contain multiple entries. Disambiguation strategy is crucial in obtaining sufficient information for the correct understanding of users' intents with as few dialog turns as possible. Confidence measures are important for the dialog manager to take the appropriate action with each of the hypothesized interpretations, such that the dialog can recover gracefully from ASR and SLU errors.
4. **Feedback loop:** No systems can be perfectly built at the initial deployment. Dialog system tuning is often performed painstakingly by spoken dialog experts, starting from error analysis from the logged interaction data to find the flaws in dialog and prompt design, language/understanding model development, system implementation, etc. An interesting research topic is the automatic or semi-automatic discovery and remedy of design/implementation flaws.

Like other SLU tasks, the grand challenge in voice search application is robustness. The CSELT's study on Telecom Italia's DA system (Billi et al. 1998) showed that even though the automation rate was 92% in a laboratory study, the actual field trial automation rate was only 30% due to unexpected behavior of novice users and environment noise.

2.1.4 Data Sets

Unlike the research activities in the frame-based SLU, which are sponsored by public fundings and participated by multiple institutions from both academia and industry, the activities in voice search are largely conducted by software or telecommunication companies, each works with their own proprietary data collected from the services they provide. Out of the concerns about their users' privacy, it is almost impossible for them to share the data with the research community – there are several failed attempts from the researchers in these organizations. Due to this unfortunate situation, there is currently no common data set for voice search research. Researchers from academia have to collect their own data to conduct relevant research (Vertanen and Kristensson 2009).

2.1.5 Evaluation Metrics

Many evaluation metrics can be found in voice search related publications, some of them are used to assess the performance of a component like speech recognition or language models. Here we focus only on the metrics for an end-to-end evaluation of a voice search system. Many of them, not surprisingly, are commonly used by the information retrieval community. Note that there can be more than one listings that are the "correct" answers to a query (imagine a user search for "Starbucks in Seattle", where you may find two or three Starbucks coffee shops within a block, and also a towing company with the same name). Hence the metrics are designed to reflect a system's capability to find the correct answers and to reject the incorrect ones. A listing returned from a search engine can be either a true positive (TP, correct answer) or a false positive (FP, incorrect answer); a listing in the database that is not returned by the search engine can be a true negative (TN, correct rejection) or a false negative (FN, incorrect rejection).

1. **Precision, Recall and F$_1$-score:** *Precision* is the percentage of correct answers among all the answers from the voice search system:

$$PR = \frac{N(TP)}{N(TP) + N(FP)} \tag{2.1}$$

Precision reflects a system's capability in rejecting incorrect answers. It does not measure its capability in finding as many correct answers as possible. So recall is introduced for that purpose:

$$RE = \frac{N(TP)}{N(TP) + N(FN)} \tag{2.2}$$

F_1*-score* is the harmonic mean of precision and recall:

$$F_1 = \frac{2 * PR * RE}{PR + RE} \tag{2.3}$$

A system may have multiple operating points (often set by different thresholds to accept a hypothesis), at which different precision and recall scores can be observed. A *recall-precision curve* plots the different precision scores in relation to the corresponding recall scores at the same operating point. Similarly, a *receiver operating characteristic (ROC) curve* plots the true positive rates (recall) in relation to the corresponding false positive rates ($N(FP)/\left[N(FP)+N(TN)\right]$).

2. **Mean Reciprocal Rank (MRR):** The precision and recall does not reflect the ranking power of a voice search system – if two results are returned, one is correct and the other is not, different placement of the results will not alter the value of precision/recall/F_1-score. Ideally, placing a relevant answer at a higher rank should be favored by the evaluation metric. The *reciprocal rank* of the search results for a query is the multiplicative inverse of the rank of the first correct answer. The *mean reciprocal rank* is the reciprocal ranks averaged over a set of test queries:

   $$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i} \tag{2.4}$$

   where N is the number of test queries, and r_i is the rank of the first correct answer for the i-th query.

3. **Mean Average Precision (MAP):** The mean reciprocal rank metric does not take precision or recall into consideration. The *mean average precision* is proposed to fix this problem. The *average precision* (AP) is the average of precisions computed at different cut-off points in the result list:

   $$AP = \frac{\sum_{r=1}^{N(TP)+N(FP)} PR(r) * COR(r)}{N(TP)+N(FN)} \tag{2.5}$$

   where r is the rank of a result (smaller number indicates higher rank in a result list), $COR(r)$ is a binary function that has value 1 only when the result at rank r is a correct answer to the query. $PR(r)$ is the precision of the top r search results.

   Mean average precision (MAP) is the mean of the individual precision scores over a test set of queries.

   $$MAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.6}$$

   where AP_i is the average precision of the results for the i-th query.

4. **Normalized Discounted Cumulative Gain (NDCG):** The binary judgment of correctness of a search result may not be enough to distinguish the quality of different results. A relevance rating with more grades is more suitable in some cases. For example, the product search results for the query “black iPod player” may include a perfect result of an iPod player that is black, and another result without color information. It does not make sense to label the second result as incorrect, since it is still more relevant than a result of an MP3 player of a different brand. A 5-grade relevance ranking (“poor”, “fair”, “good”, “excellent” and “perfect”) is ofter used, corresponding to the numeric value 1-5.

The relevance of a result is discounted by a factor according to its rank in the search results. The discounting factor penalizes presenting a relevant results at a lower rank. The *discounted cumulative gain (DCG) at k* is the sum of the discounted relevance at top k different ranks. A logarithm discounting factor is often used:

$$DCG(k) = \sum_{r=1}^{k} \frac{2^{rel_r} - 1}{log_2(r+1)} \tag{2.7}$$

where rel_r is the relevance rating for the r-th search result.

The DCGs of the search results are not directly comparable across different queries, since they may result in different number of results. For each query, an oracle maximum DCG $mDCG(k)$ can be calculated by arranging the k most relevant results according to their relevancy, such that the more relevant results are always presented before the less relevant one. The *normalized discounted cumulative gain (NDCG) at rank k* over a test set of N queries is

$$NDCG(k) = \frac{1}{N} \sum_{i=1}^{N} \frac{DCG_i(k)}{mDCG_i(k)} \tag{2.8}$$

5. **M-best Search Accuracy:** *M-best Search Accuracy* ACC_M is the percentage of the correct answers among the top M answers:

$$ACC_M = \frac{1}{N * M} \sum_{i=1}^{N} \sum_{r=1}^{M} COR_i(r) \tag{2.9}$$

where $COR_i(r)$ is a binary function that has value 1 only when the result at rank r is a correct answer to the i-th query in the test set.

When a voice search system has limited capacity to present multiple search results, as in a telephony system, one-best and two-best search accuracy become more important and are popularly used.

2.2 Technology Review

In this section we review the technology that addresses the challenges in voice search applications, with a focus on the sub-problems of SLU/search and language modeling. Not surprisingly, much of the technology is developed with the DA systems because they are the most popular voice search applications so far. However, the technology is often applicable to other applications as well. For example, the product/business rating systems (Zweig et al. 2007) directly used the technology developed in a DA application (Yu et al. 2007).

2.2.1 Speech Recognition

A detailed error analysis for proper name recognition was reported in an auto-attendant system (Gao et al. 2001). Figure 2.2 shows the distribution of different causes of errors – besides 35% of common recognition errors, 31% were noise related and 22% were

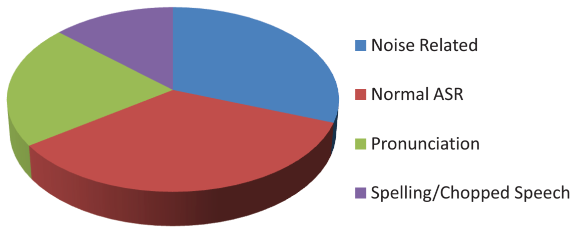


Figure 2.2 Distribution of different causes of ASR errors, from the error analysis of an auto-attendant system in (Gao et al. 2001).

pronunciation related. Many of the calls were made in a noisy environment over different noise channels. Therefore, noise-robustness is crucial to improve the ASR accuracy. On the other hand, there were many foreign names that are difficult to pronounce in an auto-attendant/DA system. In fact, pronunciation is a pervasive problem that poses challenge in many other voice search applications as well. For example, users may specify a multilingual query“*Petit Bonheur* by *Salvatore Adamo*” in music search. Hence pronunciation modeling is another important topic in ASR for voice search. In addition, better acoustic and language models are always important to reduce the ASR error rate.

Acoustic Modeling

IBM’s auto-attendant system applied speaker clustering in its acoustic model (Gao et al. 2001). Simple HMMs that have one Gaussian per context independent phone state were trained first for each speaker. Then the vectors of the means of these models were clustered with the k-means algorithm. For each test utterance, the cluster model that yielded the highest likelihood was selected. In doing so, different channel and noise conditions can be more precisely modeled by different cluster models, so noise-related problems are alleviated. In addition to speaker clustering, speaker adaptation is effective to bring the performance of a speaker-independent system closer to that of a speaker-dependent system. Unlike normal speaker adaptation, the adaptation in (Gao et al. 2001) was *massive* in the sense that the adaptation data was obtained from a pool of recent callers rather than a single speaker. The massive adaptation is helpful due to the fact that a caller often calls the same set of individuals, and that a caller may try a name repeatedly when a recognition error occurs. While massive adaptation is helpful to bring down the error rate for frequent callers, unsupervised utterance adaptation aims at improving the accuracy from an unknown speaker. In this adaptation scheme, the test utterance itself was used for adaptation with a two pass decoding. In the first pass, a speaker independent system or the system after massive adaptation was used to obtain the automatic transcript. Then a forward-backward algorithm was applied to obtain the adaptation statistics. After adapting the acoustic models using the collected statistics, the caller’s utterance was decoded in a second pass with the adapted model – this second pass may adversely increase the latency of a voice search system. Overall, with all these acoustic model enhancements and an unsupervised derivation of pronunciations (to be described below), a 28% error reduction was observed.

Pronunciation Modeling

One approach to improved pronunciation model is via augmenting the dictionary with pronunciation variants. Data-driven algorithms are commonly applied, which typically include four steps: generating phonetic transcriptions with a recognizer; aligning the auto transcriptions with manually created canonical pronunciations; deriving rules mapping from the canonical pronunciations to the variants; and pruning the rules. One limitation of this approach is that the canonical reference pronunciations must be available.

The IBM auto-attendant system (Gao et al. 2001) adopted an acoustics-only based pronunciation generation approach (Ramabhadran et al. 1998). The advantage of this approach is that no canonical pronunciation is required. This makes it more practical in voice search applications since many words do not exist in a pronunciation dictionary. With this approach, a trellis of sub-phone units was constructed from an utterance. The transition probabilities in the trellis were derived by weighting the transition probabilities of all the context-dependent realizations of the sub-phone units in a HMM acoustic model. A Viterbi search was performed to obtain the best sub-phone sequences from the trellis and a pronunciation was subsequently derived from the sequence. Experiments in (Gao et al. 2001) showed a 17% relative error reduction when the test set and training set had overlapping unseen words.

Trade-offs often have to be made in adding pronunciation variants to a dictionary. The additional pronunciations, on the one hand, make the word models match the actual acoustic signal more precisely; on the other hand, give rise to a large number of highly confusable word models. Instead of augmenting an existing pronunciation dictionary with variants, a pronunciation distortion model was introduced in (Béchet et al. 2002) to rescore the n-best hypotheses generated from a first recognition pass. The distortion model incorporates the "knowledge source" about the common distortions observed in a specific spoken language. For example, only insertions were considered in the distortion model for French in (Béchet et al. 2002) because it is frequently observed that silence segments are often inserted between certain pairs of consonants like [*m*][*n*], and a schwa is often inserted after a consonant at the end of an utterance. Formally, let A and W denote the acoustic signal and the text of a caller's utterance, and τ_w a phone sequence that may be distorted from the canonical pronunciation of W, then a hypothesis $\hat{W}$ can be selected from the first pass n-best recognitions according to the following decision rule:

$$\begin{aligned}\hat{W} &= \arg\max_{W} P\left(W \mid A\right) = \arg\max_{W} \sum_{\tau_w} P\left(W, \tau_w \mid A\right) \\ &\approx \arg\max_{W,\tau_w} P\left(W, \tau_w \mid A\right) \approx \arg\max_{W,\tau_w} P\left(\tau_w\right) P\left(A \mid \tau_w\right) P\left(W \mid \tau_w\right) \qquad (2.10)\end{aligned}$$

The last approximation in the equation includes an application of the Bayes rule and an assumption of independence between A and W given τ_w. The prior of a distorted phone sequence, τ_w, can be written in terms of η_w, which is the the canonical pronunciation of W; and δ_w, which is the difference between τ_w and η_w:

$$P\left(\tau_w\right) = P\left(\eta_w \mid \delta_w\right) = P\left(\delta_w \mid \eta_w\right) P\left(\eta_w\right) \qquad (2.11)$$

In theory, $P\left(\delta_w\ ,\ \eta_w\right)$ and $P\left(\eta_w\right)$ can be estimated from data. In (Béchet et al. 2002), a uniform distribution over all plausible insertions was used instead for $P\left(\delta_w \mid \eta_w\right)$ due to the lack of data. $P\left(A \mid \tau_w\right)$ in the decision rule of Eq. (2.10) can be obtained from the acoustic model with all possible alignments between A and τ_w. Since only insertion is considered in (Béchet et al. 2002), $P\left(W \mid \tau_w\right)$ was obtained by multiplying the probabilities of all successful insertions. Experiment results showed that the rescoring had improved the one-best ASR accuracy from 50% to 59.8%.

Language Modeling

Early DA systems compiled directory entries into a finite-state grammar as the language model for ASR. This rule-based language model does not scale up well with directory size due to the increased perplexity. It was found that the ASR accuracy decreases linearly with logarithmic increases in directory size (Kamm et al. 1995). On the other hand, it was noticed that the distribution of the requested listings followed the Zipf's law. 10% (20%) of call volumes were covered by only 245 (870) listings. So in (Kamm et al. 1995; Natarajan et al. 2002), a semi-automated DA system was built that only covered the frequently requested listings and relayed the remaining requests to human operators.

One problem of the rule-based LMs constructed from the database listings is their poor coverage. Callers seldom say a business name exactly as it appears in the database – just consider the earlier example of "Sears Roebuck & Co" versus "Sears." It was mentioned in (Béchet et al. 2000) that variant expressions for business names could be semi-automatically derived from data. Although it did not report how this was achieved, a straightforward method would compare a caller's utterance (e.g., "Kung-Ho Chinese Restaurant") with the actual listing released to the caller (e.g. "Kung-Ho Cuisine of China") by operators and learn that "Chinese Restaurant" is a synonym of "Cuisine of China." This "synonym" rule-based approach is usually expensive; the rule coverage is highly restricted by the data available; and the rules may be over-generalized without careful crafting.

The problem was tackled without using the data from the callers in (Jan et al. 2003). A method was proposed to automatically construct a finite state signature LM from a business directory database alone, which would accept different query variants. Here a *signature* is a subsequence of the words in a listing that uniquely identifies the listing. For example, with the listings "3-L Hair World on North 3rd Street" and "Suzie's Hair World on Main Street," "3-L", "Hair 3rd", and "Hair Main" are signatures because they occur in only one listing. On the contrary, the subsequences "Hair World" and "World on" are not signatures because they appear in both listings. Based on the signatures, a finite state transducer can be constructed as follows (the example is taken from (Jan et al. 2003)):

$S :=$ 3-L Hair World? On? North? 3rd ? Street? : 1 |
3-L Hair? World? On? North 3rd ? Street? : 1 |
3-L Hair? World on? North? 3rd ? Street? : 1 |
3-L? Hair World? On? North 3rd ? Street? : 1 |
Suzie's? Hair World? On? Main Street? : 2 |
Suzie's Hair World? On? Main? Street? :2 |
Suzie's Hair? World on? Main? Street? :2 |
Suzie's? Hair? World on? Main Street? :2

where each entry in the grammar corresponds to a signature. The terms in a signature are obligatory, whereas the terms in a listing but not in the signature are optional (marked by '?'.) The numbers after ':' is the semantic output from the transducer that represents the ID of a listing in the database. In doing so, every utterance matched by a rule can be uniquely associated with a listing. Because the non-essential words are optional, this makes the grammar more robust to utterances that omit these words. When the directory gets larger, an entry may bear no signature because each of its subsequences can be a subsequence of another entry. This problem was handled with the "confusion sets" in (Jan et al. 2003).

The rationale behind the signature grammar is that any term in an entry is droppable as long as the drop does not cause the confusion with another entry. While this is very practical in reducing the search ambiguity, it may be risky in modeling human language – speakers are very likely to drop terms that would lead to ambiguity. For example, they often say "Calabria" instead of "Calabria Restaurant" even though the former may cause confusions with "Calabria Electric" and "Calabria Jack J Do."

Another approach to improved the robustness is via statistical n-gram models (Bacchiani et al. 2008; Natarajan et al. 2002; Yu et al. 2007). An n-gram model is more robust because it does not require a user's utterance to match a rule exactly; because it provides a statistical framework for fair comparison between different hypotheses; and because it has well-studied smoothing algorithms to estimate the likelihood of unseen events more accurately. Ideally, a statistical n-gram model should be built from the transcripts of real calls, which demonstrate not only the different ways callers refer to businesses but also the probability of each such ways. Unfortunately, it is not realistic to collect enough calls to provide a good coverage for a large listing set, especially during the early stage of development. An interpolated LM was proposed to estimate the n-gram probability in (Yu et al. 2007):

$$P\left(w\right) = \lambda P_t\left(w\right) + \left(1-\lambda\right) P_l\left(w\right), \tag{2.12}$$

where $P_t\left(w\right)$ is the LM built using the transcripts of real calls, $P_l(w)$ is the LM built using a listing database, and λ is the interpolation weight, which was tuned with a cross-validation set collected under real usage scenario. Here $P_t\left(w\right)$ can be constructed from the transcribed data straightforwardly. Building $P_l\left(w\right)$, on the other hand, takes more considerations because the database entries may not reflect the actual ways that callers refer to them. A statistical variation model was introduced to account for the common differences between the database listings and the actual callers' queries. The model was based on the rationale similar to that of the signature model, namely callers are more likely to say the words that distinguish one listing from others. However, instead of making risky binary decisions, it modeled the importance of a word statistically according to its discriminative capability and its positions in a listing (based on the observations that callers are more likely to say the initial words in a listing). Here, the discriminative capability of a word was determined by its inverse document frequency (see Section 2.2.2), and a position importance weight $w_l^i, (0 < w_l^i \leq 1)$ was associated to each word position. A word was droppable with a probability inversely proportional to its importance. In addition, the model took into account the business category information for smoothing – each word had a probability to be followed by the category words (e.g., "restaurant.") This probability correlated to the importance and the category-indication capability (a mutual-information based measure) of a word. Furthermore, an efficient interpolation with a large vocabulary background LM (Yu et al. 2006) had provided additional robustness. The internal investigation in Microsoft has revealed that the statistical

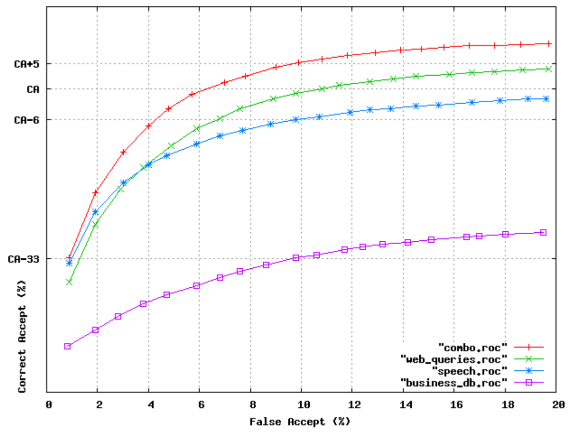


Figure 2.3 ROC curves of the voice search performance for business listings with language models trained with different training materials. The bottom (business_db.roc) is the curve for the language model trained with the listing data alone; the top (combo.roc) is the curve for the language model trained with the listing data and the usage data from transcribed voice queries and logged Web queries. The curves in the middle show the contributions from the transcribed voice search usage data (speech.roc) and the Web search queries (web_queries.roc). The big gaps between the bottom curve and the rest show the style mismatch between the listing data and the actual queries issued by the users (Courtesy of Michiel Bacchiani).

language model, together with the vector space model for listing search (see Section 2.2.2), has greatly outperformed the signature-based approach – at the same precision level, the recall has been almost doubled.

Similarly, Bacchiani et al. (2008) uses a large amount of text from different sources to train an n-gram language model. The data sources include business listings, transcribed voice search queries and the text queries logged by a local search engine. Systematic studies show that the latter two types of queries significantly improve the search performance, as illustrated by Figure 2.3.

In (Li et al. 2008), a machine-translation based approach was used to enrich the language model training data in a directory assistance application. It aimed at modeling the linguistic variations – users may refer to a listing in a different way as the original form it appears in the database; and there are often multiple paraphrases for the same listing. For example, the listing *Kung Ho Cuisine of China* can be expressed as *Kung Ho*, *Kung Ho Chinese Restaurant*, or *Kung Ho Restaurant*. In this case, the LM trained using listing in the original form in the database may not best predict what users will actually say.

Li et al. (2008) used transcribed queries to search for the listing database. If the listing of the top search result was close enough to the corresponding query according to some distance measure ((Li et al. 2008) used the Tf-Idf weighted vector space model, which we will discussed in Section 2.2.2), the listing/query pairs are used as the training examples for a machine translation system. The trained translation model is applied to the listings in a database to obtain a new data set that contains the listings in the original form and up to 3-best translations (paraphrases). The new data set, together with the transcribed queries,

Table 2.2 Perplexity comparison between the language models trained with the query augmented listings (QAL) and the query and translation augmented listings (QTAL).

# of queries in training	1k	3k	7k	10k	14k
QAL	1534	1414	1404	1054	888
QTAL	409	314	229	217	190
Relative Reduction %	73.4	77.8	79.3	80.4	78.5

forms a set of query and translation augmented listings (QTAL). QTAL was used to train a language model. Its perplexity, measured on a held-out transcribed query set, was compared with the language model trained with a set of query augmented listings (QAL) – the QTAL set minus the paraphrases from the machine translation component. Table 2.2 shows significant perplexity reduction by applying this approach.

2.2.2 Spoken Language Understanding/Search

The task of spoken language understanding is to map a user's utterance to the corresponding semantics. In voice search, the semantics is the intended entry in a database. Hence the SLU becomes a search problem.

Traditionally, SLU is only an optional component for voice search. In early voice search applications like residential DA, SLU was not an issue, since there was not much expression variance in saying a person's name. Search was basically a database lookup, with careful considerations of initials, titles and homophones. If a finite state based LM is used for ASR, each rule is uniquely associated with a listing or a confusion set, there is no need of a separate search component either. However, due to the deficiency of the listing data based finite-state LM in modeling the actual language usage in queries, n-gram LMs are often adopted in recent more advanced voice search applications. In such a case, recognitions are no longer associated with a specific database listing. Hence a separate search step is necessary.

A majority of recent commercial products related to mobile local/Web search use a text-based Web search engine as the search component, and ASR is used as an interface to voice-enable the search engine. While this is a practical solution, it is sub-optimal because the search algorithms were developed without the consideration of robustness to ASR errors, and the voice search application has to adopt the same user interface as text search, which prohibits the system from taking full advantage of a spoken dialog system.

As voice search applications are getting more popular, SLU in voice search is becoming an important issue. Many applications in a special vertical may not have an existing text search engine, such as the task of music/video search on a local device. Some voice search applications based on the existing text search engines also introduced a separate SLU component (Feng et al. 2009). It is important for the SLU component to address the following problems:

1. **Improving the robustness to linguistic variance of spoken language**. An important difference between a spoken language only directory assistance dialog system and a speech-enabled local search on a mobile device lies in the characteristics of the input to the systems. In speech-enabled mobile search, users are directed to speak into a

specific search input box, hence their utterance are more likely to be similar to their text counterparts. In telephony directory assistance, users' speech may be more casual and contain "*carrier phrases*", for example, "*I need the number of* pizza hut in downtown." It is important to treat the carrier phrases differently from the name of a business listing during search.

2. **Handling the search from structured data**. Many backends of voice search applications have structured databases that contain multiple fields (columns). For example, product search may have a backend database with the brand, model, product name, category fields, etc. Music search may have a database with the artist, composer, title, and genre fields. Users may issue queries that specify information for more than one fields, like "Yellow Submarine by the Beatles." Local search may have the locality, business name, business category, opening hours, etc. Most existing text search engines adopt a bag-of-words approach that ignores the structural information.
3. **Improving the robustness to recognition errors**. Speech recognition is far from perfect. Directly feeding the one-best ASR results to a search engine is a suboptimal solution. A SLU component can bridge the information from the voice search semantic space (listings) and the information from the recognizer about the competing hypotheses, such that the ASR results that make more sense semantically can be chosen to improve the overall voice search performance.

Robustness to Linguistic Variance in Spoken Language

To improve the robustness to linguistic variance in spoken language, BBN adopted a channel model (Natarajan et al. 2002). Given a locality C[1] and a query Q recognized from a user's utterance, it looks for the listing $\hat{L}$ according to the following decision rule:

$$\begin{aligned}\hat{L} &= \arg\max_L P\left(L \mid C, Q\right) = \arg\max_L P\left(C, Q \mid L\right) P\left(L\right) \\ &\approx \arg\max_L P\left(C \mid L\right) P\left(Q \mid L\right) P\left(L\right)\end{aligned} \tag{2.13}$$

In (Natarajan et al. 2002), the prior distribution $P(L)$ and the locality distribution $P\left(C \mid L\right)$ were estimated from the training data. The training data were the transcripts of real users' utterances augmented with database listings. The query distributions $P\left(Q \mid L\right)$ were modeled with a two-state Hidden Markov Model (HMM) illustrated by Figure 2.4. In this model, a word w in Q is generated from either the General English (GE) state for carrier phrases or the state corresponding to a listing l, which is a value of the random variable L. With this model,

$$P(Q \mid L) = \prod_{w \in Q} \left(a_0 P\left(w \mid GE\right) + a_1 P\left(w \mid L\right)\right). \tag{2.14}$$

Here the transition weights a_0 and a_1 were tied across the HMMs for all values of L. The transition and emission probabilities were estimated from training data. This model is robust due to the inclusion of the GE state, which captures carrier phrases like "*I need the number*

[1]DA dialogs often start by asking users for the city and state information. See Section 2.2.3 for details.

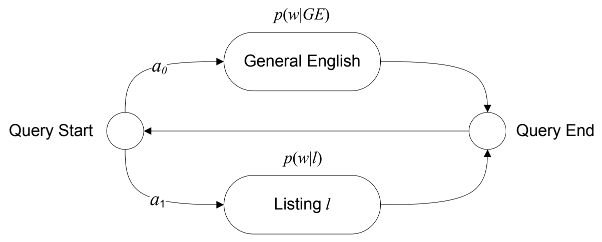


Figure 2.4 An HMM model is used to separate the listing terms from the general English terms (carrier phrases). The model is trained with a combination of users' queries and listing data (Reproduced from (Natarajan et al. 2002)).

of" or ASR errors. The combination of the real user data and the database listings facilitates high accuracy on frequently requested listings and simultaneously enables broad coverage of less frequently requested listings.

The HMM based listing selection requires training data, which is only realistic for a subset of listings that are most popularly sought for. Yu et al. (2007) applied the Tf-Idf (term frequency – inversed document frequency) weighted vector space model (VSM) to business listing and product name search. The Tf-Idf weighted vector space model is broadly used in information retrieval (IR). It represents a query (document) with a vector q (d). The relevance (or similarity) of the document to the query is measured as the cosine between the two vectors:

$$\cos(q, d) = \frac{q \cdot d}{\|q\| \, \|d\|} \tag{2.15}$$

For a document d, each element in its vector is a weight that represents the importance of a term (e.g., a word or a bigram) in the document. Intuitively, the importance should increase proportionally to the number of times the term appears in d and decreases when the term appears in many different documents. The *term frequency* (TF) $tf_i(d)$ is the frequency of the term i in d, and the *inverse document frequency* (IDF) is the logarithm of the total number of documents divided by the number of documents containing i:

$$tf_i(d) = \frac{n_i(d)}{\sum_k n_k(d)} \qquad idf_i = \log \frac{|D|}{|\{d : i \in d\}|} \tag{2.16}$$

where $n_i(d)$ is the number of occurrences of term i in d, and D is the entire document collection. The weight for term i in the vector is the product of its TF and IDF scores. The vector for a query can be defined similarly. There are other alternatives to define the TF and IDF scores, but the gist of the metrics are the same – TF measures the relevance of a term to a query/document, while IDF discounts the relevance if the term occurs in too many documents. The vectors for queries may sometime omit the IDF scores, since they are already taken into consideration by the IDFs in the document vectors. A survey of the vector space model can be found in (Greengrass 2001).

For voice search, each listing is treated as a "document" and represented by a vector. The standard VSM has been enhanced for voice search in (Yu et al. 2007) in the following two aspects:

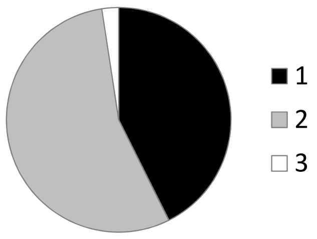
1. Special treatment of duplicate words in listings and queries. In traditional IR, documents and queries are generally long. The term frequency resembles the true distribution underlying a document/query. Listings and queries in voice search, on the other hand, are short in general, so the surface term frequency may not be a reliable estimate of the true underlying distribution. A small noise in the surface form is more likely to bring quite different search results. For example, the query "Big 5," intended for "Big 5 Sporting Goods", results in the listing "5 star 5" –the additional 5 in the listing brings it closer to the query. Since the term frequency is not reliable for search among short listings, each term gets a unit count in voice search. A duplicate word is treated as a different term, e.g., by replacing the second 5 in the example with 5_2nd. This effectively adds another dimension to the vector space. Since the Idf of this new dimension is much higher, it plays a more important role in query matching. A query without duplicate words like "Big 5" will have a larger angle from a listing with the duplicate "5". The angle will be significantly reduced if the query does contain the 2nd term. So "5 5" will match "5 star 5" better.
2. Inclusion of category information in the listing vectors in addition to the listing names. Callers often voluntarily provide category information (like restaurant, hospital, etc.) in their queries. These category words can be identified according to the mutual information between them and the categories in a database. If category information is detected in a user's query, the category information about a listing in the database can be appended to the listing's vector so it can be compared with the query's category directly by the vector space model. With this enhancement, the VSM would rank the listing "*Calabria Ristorante Italiano*" higher than "*Calabria Electric*" for the query "*Calabria Restaurant*".

Search for Structured Data

In mobile local search, the problem related to the structured data search is often alleviated by directly speech enable two separate input boxes, one for the location term[2] and the other for the search term. However, this solution does not fully take advantage of a spoken dialog system – as shown in (Feng et al. 2009), the separation of the two terms may add extra burden to the users – the distinction of the two terms is not always clear, as manifested by user's query like "restaurants near Manhattan" as the search term and "New York City" as the location term. In addition, the solution may not be sufficient to separate a search **subject** (often business names or categories) from the *constraints*, as in the exemplar query "**night clubs** *open Christmas day.*"

AT&T's Speak4It (Feng et al. 2009) allows users to include both the search term and the location term in a single utterance, and uses the frame-based SLU described in Chapter 1 to separate them from the utterance. More specifically, it uses n-gram model as the semantic prior model, and slot specific n-gram models as the lexicalization model – the same model used by CHRONUS.

[2]Same as the "locality" in the previously mentioned BBN's HMM model for voice search



All rise, I guess, from Blues
Sarah, in the arms of an angel
Play legally blonde soundtrack
Glenn Miller, jazz.
Anton Bruckner, 7th Symphony, Leonard Bernstein

Figure 2.5 Left: distribution of queries containing 1, 2, and 3 fields. Right: examples of queries specifying information for multiple fields.

To separate the search subjects from the constraints, it assigns a probability $P_{subject}(s)$ to a segment s for being a search subject based on the observation that the queries with a specific constraints occurs far fewer than the queries without any constraints (e.g., "night club" is more popular than "night club open Christmas day."). A corpus C of likely subjects is constructed to include the simple local search queries and the listing names. Here a simple query is one with no more than five words and containing no constraint indicators like the word "with" or "that." $P_{subject}(s)$ can be estimated from C:

$$P_{subject}(s) = \left(\lambda * \frac{Freq(s \in C)}{|C|} + (1-\lambda) * \frac{1}{|C|}\right)^{\gamma} \tag{2.17}$$

In (Song et al. 2009), HMM based models were introduced to handle search from structured music meta-data, where a query may contains multiple fields. It observed that more than half of users' queries had contained the specification for more than one fields (Figure 2.5.) An HMM can model the field-specific information in a query. Formally, given a query $Q = w_1, \ldots, w_n$, we need to find the entry $\hat{E}$ from the database, such that

$$\begin{aligned} \hat{E} &= \arg\max_E P(E \mid Q) = \arg\max_E P(Q \mid E)P(E) \\ &\approx \arg\max_E P(Q \mid E) = \arg\max_E \sum_F P(Q, F \mid E) \end{aligned} \tag{2.18}$$

here F is the segmentation of Q into multiple fields, F_i represents the field that the word w_i belongs to. While $P(E)$ can be modeled with the music popularity statistics, an uniform distribution is used in Eq. (2.18) instead. $P(Q, F \mid E)$ is a distribution that is specific to the entry E, which is modeled by an HMM:

$$P(Q, F \mid E) \approx \prod_{i=1}^{n} P(w_i \mid F_i; E)P(F_i \mid F_{i-1}; E) \tag{2.19}$$

Unlike the HMMs used by BBN for information retrieval, the HMMs here do not need any training data. The emission probabilities for a field are estimated with the database content of the field:

$$P(w \mid F, E) = \lambda_f P_{MLE}(w \mid F, E) + \lambda_e P_{MLE}(w \mid E) + \lambda_c P_{MLE}(w) \tag{2.20}$$

where $\lambda_f + \lambda_e + \lambda_c = 1$. In other words, the emission probability is the linear interpolation of the maximum likelihood estimate of the field dependent unigram, the entry dependent unigram and the entry-independent unigram probability. While the interpolation weights can be set using held-out data, it is found that the search performance is not very sensitive to their values as long as none of the weights is set too close to 0.

The transition probability is assigned as follows to penalize frequent field hopping:

$$P(F_i \mid F_{i-1}; E) = \begin{cases} \gamma & \text{if } F_i = F_{i-1} \\ (1-\gamma) \times P(F_i \mid E) & \text{otherwise} \end{cases} \tag{2.21}$$

where γ is a parameter related to the likelihood that a query will stay at the specification of one field of a structured entry. A large value for the parameter will effectively avoid frequent field jumping. Its value can be tuned with a held-out set. $P(F \mid E)$ is the field popularity probability, which is the field prior probability $P(F)$ normalized according to the existence of the fields in the database for the entry E:

$$P(F = x \mid E) = \begin{cases} \frac{P(F=x)}{1.0 - \sum_{\forall y \notin E} P(F=y)} & \text{if } x \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where $P(E)$ can be either a uniform distribution or a distribution determined based on the domain knowledge about the popularity of the fields in users' queries.

In (Song et al. 2009), it has shown that the HMM sequential rescoring model has reduced the end-to-end search error rate by 28% relatively on text queries and up to 23% relatively on spoken queries compared to the baseline system that used language model IR on the database with the contents of different fields collapsed into a single bag of words. In addition, the paper has introduced an error model for HMM emissions based on phonetic confusability, which has further improved the end-to-end search accuracy consistently across different levels of speech recognition accuracy.

Robustness to Speech Recognition Errors

To improve the robustness to ASR errors, Wang et al. (2008) used character n-gram unigrams/bigrams instead of word unigrams/bigrams as terms in the vector space model. The rationale is that the acoustically confusable words may have shared sub-word units orthographically. For example, the listing "Lime Wire" is rewritten as a sequence of character 4-grams – \$Lim Lime ime_ me_W e_Wi _Wir Wire ire\$, where "\$" indicates the start and the end of the listing and "_" indicates separation of words. If a caller's query "Lime Wire" is incorrectly recognized as "Dime Wired", there is no word overlapping but still much character n-gram overlapping between the ASR output and the intended listing.

Feng et al. (2009) used the listing information to select an ASR path in the word confusion network, a compact representation of the recognition lattice. Figure 2.6 shows an example of the word confusion network, where each position in an utterance is associated a set of confusable words with their negative log posterior probabilities obtained from the speech recognizer. Although the one-best path from the network, "Gary Crites Springfield Missouri," has the listing names "Dairy Queen" mis-recognized as "Gary Crites," The correct information is buried in the word confusion network. The knowledge from listing in the semantic space can help unearth the correct recognition – "Dairy Queen" is more like to be a valid listing name than "Gary Crites."

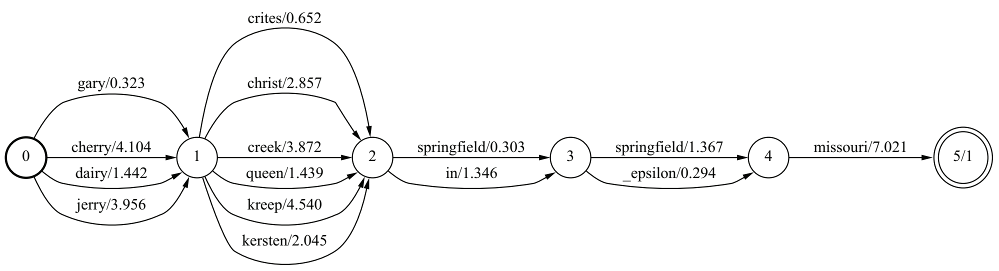


Figure 2.6 A word confusion network is a compact representation of recognition lattice that lists a set of words for each position (Courtesy of Junlan Feng).

Feng et al. (2009) segments the one-best ASR to find the search term, gets the alternative phrases s of the search term in the confusion networks, and rescore them with the subject likelihood:

$$P(s) = P_{WCN}(s)P_{subject}(s)^{\lambda}, \tag{2.23}$$

where $P_{WCN}(s) = \prod_{w \in s} P_{wcn}(w)$ is the posterior probability of the segment s according to the word confusion network, $P_{subject}(s)$ has been introduced earlier for subjects/constraints separation.

Similar idea was investigated with a flat direct model for speech recognition in (Heigold et al. 2009). It is basically a Maximum Entropy (MaxEnt) model used to rescore the n-best search results. MaxEnt is a condition model $P(\mathbf{y} \mid \mathbf{x})$ defined with respect to a set of features $\mathcal{F} = \{f_k(\mathbf{x}, \mathbf{y})\}$, with the constraints that the expected value of a feature predicted according to the conditional distribution equals to the empirical value of the feature observed in the training data:

$$\begin{aligned} \mathbb{E}_{\tilde{P}(\mathbf{x})P(\mathbf{y} \mid \mathbf{x})} f_k(\mathbf{x}, \mathbf{y}) &= \sum_{\mathbf{x},\mathbf{y}} \tilde{P}(\mathbf{x})P(\mathbf{y} \mid \mathbf{x}) f_k(\mathbf{x}, \mathbf{y}) \\ = \mathbb{E}_{\tilde{P}(\mathbf{x},\mathbf{y})} f_k(\mathbf{x}, \mathbf{y}) &= \sum_{\mathbf{x},\mathbf{y}} \tilde{P}(\mathbf{x}, \mathbf{y}) f_k(\mathbf{x}, \mathbf{y}), \forall f_k(\mathbf{x}, \mathbf{y}) \in \mathcal{F} \end{aligned} \tag{2.24}$$

where $\tilde{P}$ stands for empirical distributions over a training set. There can be many possible distributions $P(\mathbf{y} \mid \mathbf{x})$ that satisfies Eq. (2.24). The maximum entropy principle states that the target distribution should have the maximum entropy subject to the condition of Eq. (2.24). In other words, the model should make no more assumptions other than the expected feature values according to the empirical distribution from the training data.

It has been proven that the maximum entropy distribution subject to the condition of Eq. (2.24) have the following exponential (log-linear) form (Berger et al. 1996):

$$P(\mathbf{y} \mid \mathbf{x}; \Lambda) = \frac{1}{Z(\mathbf{x}; \Lambda)} \exp\left\{\sum_k \lambda_k f_k\left(\mathbf{y}, \mathbf{x}\right)\right\} \tag{2.25}$$

As in CRFs, $\Lambda = \{\lambda_k\}$ is a set of parameters. The value of λ_k determines the impact of the feature $f_k(\mathbf{y}, \mathbf{x})$ on the conditional probability, and can be estimated with numerical

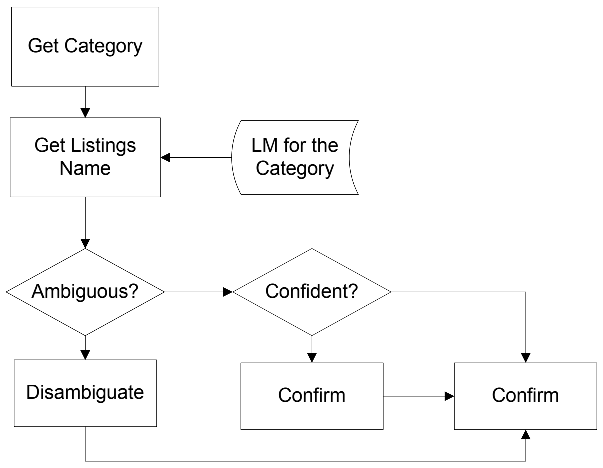


Figure 2.7 A typical dialog flow of a voice search application.

algorithms by maximizing the conditional probability of the training data $\sum_{\mathbf{x},\mathbf{y}} P(\mathbf{y} \mid \mathbf{x})$. $Z(\mathbf{x}; \Lambda) = \sum_{\mathbf{y}} \exp\left\{\sum_k \lambda_k f_k(\mathbf{y}, \mathbf{x})\right\}$ in Eq. (2.25) is a partition function that normalizes the distribution. The difference between MaxEnt and CRFs lies in the form of the label **y**. In the CRFs, **y** is a sequence of random variables that may be inter-dependent, while in the MaxEnt model, **y** is a single random variable, which make the model simpler in terms of learning and inference.

Heigold et al. (2009) used a rich set of features extracted from the candidate listing **y** and from the speech input **x**, where **x** was consists of not only the MFCC features but also the n-best results from the recognizer. The feature set included character and/or word n-grams, the posterior probability of word n-grams spotted in **x** with a dynamic time warping (DTW) algorithm, etc. It found that the direct flat model had reduced the error rate of a HMM recognition baseline from 17.4% to around 13%.

2.2.3 Dialog Management

Figure 2.7 shows the common dialog strategy in voice search applications. The dialog starts with prompting a user for the category information about the item they are looking for to narrow down the downstream LM and search spaces. The category can be the locality information in a DA system (Bacchiani et al. 2008; Yu et al. 2007); the business/product separation (national business, local business or product) in a voice rating system (Zweig et al. 2007), or a “search-by” attribute of the music meta-data (e.g., title, album, genre, artist, etc) in a music search dialog system (Mann et al. 2007). A category-specific LM is subsequently used to recognize the user’s query containing the listing information, and the search component looks for the listing in a category-specific database. If multiple listings are found, a disambiguation sub-dialog is engaged; otherwise the dialog system either directly sends the user the listing information or asks for user’s confirmation if the confidence score is low.

***User:** Tell me about restaurants in London.*
***System:** I know of 596 restaurants in London. All price ranges are represented. Some of the cuisine options are Italian, British, European, and French.*
***User:** I'm interested in Chinese food.*
***System:** I know of 27 restaurants in London that serve Chinese cuisine. All price ranges are represented. Some are near the Leicester square tube station.*
***User:** How about a cheap one?*
***System:** I know of 14 inexpensive restaurants that serve Chinese cuisine. Some are near the Leicester Square tube station. Some are in Soho.*

Figure 2.8 Search result summarization and disambiguation.

Many voice search applications adopt some task-specific dialog strategies. For example, based on the finding that the accuracy on spelled names is much higher than that on spoken names, the residential DA in (Schramm et al. 2000) exploited a multi-stage dialog strategy to improve the accuracy of proper name recognition. A listing was identified by first recognizing its spelling from a caller. The spelling word graph greatly reduced the listing space for subsequent recognition of names and addresses.

Disambiguation

Most voice search dialog systems adopt an application-specific disambiguation strategy. In residential DA, people with the same name are disambiguated with their addresses (Schramm et al. 2000). In business DA, business categories are used for disambiguation (Yu et al. 2007) – from the set of businesses returned by the search component, a list of possible categories is compiled. For example, the query "*Calabria*" results in multiple search results, "Calabria Ristorante Italiano" in category "**Restaurants**", "*Calabria Jack J Do*" in "**Doctors and Clinics**" and "Calabria Electric" in "**Electric Contractors**." These categories are read to the user for selection. All the matching listing names in the selected category are subsequently read to the user, until one is selected or the list is exhausted. Similar disambiguation strategy is used in a multi-modal voice search application (Mann et al. 2007), where multiple music titles are displayed in a graphical user interface (GUI) for users' selection when they belong to the same category, or the different categories are displayed first for disambiguation. The GUI allows users to scan the information visually, which makes the multi-modal interaction more effective.

One problem of the hard wired disambiguation strategy is its inefficiency with long category/entry lists in a speech only interface. It has been suggested that spoken dialog strategies such as summaries are a verbal equivalent of the visual scanning behavior that makes GUIs effective (Polifroni and Walker 2006). Hence summarization can be used when the search component returns a big ambiguous set. Figure 2.8 shows an exemplar dialog taken from (Polifroni and Walker 2006). Here the ambiguous listings are summarized along common attributes like price ranges and cuisines, which guide users to provide the most effective information for disambiguation. In contrast to the hard wired disambiguation strategy, the attributes were selected automatically by using a decision theoretic user model and using the association rules derived from the database subset in the dialog focus (Polifroni and Walker 2006).

Confidence Measure

Confidence measures are used to determine what to do with the search results for a spoken query. The results will be played to callers if the confidences are high, otherwise a confirmation/disambiguation sub-dialog will be invoked. Confidence measures are also used to determine when to elevate an automated service conversation to a live agent in an early dialog stage if the confidence on the key information (e.g., an individual's last name in a residential DA system) is too low (Schramm et al. 2000).

ASR only confidence measures were used in many early residential DA systems because search was not a significant source of uncertainty. A well studied confidence measure is the word or sentence posterior probability that can be calculated from an ASR lattice, which was shown to be more effective than some other heuristics (Wessel et al. 2001). A sentence posterior probability obtained from an n-best list was used in (Schramm et al. 2000) for DA. Another confidence measure originally proposed for utterance verification (Lee 1997) was applied in (Béchet et al. 2000). It is based on hypothesis testing that leads to a measure of likelihood ratio.

In late voice search applications where statistical search is applied for robustness, confidence measures that take into account of uncertainties from different system components are more adequate. BBN's DA system applied a Generalized Linear Model classifier to compute confidence score from a set of features extracted from spoken queries and listings (Natarajan et al. 2002). The feature set included word confidences, ASR n-best frequency, etc. Among them, the two most important features were the required and allowable word sets. Much like IBM's signatures, the required word set for a listing is a set of word tuples, at least one of which must be present in a recognized query in order to associate the listing with the query. The allowable word set is a list of words that are allowable in a query to be associated with the listing.

A confidence model based on a maximum entropy classifier was introduced for the Microsoft Research's experimental business DA system (Wang et al. 2007). Unlike the required and allowable set features in (Natarajan et al. 2002), it takes into consideration the importance of words in a listing with features based on the automatically acquired Idf statistics of the word. The classifier takes multiple features drawn from the ASR, the search component and the dialog manager, and the combined features extracted from multiple components. For example, the *search related features* for a hypothesized listing L and a recognized query Q include the VSM similarity between L and Q; the ratio between the maximum Idf value among the words existing in both L and Q and the maximum Idf value among all the words in L. The *combined features* attempt to model the dependency among features across different components of voice search. One such feature is the ASR confidence on the word that also exists in L and has the highest Idf value, i.e. the ASR confidence on the word that contributes the most to the search result. The effectiveness of the features were studied with statistical significance tests, which gave rise to several application-independent features for confidence measures in the general voice search framework (Wang et al. 2007).

2.2.4 Closing the Feedback Loop

Every spoken dialog system needs to be tuned, often through multiple iterations, for improved performance. This involves a painstaking process of error analysis from logged data. An automatic or semi-automatic tuning tool is one of the most wanted items by many dialog experts. Due to the extreme difficulty of the problem, little work has been seen on automatic remedy for design/implementation flaws in the feedback loop. Most research work focused on automatic flaw discovery from logged data.

In (Popovici et al. 2002), an unsupervised learning algorithm was proposed to obtain the linguistic variants of listings that were not modeled in the Telecom Italia's DA system. A phone-looped model was exploited to obtain the phonetic transcriptions for the utterances that failed the automated service and got routed to the operators. The phonetic transcriptions were clustered with a furthest neighbor hierarchical clustering algorithm, where two clusters with the shortest distance were merged in iterative steps. The distance between two clusters was defined as the furthest distance between two instance phonetic transcriptions in the clusters, and the distance between two phonetic transcriptions was obtained with the Viterbi alignment using the log-probability for phone insertion, deletion and substitution, where the probabilities were trained using a set of field data by aligning each decoded phonetic sequence with its corresponding manual transcription. A cluster in the hierarchy was selected according to the following criteria – the number of instances in the cluster must exceed a threshold and the dispersion of the cluster must be smaller than another threshold. The central element of a selected cluster was presented to a spoken dialog expert as a candidate variant of a business listing.

A similar algorithm was proposed in (Li et al. 2005) to discover the semantic intents that were not covered by an auto-attendant spoken dialog system in Microsoft (Ollason et al. 2004). The system was originally designed to connect a caller to a Microsoft employee with name dialing. It was later found that in addition to name dialing an employee, callers often ask for connections to an office, such as "*security*" or "*shuttle service.*" To discover these uncovered intents, a language model based acoustic clustering algorithm was proposed. Unlike the algorithm in (Popovici et al. 2002) that clusters the one-best phonetic transcriptions, it treats the *word* transcription and the cluster they belong to as hidden variables, and optimizes the parameters associated with them with respect to an objective function. Specifically, given a fixed number of clusters, it builds a cluster-specific language model $P(w \mid c)$ and a cluster prior model $P(c)$ to maximize $P(x) = \sum_{c,w} P(x, w, c) = \sum_{c,w} P(x \mid w) P(w \mid c) P(c)$, the likelihood of the observed acoustic signal x. In practice, recognition was decoupled from cluster training – a task independent large vocabulary ASR was used to obtain the hypotheses w and their posterior probabilities $P(w \mid x)$. Since w and c are hidden variables, the Expectation-Maximization (EM) algorithm was used to estimate the probability $P(c)$ and $P(w \mid c)$ by maximizing the objective function. Here the EM algorithm took as input the hypotheses w and $P(w \mid x)$ obtained from the task-independent ASR. In (Li et al. 2005), unigram language models were used for $P(w \mid c)$. With these cluster-specific distributions, a KL-divergence based distance measure was used in hierarchical clustering. The EM algorithm was subsequently applied for several iterations to re-estimate the model parameters after merging two clusters. The cluster priors obtained from the EM algorithm was used to rank the clusters for presentation to spoken dialog experts.

2.3 Summary

This chapter reviews the problem of voice search, which is one of the most actively investigated technology underlying many practical applications. We have compared voice search with other spoken language understanding technologies for human-computer interaction, discussed the challenges in developing voice search applications, and reviewed some important research work targeting at these problems. As in other SLU applications, robustness is the central issue in voice search. The technology in acoustic modeling aims at improved robustness to environment noise, different channel conditions and speaker variance; the pronunciation research addresses the problem of unseen word pronunciation and pronunciation variance; the language model research focuses on linguistic variance; the studies in SLU/search give rise to improved robustness to linguistic variance and ASR errors; the dialog management research enables graceful recovery from confusions and understanding errors; and the learning in the feedback loop speeds up system tuning for more robust performance.

While tremendous achievements have been accomplished in the past decade on voice search, big challenges remain. Many voice search dialog systems have automation rates around or below 50% in field trials. This provides a fertile ground and great opportunities for future research.

References

Acero A, Chambers R, Ju YC, Li X, Odell J, Nguyen P, Schölz O and Zweig G 2008 Live search for mobile: Web services by voice on the cellphone. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, Las Vegas, Nevada, USA.

Andreani G, Fabbrizio GD, Gilbert M, Gillick D, Hakkani-Tür D and Lemon O 2006 Lets DiSCoH: Collecting an annotated open corpus with dialogue acts and reward signals for natural language helpdesk. *Proceedings of the IEEE/ACL Workshop on Spoken Language Technology*, pp. 218–221, Aruba.

Bacchiani M, Beaufays F, Schalkwyk J, Schuster M and Strope B 2008 Deploying GOOG-411: Early lessons in data, measurement, and testing. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5260–5263.

Béchet F, de Mori R and Subsol G 2002 Dynamic generation of proper name pronunciations for directory assistance. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE.

Béchet F, Os Ed, Boves L and Sienel J 2000 Introduction to the IST-HLT project speech driven multimodal automatic directory assistance (SMADA) . *Proceedings of the 6th International Conference on Spoken Language Processing*, pp. 731–734, Beijing, China.

Berger AL, Della Pietra SA and Della Pietra VJ 1996 A maximum entropy approach to natural language processing.. *Computational Linguistics* **22**(1), 39–72.

Billi R, Cainavesio F and Rullent C 1998 Automation of Telecom Italia directory assistance service: fieldtrial results. *Proceedings of the 4th IEEE Workshop on Interactive Voice Technology for Telecommunications Applications*.

Bohus D, Puerto SG, Huggins-Daines D, Keri V, Krishna G, Kumar R, Raux A and Tomko S 2007 ConQuest - an open-source dialog system for conferences. *Proceedings of the Human Language Technology Conference and the Conference of North American Chapter of the Association for Computational Linguistics*, pp. 9–12.

Buntschuh B, Kamm C, Fabbrizio GD, A. Abella MM, Narayanan S, Zeljkovic I, Sharp RD, Wright J, Marcus S, Shaffer J, Duncan R and Wilpon JG 1998 VPQ: A spoken language interface to large scale directory information. *Proceedings of the International Conference on Spoken Language Processing*, pp. 2863–2866, Sydney Australia.

Feng J, Banglore S and Gilbert M 2009 Role of natural language understanding in voice local search. *Proceedings of INTERSPEECH*.

Gao Y, Ramabhadran B, Chen J, Erdogan H and Picheny M 2001 Innovative approaches for large vocabulary name recognition. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 53–56.

Greengrass E 2001 *Information Retrieval: A Survey*.

Heigold G, Zweig G, Li X and Nguyen P 2009 A flat direct model for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3861–3864, Taipei, Taiwan.

Jan E, Maison B, Mangu L and Zweig G 2003 Automatic construction of unique signatures and confusable sets for natural language directory assistance applications. *Proceedings of the Eurospeech Conference*, pp. 1249–1252.

Kamm CA, Shamieh CR and Singhal S 1995 Speech recognition issues for directory assistance applications.. *Speech Communication* **17**(3–4), 303–311.

Lee CH 1997 A unified statistical hypothesis testing approach to speaker verification and verbal information verification. *Proceedings of COST 250*, pp. 63–72, Rhodes, Greece.

Lennig M, Bielby G and Massicotte J 1994 Directory assistance automation in Bell Canada: trial results. *Proceedings of the 2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA94)*, pp. 9–13. IEEE.

Li X, Gunawardana A and Acero A 2005 Unsupervised semantic intent discovery from call log acoustics. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 45–48, Philadelphia, PA, USA.

Li X, Ju YC, Zweig G and Acero A 2008 Language modeling for voice search: a machine translation approach. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, Nevada, USA.

Mann S, Berton A and Ehrlich U 2007 How to access audio files of large data bases using in-car speech dialogue systems. *Proceedings of INTERSPEECH*, pp. 138–141, Antwerp, Belgium.

Natarajan P, Prasad R, Schwartz RM and Makhoul J 2002 A scalable architecture for directory assistance automation. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. I–21–I–24.

Ollason D, Ju YC, Bhatia S, Herron D and Liu J 2004 MS Connect: A fully featured auto-attendant: System design, implementation and performance . *Proceedings of the International Conference on Spoken Language Processing*, pp. 2845–2848, Jeju, Korea.

Polifroni J and Walker M 2006 An analysis of automatic content selection algorithms for spoken dialogue system summaries. *Proceedings of the IEEE/ACL Workshop on Spoken Language Technology*, Aruba.

Popovici C, Andorno M, Laface P, Fissore L, Nigra M and Vair C 2002 Learning new user formulations in automatic directory assistance. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. I–17– I–20.

Ramabhadran B, Bahl LR, deSouza P and Padmanabhan M 1998 Acoustics-only based automatic phonetic baseform generation. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 309–312.

Schramm H, Rueber B and Kellner A 2000 Strategies for name recognition in automatic directory assistance systems.. *Speech Communication* **31**(4), 329–338.

Song YI, Wang YY, Ju YC, Seltzer M, Tashev I and Acero A 2009 Voice search of structured media data. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Taipei, Taiwan.

Vertanen K and Kristensson PO 2009 Recognition and correction of voice web search queries. *Proceedings of INTERSPEECH*, pp. 1863–1866.

Wang YY, Yu D, Ju YC and Acero A 2008 An introduction to voice search.. *IEEE Signal Processing Magazine* **25**(3), 29–38.

Wang YY, Yu D, Ju YC, Zweig G and Acero A 2007 Confidence measures for voice search applications. *Proceedings of INTERSPEECH*, pp. 2721–2724, Antwerp, Belgium.

Wessel F, Schlüter R, Macherey K and Ney H 2001 Confidence measures for large vocabulary continuous speech recognition.. *IEEE Transaction on Speech and Audio Processing*.

Yu D, Ju YC, Wang YY and Acero A 2006 N-gram based filler model for robust grammar authoring. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. I565–I568, Toulouse, France.

Yu D, Ju YC, Wang YY, Zweig G and Acero A 2007 Automated directory assistance system - from theory to practice. *Proceedings of INTERSPEECH*, pp. 2709–2712, Antwerp, Belgium.

Zweig G, Ju YC, Nguyen P, Yu D, Wang YY and Acero A 2007 Voice-rate: a dialog system for consumer ratings. *Proceedings of NAACL/HLT (Demonstration)*, pp. 31–32. Association for Computational Linguistics, Rochester, New York, USA.