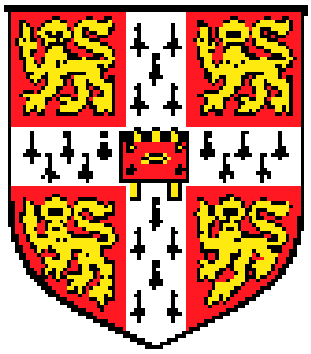
A Statistical Approach to Spoken Language Understanding



Yulan He

New Hall College

University of Cambridge

A thesis submitted to the University of Cambridge for the degree of *Doctor of Philosophy*

August 2004

I would like to dedicate this thesis to my parents who love me, support me for so long and my husband who is always there for me

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where stated. It has not been submitted in whole or part for a degree at any other university. The length of this dissertation including footnotes does not exceed 65000 words and it contains 34 figures.

Acknowledgements

First and foremost, my gratitude goes to my supervisor, Steve Young, for his guidance, patience, and tremendous support throughout my PhD course in Cambridge. Without his invaluable advice, it would not have been possible for me to complete the PhD studies in three years.

I would like to thank my departmental colleagues, Andrew Liu, Lan Wang, and KK Ye for their help, encouragement and sharing happy hours during lunchtime and afternoon tea breaks, which made my stay here pleasant.

Thanks are also due to Gunnar Evermann, Do Yeong Kim, Phil Wooland, Mark Gales, Patrick Gosling, and Anna Langley for assistance with the HTK Toolkit, advice on my project, and help on various aspects of the group's computer systems.

I would also like to thank my parents and my husband for loving me, for supporting me, and for encouraging me to be the best I could be.

Finally, I am grateful to the Nanyang Technological University, Singapore, for sponsoring my studies.

Summary

The development of spoken dialogue systems currently rely on computational linguists to define semantic grammar rules, to produce templates for appropriate response generation, to build the lexicon, etc. Reducing this need for manual processing and automating the whole process as much as possible are thus essential steps to reducing development costs and improving robustness. The research work described here focuses on statistical learning approaches for building a purely data-driven spoken language understanding (SLU) system whose three major components, the speech recognizer, the semantic parser, and the dialogue act decoder are trained entirely from data. The system is comparable to existing SLU systems which rely on either hand-crafted semantic grammar rules or statistical models trained on fully-annotated training corpora but it has greatly reduced build cost.

The core of the system is a novel hierarchical semantic parser model called a *Hidden Vector State (HVS)* model. Unlike other hierarchical parsing models which require fully-annotated treebank data for training, the HVS model can be trained using only lightly annotated data whilst simultaneously retaining sufficient ability to capture the hierarchical structure needed to robustly extract task domain semantics.

The HVS parser is combined with a dialogue act detector based on Naive Bayesian networks which have been extended and refined by introducing Tree-Augmented Naive Bayes networks (TANs) to allow inter-concept dependencies to be robustly modelled.

Finally, the two semantic analyzer components, the HVS semantic parser and the modified-TAN dialogue act decoder, have been integrated with a standard HTK-based Hidden Markov Model (HMM) speech recognizer and the additional knowledge provided by the semantic analyzer has been used to determine the best-scoring word hypothesis from the N-best lists generated by the speech recognizer. This purely data-driven spoken language understanding (SLU) system has been built and tested using both the ATIS and DARPA Communicator test sets. In addition to testing on clean data, the systems has been tested on various levels of noisy data and on modified application domains. The results support the claim that an SLU system which is statistically-based and trained entirely from data is intrinsically robust and can be readily adapted to new applications.

Keywords: semantic parsing, hidden vector state model, dialogue act detection, tree-augmented Naive Bayes, spoken language understanding, spoken dialogue systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Spoken Dialogue Systems

In the last decade, spoken dialogue interfaces have attracted much attention in the research community. One reason is that conversation or dialogue is the most natural way for people to exchange ideas, convey information, and express feelings, etc. Another motivation is that the ideal spoken dialogue system would allow the user to interact with the machine in natural and intuitive ways similar to human-human interaction without learning complicated usage instructions. A large number of spoken dialogue systems have been developed, both as research prototypes and commercial applications. Such systems try to understand and act upon what people say.

Spoken dialogue systems can be broadly classified into three categories, the system-initiative model, the user-initiative model, and the mixed-initiative model. The system-initiative model assumes complete control in guiding the user throughout the interaction. Conversely, the user-initiative model allows the user to determine his preferred course of interaction. The former often attains high task completion rates but imposes too many constraints on the user. The latter offers maximum flexibility to the user but has a lower task completion rate. In order to strike the balance between these two models, the mixed-initiative model has been widely adopted which allows both the system and the user to influence the course of interaction.

1.1.1 Typical Architecture

Typically, a spoken dialogue system requires a speech recognizer to recognize the user's utterances, a semantic analyzer to interpret the meaning of the recognized utterance, a dialogue manager to control the interaction with the user, a response generator to produce the appropriate replies, and a speech synthesizer to construct the acoustic signals based on

the response generated. The general architecture of a spoken dialogue system is illustrated in Figure 1.1.

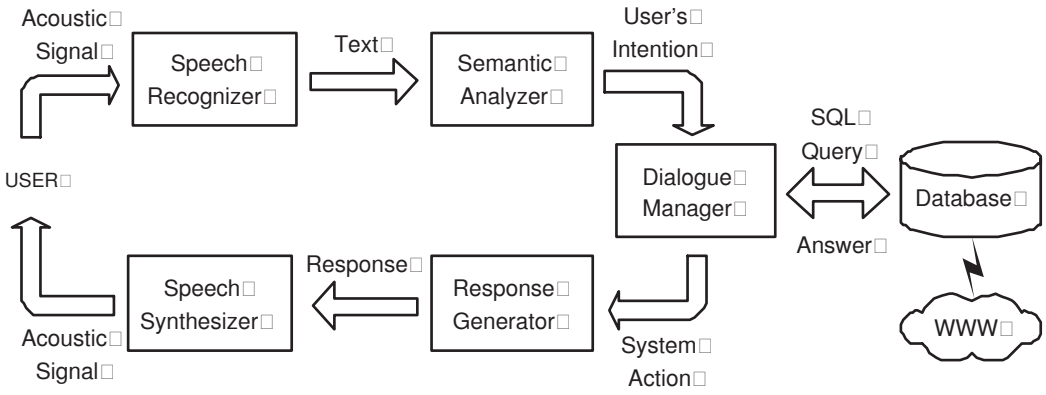


Figure 1.1: Typical structure of a spoken dialogue system.

The usability of a spoken dialogue system relies on the accuracy of its various component technologies, including speech recognition, natural language understanding for semantic processing, local discourse handling, and speech synthesis. The work here mainly focuses on semantic processing, which extracts meanings from recognized word strings and infers the user's dialogue acts or information goals based on the current input and dialogue context.

1.1.2 Projects and Applications

Substantial research has been done in spoken dialogue systems. Among the various spoken dialogue projects, the most influential ones include the U.S. DARPA ATIS program [1], the European ESPRIT SUNDIAL (Speech Understanding and DIALog) project [2], and the DARPA Communicator project [3]. From 1990 to 1995, DARPA sponsored a spoken language understanding (SLU) program to objectively measure the performance of various SLU systems. Different research sites [4; 5; 6; 7; 8] worked on the same domain, the Air Travel Information Service (ATIS) [9], data for which were collected jointly by them. The utterance understanding error rates for spoken input in the December 1994 benchmarks range from 6.5% to 44.9% for context-independent utterances (class A). The ESPRIT SUNDIAL project involved several sites working on the development of prototypes for train timetable queries in German and Italian, and flight queries in English and French. All these systems are described in [10]. More recently, the DARPA Communicator project [3] aims to support rapid, cost-effective development of multi-modal speech-enabled dialog systems. Members of the Communicator programme include AT&T, BBN, CMU, University of

Colorado (CU), IBM, MIT, MITRE, and SRI, most of whom previously participated in the DARPA ATIS programme.

Besides flight reservation or train timetable queries, spoken dialogue systems have also been applied to call routing [11; 12], accessing personal information such as retrieving voice mails, emails etc [13; 14], banking transactions such as paying a bill, transferring funds between accounts [15] or querying mortgage loans [16], voice-automated trading of stocks [17] etc.

1.2 Semantic Processing

The semantic analyzer in Figure 1.1 can be further decomposed into two modules, the semantic parser and the dialogue act decoder, as shown in Figure 1.2. The semantic parser maps the input word string onto a sequence of predefined semantic labels or concepts, while the dialogue act decoder predicts the user's goals based on the current concepts and the dialogue state.

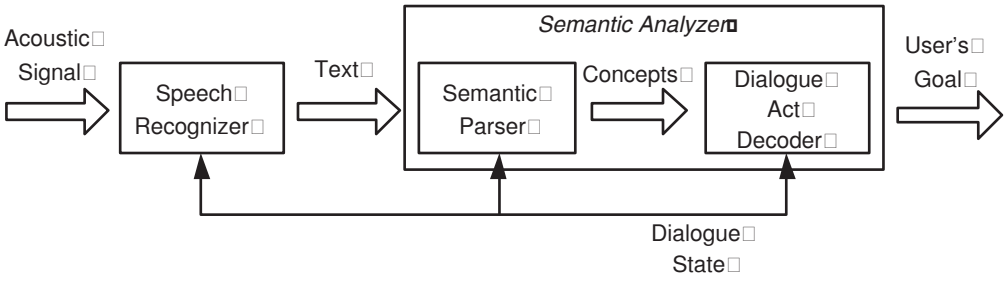


Figure 1.2: Semantic analyzer.

The outputs of a speech recognizer are normally 1-best or sometimes N-best word string hypotheses. Traditional speech recognition approaches depend on language models supported by semantic post-processors to eliminate linguistically invalid or semantically meaningless sentences from the word string list. In spoken dialogue systems, the dialogue context can be incorporated into the language models to narrow down the word string hypothesis space. That is, if a certain dialogue act is expected from the semantic analyzer, the language model should be able to predict it. Based on this, two kinds of language models can be used in the spoken dialogue system, a context-independent language model for the speech recognizer and a context-dependent language model for the semantic analyzer. The selection of the best hypothesis may be left until the dialogue act decoding stage.

1.3 Limitations of Spoken Dialogue Systems

Spoken dialogue systems provide an interface between the user and a computer-based application that permits spoken interaction with the application in a relatively natural manner. However they cannot fully model human-human conversations due to the following limitations:

- **Spontaneous speech**. Spontaneous speech is often not grammatically well-formed. It may contain sentence fragments, self-corrections, slips of the tongue, or ungrammatical combinations. In such cases, robust parsing is normally applied which does not perform a complete analysis of the recognized text but instead tries to recover chunks that can be used to extract essential items of meaning in the text. It usually requires hand-crafted grammar rules to be defined.

- **Response generation**. Most dialogue systems are essentially driven by template-filling mechanisms in which the systems try to elicit the information required to fill a template and the system responses are often determined by what is missing or needs to be clarified or confirmed in the template. Again, such template definitions require significant human efforts and are difficult to maintain and adapt.

- **Robustness**. People's spoken language is highly variable as different people use different words and sentence structures to convey the same meaning. It thus remains an open issue as to how to provide robustness for large populations of non-expert users. If the system lexicon is kept large enough to cover every possible word, high user flexibity may be achieved but the system accuracy will be poor. On the other hand, if the user input is constrained, the system accuracy may be increased but with decreased user flexibility.

- **Portability**. Spoken dialogue systems are highly domain-specific as they are normally designed only for particular applications. It involes high cost to port a system to a new application domain, especially when semantic grammar rules and the design of the dialogue strategy must be hand-crafted by the system developer. Though some systems can modularize certain common grammar rules such as those concerning dates, times etc, it is unavoidable in a new application to define new domain-specific grammar rules. Therefore, less labour intensive methods of porting or adapting systems to new domains are worth investigating.

It can be seen from the above that the development of spoken dialogue systems rely heavily on computational linguists to define semantic grammar rules, to produce templates for appropriate response generation, to build the lexicon etc. Reducing the need

for this manual processing and automating the whole process as much as possible are thus of particular interest. In addition, each individual component of spoken dialogue systems have to be integrated to produce a robust and efficient real-time system. This leads to system refinement and optimization. One good component may often compensate the weakness of another component. For example, a semantic parser may still be able to generate the correct meaning of an utterance even it has been recognized wrongly by a speech recognizer. Therefore, high-performance spoken language understanding components will eventually lead to a good spoken dialogue system. The research work here focuses on machine learning algorithms for semantic processing which is the key to the spoken language understanding process. Two major stages of semantic processing, semantic parsing and dialogue act detection, have been investigated and an integrated spoken understanding system which combines robustness and low build cost has been built.

1.4 Thesis Contribution

Given the limitations of current spoken dialogue systems described in Section 1.3, it would be desirable to develop pure data-driven or automatic learning techniques for spoken language understanding. The contribution of this thesis work can be summarized in the following aspects.

- A novel hierarchical semantic parser model, *Hidden Vector State (HVS)* model, has been proposed and implemented. Unlike other hierarchical parsing models which require fully-annotated treebank data for training, it is trainable using only lightly annotated data whilst simultaneously retaining sufficient ability to capture the hierarchical structure need to robustly extract task domain semantics.

- Dialogue detection using Bayesian networks has been extended and refined by introducing Tree-Augmented Naive Bayes networks (TANs) [18] to allow inter-concept dependencies to be robustly modelled.

- The two semantic analyzer components, the HVS semantic parser and the modified-TAN dialogue act decoder, have been integrated with a standard HTK-based [19] Hidden Markov Model (HMM) speech recognizer and the additional knowledge provided by the semantic analyzer has been used to determine the best-scoring word hypothesis from N-best list generated by the speech recognizer. By doing so, a purely data-driven spoken language understanding (SLU) system has been built but with greatly reduced build cost compared to the existing SLU systems. Moreover, robustness issues have been addressed and the results support the claim that an SLU

system which is statistically-based and trained entirely from data is intrinsically robust and can be readily adapted to new applications.

1.5 Data Corpora

The thesis work reported here focuses on statistical approaches which require a large number of spoken utterances as training data. Among various well-established spoken language databases, the ATIS [20] database has been selected for the main experimental development, primarily because it has been used by DARPA for Spoken Language Understanding (SLU) evaluations in the Human Language Technology Workshops. Hence it will be possible to compare the results with those of the DARPA participants. The sentences in the ATIS corpus have been divided into three categories, context-independent (A), context-dependent (D), or unanswerable (X). The research here will mainly focus on category A sentences. Besides the ATIS corpus, the more recent and more complex DARPA Communicator task [21] which contains hotel reservation, car rental in addition to flight information has also been used for natural language understanding evaluation. More detailed descriptions of both the ATIS and DARPA Communicator corpora may be found in Section 3.5.2.

1.6 Thesis Overview

This chapter has briefly described semantic processing in spoken dialogue systems and the limitations of existing systems which motivate the research on pursuing automatic learning techniques to perform spoken language understanding. The rest of the thesis is organized as follows.

Chapter 2 presents a literature survey of the work conducted in the relevant areas, which includes semantic parsing, dialogue act decoding, and adaptation of statistical models. The advantages and disadvantages of the various techniques are compared.

Chapter 3 discusses the Hidden Vector State (HVS) model in detail and presents the mathematical framework. It also explains how the HVS model can be trained from minimally annotated data. An experimental evaluation using both the ATIS and the DARPA Communicator Travel tasks is presented.

Chapter 4 discusses dialogue act detection using Tree-Augmented Naive Bayes (TAN) networks. Training of TAN networks is first described followed by network inference using the Probability Propagation in Trees of Clusters (PPTC) algorithm. The system performance is compared with the baseline systems using Naive Bayes and neural networks.

Chapter 5 described an integrated spoken language understanding (SLU) system which consists of three major components, a speech recognizer, a HVS model-based semantic parser, and a TAN-based dialogue act decoder. The overall end-to-end spoken language understanding evaluation results on the ATIS corpus are reported. In addition, robustness issues of the SLU system have also been addressed, where two aspects of the system performance have been investigated: noise robustness and adaptability to different applications.

Finally, Chapter 6 summarizes the work reported throughout this thesis and identifies several potential areas for future research.

Chapter 2

Related Work

This chapter gives a survey of existing research work presented in the literature on semantic processing, with emphasis on statistical approaches. Section 2.1 describes general requirements and design of statistical models for semantic analysis. Various techniques used in the two major stages of semantic analysis, namely, semantic parsing and dialogue act decoding, are then discussed in Section 2.2 and Section 2.3 respectively. As semantic parser adaptation has also been investigated in this research, some existing adaptation techniques on parser and language modelling are presented in Section 2.4.

2.1 Semantic Processing

This section gives the definition of semantic processing upon which this research framework is based. It also describes how semantic processes can be modelled using statistical approaches.

2.1.1 General Requirements

In the simple task oriented spoken dialogue systems which form the focus of this work, each user utterance is converted to a dialogue act containing a set of attribute-value pairs such as

```
REQUEST=flights, FROMLOC=Boston, TOLOC=Denver, etc
```

The goal of the semantic processing component is to annotate each natural language input so as to allow these attribute-value pairs to be extracted by a simple deterministic task-independent process. In the work reported here, such extracted semantic knowledge are stored in structures called *frames* which consist of a name of an entity and relations represented by attribute-value pairs called *slots*. For the above example, its corresponding frame structure is:

```
Frame: flights
```

```
Slots: FROMLOC=Boston, TOLOC=Denver, etc
```

A particularly simple form of annotation results from mapping each word w in an utterance W into a single discrete concept c. For example, the utterance

```
List flights from Boston to Denver
```

might be encoded as

```
DUMMY(List) FLIGHT(flights) FROMLOC(from) CITY(Boston)
TOLOC(to) CITY(Denver)
```

where `FLIGHT`, `FROMLOC`, and `TOLOC` are semantic concepts(tags) and irrelevant input words such as "list" are mapped into the `DUMMY` tag and subsequently discarded. This is the flat concept model. Although it works surprisingly well on simple tasks, one obvious drawback of the model is that it cannot directly represent hierarchical dominance relationships and hence in the example above, the role of Boston can only be inferred from its proximity to `FROMLOC`. The mapping of the annotation to the attribute-value pair `FROMLOC=Boston` requires heuristics which quickly break in the face of more complex nested sentence structures.

A more complex encoding will preserve the hierarchical structure of the sentence as in

```
DUMMY(List)
FLIGHT(flights FROMLOC(from CITY(Boston)) TOLOC(to CITY(Denver)))
```

In this case, the unambiguous extraction of the attribute-value pairs is now relatively straightforward but at the cost of considerable additional complexity in the parser.

Given a set of semantic concepts and some pre-defined dialogue act grammar rules, the dialogue acts can be detected by selecting the frame (or frames) which provides the best match. Alternatively, multiple dialogue acts may be inferred using Bayesian networks or similar classification techniques to provide a soft decision. Dialogue act detection may also be viewed as an information retrieval (IR) or call routing problem and the same algorithms used for IR and call routing can be applied to dialogue act decoding. Existing techniques for dialogue act detection will be discussed in Section 2.3 in detail.

It has to be mentioned that from the viewpoint of researchers in computational linguistics, dialogue acts [22] or conversational moves [23; 24] are used to model conversational functions that an utterance can play. A dialogue act tagging scheme, Dialogue Act Markup in Several Layers (DAMSL) [24; 25], has been developed to code various levels of dialogue information about utterances. However, in this research, dialogue acts are analogues to user's intentions or information goals of an utterance.

2.1.2 Statistical Models

The goal of semantic analysis is to convert meanings of a user's utterance into semantic representations and infer the user's intentions or dialogue acts according to the extracted meaning and existential evidence. Semantic analysis can be viewed as the problem of finding the maximum likely user generated dialogue acts A_u given a word sequence W and the current system belief about the dialogue state B_s, i.e.

$$\hat{A}_u = \underset{A_u}{\operatorname{argmax}}\ P(A_u|W, B_s) \tag{2.1}$$

Assume the word sequence W is mapped into a semantic concept sequence C, then

$$\begin{aligned} \hat{A}_u &= \underset{A_u}{\operatorname{argmax}}\ \{\sum_C P(A_u, C|W, B_s)\} \\ &= \underset{A_u}{\operatorname{argmax}}\ \{\sum_C P(A_u|C, B_s)P(C|W, B_s)\} \\ &\approx \underset{A_u}{\operatorname{argmax}}\ \{\underset{C}{\operatorname{argmax}}\ \{P(A_u|C, B_s)P(C|W, B_s)\}\} \end{aligned} \tag{2.2}$$

Equation 2.2 can be decomposed into two stages, first, solving

$$\hat{C} = \underset{C}{\operatorname{argmax}}\ P(C|W, B_s) \tag{2.3}$$

and then

$$\hat{A}_u = \underset{A_u}{\operatorname{argmax}}\ P(A_u|\hat{C}, B_s) \tag{2.4}$$

Equation 2.3 represents semantic parser process which can be implemented using Hidden Markov Models (HMM) or stochastic context-free grammar models as will be discussed in Section 2.2. Equation 2.4 represents the process of finding the most likely user dialogue act based on the decoded semantic concept sequence and the system's belief of the current dialogue state. Common techniques used for this process include Bayesian Networks, vector-based methods such as latent semantic analysis and neural networks, call routing methods etc as will be discussed in Section 2.3.

The following sections will give a survey of the relevant existing work on semantic parsing and dialogue act decoding, with a focus on statistical methods.

2.2 Semantic Parsing

In a spoken dialogue system, semantic parsing translates word sequence hypotheses generated by a speech recognizer into semantic representations. Traditionally, most semantic parser systems have been built using hand-crafted semantic grammar rules. Word patterns corresponding to semantic tokens are used to fill slots in different semantic frames

in parallel. The frame with the highest score yields the semantic representation. The above process is called *robust parsing* [26; 27; 28]. Other systems view semantic parsing as a pattern recognition problem and adopt statistical approaches to derive semantic representations. In this chapter, only systems using stochastic parsing methods will be reviewed.

2.2.1 Finite-State Semantic Models

A *finite-state tagger (FST)* treats the generation of a spoken sentence as an HMM-like process whose hidden states correspond to semantic concepts and outputs correspond to individual words in the utterance. Figure 2.1 shows an example of a parse C output using an FST model. Each word is tagged with a single discrete semantic concept label. Note here in practice, the utterance is enclosed by sentence start and end markers, `sent_start` and `sent_end`, to enable the prediction of the first and last word in the utterance and the corresponding semantic tags for them are `SS` and `SE` respectively. They are omitted here for clarity.

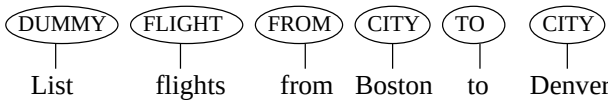


Figure 2.1: An example of a finite-state tagger parse result.

Note that if required, the output of the FST can be trivially converted to a well-formed parse tree simply by adding a sentence tag dominating all FST state tags. This parse tree will however be flat, hence this corresponds to the flat-concept model referred to previously.

Given a vocabulary $\mathcal{V}$ and a word sequence $W = (w_1 \cdots w_T)$ where $w_t \in \mathcal{V}$, assume that each word w_t is tagged with one semantic concept label c_t, to give the sequence $C = (c_1 \cdots c_T)$ where $c_t \in \mathcal{L}$ and $\mathcal{L}$ denotes the semantic concept or label space. The FST model can then finds the maximum likely concept string C, given word string W according to

$$\begin{aligned}
\hat{C} &= \underset{C}{\text{argmax}}\ P(C|W, B_s) \\
&= \underset{C}{\text{argmax}}\ P(W|C)P(C|B_s) \\
&\approx \underset{C}{\text{argmax}} \prod_{t=1}^{T} P(w_t|w_{t-1} \cdots w_1, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \cdots c_1, B_s) \\
&\approx \underset{C}{\text{argmax}} \prod_{t=1}^{T} P(w_t|w_{t-1} \cdots w_{t-n+1}, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \cdots c_{t-m+1}, B_s) \qquad (2.5)
\end{aligned}$$

The $P(W|C)$ term is called the *syntactic* or *lexical model* while the $P(C|B_s)$ term is called the *conceptual* or *semantic model*. Both the lexical and semantic models can be formulated using conditional probabilities. If $n = 1$ and $m = 2$, this becomes a conventional first order Markov model, with states labelled by semantic concepts and transitions given by the concept bigram probabilities:

$$\hat{C} = \underset{C}{\text{argmax}} \prod_{t=1}^{T} P(w_t|c_t)P(c_t|c_{t-1}, B_s) \tag{2.6}$$

One of the most notable FST models is CHRONUS [29; 30; 31; 32], which is a statistical spoken language understanding system developed by AT&T. Its FST is essentially a first order Markov model as described in equation 2.6.

In AT&T's implementation, words and phrases are first classified into categories in order to reduce the effective size of the lexicon and enhance the robustness of the stochastic conceptual representation [29]. For example, Boston and Denver are grouped under the category city_name.

After word/phrase categorization, a set of fully annotated training data must be generated. The ATIS corpus does not contain any annotated data, instead, the SQL query for each training utterance is included. Bootstrapping is therefore used to gather annotated training data. In AT&T's work, 547 sentences were manually tagged, and these were then used to calculate estimates of the model parameters based on relative frequency counts. This initial model was used to annotate the remaining sentences in the corpus. Each newly annotated sentence was processed to generate an SQL query to fetch results from the database, which were then compared with the reference answers. If the results matched, the corresponding annotated sentence was assumed correct and was therefore added to the training set. The whole process was then iterated until no further addition could be made to the training set.

Given the availability of the fully-annotated training data, the model parameters can be simply estimated from the frequency counts and smoothing. The maximum likelihood estimates are

$$\hat{P}_{ML}(c_t|c_{t-1}) = \frac{N(c_{t-1}, c_t)}{N(c_{t-1})} \tag{2.7}$$

$$\hat{P}_{ML}(w_t|c_t) = \frac{N(w_t, c_t)}{N(c_t)} \tag{2.8}$$

where $N(\cdot)$ denotes an event count. In practice, discounting is used to replace the original counts with modified counts so as to redistribute the probability mass from the more commonly observed events to the less frequent and unseen events. If the actual number of occurrences of an event E is $N(E)$, then the modified count is $d_{N(E)}N(E)$, where $d_{N(E)}$ is

known as the discount factor. One discounting strategy is the Witten-Bell method where the discounting factor is not dependent on the event's count, but on r, the number of types which followed the particular context. By applying Witten-Bell discounting and denoting the size of the semantic concept space as M, Equation 2.7 becomes

$$\hat{P}_{ML}(c_t|c_{t-1}) = \begin{cases} \frac{N(c_{t-1},c_t)}{M+r} & \text{if } N(c_{t-1},c_t) > 0 \\ \frac{r}{M+r} & \text{if } N(c_{t-1},c_t) = 0 \end{cases} \tag{2.9}$$

The discounting formula for Equation 2.8 can be derived in a similar manner. Other commonly used discounting methods include Good-Turing, linear discounting, nonlinear discounting, etc [33].

The trained semantic decoder then finds the most likely state, and hence concept, sequence using standard Viterbi decoding.

2.2.2 Stochastic Context-Free Grammar Models

The finite-state models described in the previous section are not able to capture long distance semantic relevance and therefore fail to represent nested semantic structures. Such limitations can be overcome by allowing the finite state networks to become recursive. In this case, each state can generate either a word or a subnetwork. This is formally equivalent to using stochastic phrase structure rules and it essentially extends the class of supported languages from *regular* to *context-free*. Examples of such *Stochastic Context-Free Grammar(SCFG)* models are non-lexicalized SCFG models such as BBN's Hidden Understanding Model (HUM) [34; 35] and hierarchical HMMs (HHMM) [36; 37], lexicalized models such as the immediate-head parsing model [38], history-based models such as the structured language model (SLM) [39] and the semantic structured language model [40].

2.2.2.1 Non-Lexicalized SCFG Models

Non-lexicalized SCFG models represent a parse tree as n events, $Event_1 \cdots Event_n$, where $Event_i$ corresponds to the ith context-free grammar rule. Two major weaknesses of such non-lexicalized SCFG models are lack of sensitivity to lexical dependencies and lack of sensitivity to structural preferences such as preferences for right-branching or left-branching structures. Examples of non-lexicalized SCFG models are BBN's Hidden Understanding Model (HUM) [34; 35] and hierarchical HMMs (HHMM) [36; 37].

Hidden Understanding Model (HUM)

The Hidden Understanding Model (HUM) [34; 35; 41; 42] developed by BBN also used HMMs to provide a probabilistic framework for spoken utterances. However, unlike

CHRONUS, the annotation of each sentence is represented as a semantic tree, which consists of non-terminal and terminal conceptual nodes. Each terminal conceptual node is attached to individual words of an utterance, while a non-terminal node is designated as an abstract semantic concept such that a parse tree can be built automatically through the state evolution of an HMM. The idea of HUM is to automatically discover complex Context-Free Grammar (CFG) rules based on treebank data. An example of an HUM parse result is shown in Figure 2.2.

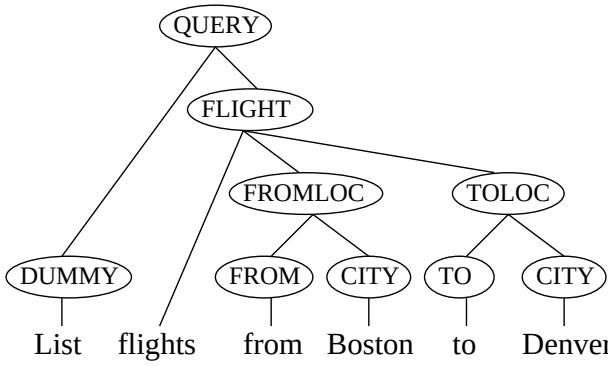


Figure 2.2: An example of an HUM parse result.

Given a word sequence W, the meaning M is searched by maximizing:

$$P(M|W) = \frac{P(W|M)P(M)}{P(W)} \approx P(W|M)P(M) \tag{2.10}$$

where $P(M)$ is called the *semantic language model* and $P(W|M)$ is called the *lexical realization model*.

In the *semantic language model*, each abstract concept corresponds to a probabilistic state transition network, with each constituent state denoting one of its component concepts. Every component concept state is fully connected to every other state. All such networks are then combined into a single probabilistic recursive transition network (PRTN) through their unique entry and exit states. Individual component networks of the PRTN are analogous to the right hand sides of production rules in a conventional CFG. The *lexical realization model* is similar to the semantic language model except that the lexical realization networks specify transition probabilities between words instead of semantic concepts. Since the semantic language model and lexical realization model are both probabilistic networks, the understanding problem can be formulated as finding the highest probability path through the combined network, therefore:

$$P(Path) = \prod_{t \in Path} \begin{bmatrix} P(state_n|state_{n-1}, context) & \text{if t in semantic language model} \\ P(word_n|word_{n-1}, context) & \text{if t in lexical realization model} \end{bmatrix} \tag{2.11}$$

In the semantic language model, *context* refers to the parent concepts for $state_n$ and $state_{n-1}$, while in the lexical realization model, *context* is the immediately dominating semantic concept of $word_n$. Thus, in Figure 2.2, $P(CITY|FROM, FROMLOC)$ refers the probability of entering CITY from FROM within the context of the FROMLOC network. $P(boston|from, CITY)$ is the probability that the word boston follows the word from within the context of a FROM concept.

As in CHRONUS, HUM also uses hand labelled data to estimate the model parameters, which are subsequently smoothed using a technique similar to [43; 44]. A modified version of the Early Parsing Algorithm [45] is used to search for the N-most-likely meaning trees in a top-down fashion. It was reported that HUM achieved 16.5% answer error rate for the December 1994 ATIS spoken language understanding evaluation [34] compared to 8.9% achieved by CHRONUS [32]. In the December 1994 ATIS natural language evaluation, HUM obtained 9.5% answer error rate [35] which is also worse than 3.8% obtained by CHRONUS [32] for context-independent (category A) utterances. One possible reason is the lack of sufficient training data to reliably train a complicated recursive parsing model such as the HUM.

Hierarchical Hidden Markov Models (HHMM)

A *hierarchical hidden Markov model (HHMM)* [36] generalizes the normal HMM by making each of its hidden states a similar stochastic model on its own, i.e. each state is an HHMM as well. States are therefore categorized into two classes, states which emit output symbols through the usual HMM state output mechanism belong to *production states*, and states that output sequences rather than a single observable symbol are called *internal states*. The production of sequences is performed by recursive activation of the sub-models of a internal state, which may be composed of sub-sub-models as well. This process terminates when a production state is reached.

The activation of a lower sub-model is called a *vertical transition*. Upon completion of a vertical transition, the control returns to the original state and a state transition within the same level, called *horizontal transition*, is performed.

HHMMs are less expressive than SCFGs since they only allow hierarchies of bounded depth, so they are more efficient and relatively easier to learn. However, the training procedure described in [36], which is based upon a straightforward adaptation of the Inside-Outside algorithm [46], is still complicated and takes $O(T^3Q^D)$ time, where T is the length of the sequence, D is the depth of the hierarchy, and Q is the maximum number of states at each level of the hierarchy. Murphy [37] proposed to represent an HHMM as a dynamic Bayesian network (DBN) and derived a much simpler approximate inference

algorithm, which takes at most $O(TQ^{2D})$ time. However, the empirical consequences of the constraints implied are unknown.

2.2.2.2 Lexicalized Models

The major weakness of non-lexicalized SCFG models is their lack of sensitivity to lexical information. This can be avoided by introducing lexicalization of non-terminals of parse trees. Each non-terminal in the parse tree is modified to include a *headword* which is assigned based on a function that identifies the *head* of each grammar rule. The *head* is one of the child non-terminals in the rule. More precisely, the function $head(X \rightarrow Y_1 \cdots Y_n)$ returns a value $1 < h < n$. For example, in the rule PP $\rightarrow$ IN NN in Figure 2.3 (above the words "from boston"), the IN is the head of the phrase and $head(PP \rightarrow IN\ NN) = 1$. In this case, $headword(PP) = headword(IN) = from$.

Example of lexicalized SCFG models may be found in [38; 47; 48; 49]. Here, only the recently proposed immediate-head parsing model in [38] is reviewed. An example of an immediate-head parse result is given in Figure 2.3.

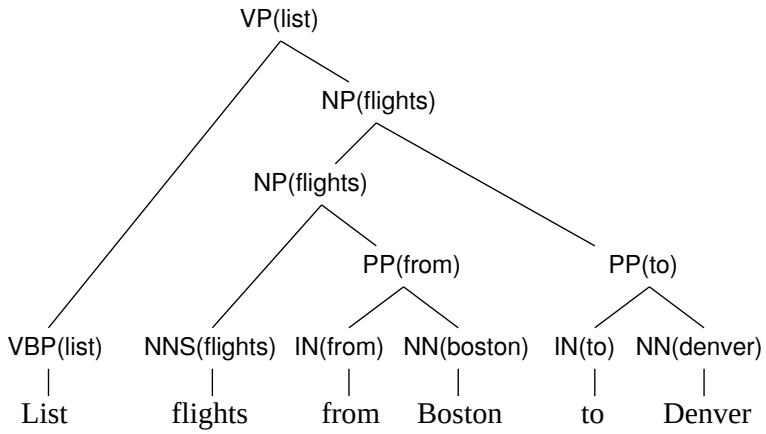


Figure 2.3: An example of an immediate-head parse result.

Charniak's immediate-head parsing model performs parsing in a top-down manner. It assigns a probability to a parse π by a top-down process of considering each constituent c in π and, for each c, first guessing the pre-terminal part-of-speech (POS) tag of c, $t(c)$, then the lexical head of c, $h(c)$, and then the expansion of c into further constituents $e(c)$. Thus the probability of a parse is given by

$$P(\pi) = \prod_{c \in \pi} P(t(c)|l(c), H(c)) \cdot P(h(c)|t(c), l(c), H(c)) \cdot P(e(c)|l(c), t(c), h(c), H(c)) \tag{2.12}$$

where $l(c)$ is the label of c and $H(c)$ is the relevant history of c which may be defined as consisting of the label, head, and head part-of-speech for the parent of c.

Charniak's model encodes preferences for right-branching structure implicitly through the use of higher probabilities on right-branching grammar rules. The parser encourages right branching with a "bonus" multiplicative factor of 1.2 for constituents that end at the right boundary of the sentence, and a penalty of 0.8 for those that do not. For example, for an NP PP PP sequence, the perference for a right-branching structure is encoded through a much higher probability for the rule NP $\rightarrow$ NP PP rather than NP $\rightarrow$ NP PP PP as illustrated in Figure 2.3 the top-level POStags for flights from Boston to Denver is [NP [[NP PP] PP]].

This immediate-head parsing model differs from strict left-to-right parsers, which will be discussed in the following section, in that it is able to take into consideration the immediate head of a constituent that occurs after this constituent which is not available for conditioning by a strict left-to-right parser. However, by doing so, the immediate-head parsing model complicates left-to-right search since heads are often to the right of their children. Also, unlike strict left-to-right parsers which can be easily integrated with the standard trigram language model, it is not straightforward to do this for the immediate-head parsing model.

2.2.2.3 History-Based Models

In history-based models, the probability of each parser action is conditioned on the history of previous actions in the parse, i.e. potentially all previously built structure, not just the non-terminal being expanded as in simple SCFGs. Assume a word sequence W is parsed to give a tree C, history-based models map (W, C) into a sequence of decisions $< d_1, d_2 \cdots d_T >$, the joint probability of W and C is then given by

$$P(W, C) = \prod_{t=1}^{T} P(d_t | \Phi(d_{t-1} \cdots d_1)) \tag{2.13}$$

The conditioning context for each d_t, $< d_{t-1} \cdots d_1 >$, is referred to as "history", and is equivalent to some partially built structure. Φ is a function to group histories into equivalence classes. A normal SCFG model may be viewed as a history-based model using leftmost derivations.

Examples of history-based models include Chelba's structured language model (SLM) and Erdogan's semantic structured language model, which are different from Charniak's work in that non-terminal labels are extended to include both lexical items (headwords) and semantic categories.

Structured Language Model

Chelba *et al.* proposed the use of a structured language model (SLM) [50] as a statistical parser to extract essential information from a sentence to fill slots in a semantic frame. Here, a sentence is parsed as a two-level semantic parse tree: the root node is tagged with a semantic frame label and spans the entire sentence; the leaf nodes are tagged with slot labels and span the strings of words relevant to the corresponding slot.

The SLM was originally defined as a strict left-to-right, right-branching syntactic parser [39; 51; 52] and it assigns a joint probability $P(W,T)$ to a sentence-tree pair. His work is quite similar to that discussed in [47] where non-terminal labels are extended to include lexical items (head words) or semantic categories. In order to use SLM in semantic parsing, some constraint operations have to be defined. However, before describing constrained SLM parsing, the standard formulation of SLM will be discussed first.

The SLM model operates on binary-branching trees. Three kinds of operations are defined. WORD-PREDICTOR predicts the next word based on the two previous head-words exposed in the parse. TAGGER assigns a POS tag to the currently predicted word. And finally PARSER gradually joins up the subtrees. There are three possible moves in the PARSER component: join the right-most adjacent two trees with the new head word provided by the left tree (adjoint-left), or with the new head word provided by the right tree (adjoint-right), or to choose not to join the substrees (null), but pass the control to the WORD-PREDICTOR component. This model provides trigram probabilities conditioned on head words that may fall outside the three-word window of a traditional trigram model, thus it successfully captures long distance relevance in a sentence. A parse example using the SLM is shown in Figure 2.4, which is different from the result shown in Figure 2.3 by Charniak's immediate-head parsing model as NN(Boston) is adjoined to the right PP phrase first to form a NP phrase before adjoining to the left IN(from).

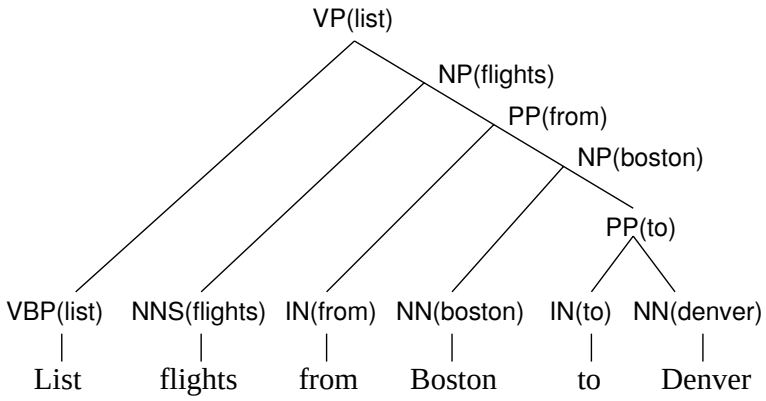


Figure 2.4: An example of SLM parse result.

Let W be a sentence of length n words to which sentence markers are added so that $w_0 =$<s> and $w_{n+1} =$</s>, the SLM assigns a probability $P(W,T)$ to the sentence W and its every possible binary parse T using the three models:

$$P(W,T) \approx \prod_{k=1}^{n+1} [\underbrace{P(w_k|h_0,h_{-1})}_{predictor} \underbrace{P(t_k|w_k,h_0,h_{-1})}_{tagger} \underbrace{\prod_{i=1}^{N_k} P(p_i^k|h_0,h_{-1})]}_{parser} \tag{2.14}$$

where:

- w_k is the word predicted by the WORD-PREDICTOR;
- h_0 and h_{-1} are the exposed heads, each head is a pair (headword, non-terminal label), or (word, POStag) in the case of root-only tree;
- t_k is the POS tag assigned to w_k by the TAGGER;
- $N_k - 1$ is the number of operations that the PARSER executes at sentence position k before passing control to the WORD-PREDICTOR (the N_k-th operation at position k is the null transition);
- p_i^k denotes the i-th PARSER operation carried out at position k in the word string. The $p_1^k ... p_{N_k}^k$ sequence of PARSER operations at position k grows the word-parse $(k-1)$-prefix into a word-parse k-prefix.

Training data is divided into two parts, about 90% forms the "main" data and the remaining forms the "check data". Each model component is initialized in two stages. First counts from the "main" data are gathered, then the interpolation coefficients are estimated using counts gathered from the "check data". An N-best EM [53] variant is then employed to jointly estimate the model parameters such that the likelihood of the training data is increased.

Returning now to the constrained parsing needed for semantic extraction, the binary parses that generated by the standard SLM may not match the semantic parse S for a given sentence W which consists of a set of constituent boundaries along with semantic tags, i.e. the constituent proposed by the SLM may cross the semantic constituent boundaries. Thus, each semantic constituent in S is considered as a constaint and is defined such that $c =< l, r, Q >$ where l and r are the left and right boundary of the constraint respectively, Q is the set of allowable non-terminal tags for the constraint. During training, the SLM operation is constrained such that the parses must match the constraints c_i, $i = 1...C$ as it proceeds left to right through a given sentence W. In summary, the training procedure for using the SLM as a semantic parser is:

- *Initialization.* Initialize the SLM as a syntactic parser from a treebank.
- *Syntactic parsing.* Train the SLM as a matched constrained parser such that parses match the constraint boundaries $c.l$, $c.r$, $\forall c$ for a given sentence.
- *Augmentation.* Enrich the non/pre-terminal labels in the resulting treebank with semantic tags.
- *Syntactic+semantic parsing.* Train the SLM as an L(abel)-matched constrained parser, i.e., boundaries and tags of the semantic constituents are matched which include $c.l$, $c.r$, $c.Q$, $\forall c$.

If $SEM(T)$ is defined as the function that maps a parse tree T containing both syntactic and semantic information to the tree containing only semantic information, referred to as the *semantic projection* of T, then the trained model can be used to retrieve the semantic parse by taking the semantic projection of the most likely parse:

$$S = SEM(\underset{T_i}{\text{argmax}}\ P(T_i, W)) \tag{2.15}$$

Semantic Structured Language Model

Inspired by Chelba, Erdogan proposed a semantic structured language model [40] which makes use of the IBM's decision-tree based statistical semantic classer and parser [54]. Semantic analysis is performed in two steps, first words that represent the same concept are grouped using a *semantic classer*, then a full hierarchical parse of semantic constituents is built using a *semantic parser*. The parser predicts parser actions (tagging a word, extending a parse by connecting constituents to parent labels, etc) based on the current context using a decision tree [55]. Both classer and parser require a fully annotated training corpus of in-domain sentences to train decision tree probabilities.

The semantic information output by the parser described above is then incorporated with the lexical information in a maximum entropy model in the following way:

$$P(o|h) = \frac{e^{\sum_i \lambda_i f_i(o,h)}}{\sum_{o\prime} e^{\sum_i \lambda_i f_i(o\prime,h)}} \tag{2.16}$$

where o is the outcome and $o \in \mathcal{V} \cup \mathcal{L}$, that is, o can be taken from word (or class) vocabulary $\mathcal{V}$ or semantic label vocabulary $\mathcal{L}$, h represents the history of context and f_i are indicator functions, or *features*, which are "activated" when certain outcomes o_i occur in certain context where $q_i(h) = 1$. Here, $q_i(h)$ is also an indicator function that

is activated only when the context h has certain property. In particular, $f_i(o,h)$ has the form:

$$f_i(o,h) = \begin{cases} 1 & \text{if } o = o_i \text{ and } q_i(h) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

Features such as parent label name, previously completed constituent label names as well as regular N-gram history features for both words and labels are combined in a single effective language model using maximum entropy modelling approach.

Unlike Chelba's work in which the parse tree is built bottom-up and the parent label is not available to the decoder when the current word is being decoded, Erdogan's SLM assumes the availability of complete sentences and is used to rescore N-best word hypotheses from a speech recognizer.

Experiments on the subset of the DARPA Communicator evaluation data in June 2000 show that the gains obtained in the word error rate is similar to those achieved with the SLM in other domains [56].

2.2.3 Discussion

Semantic parsing can be viewed as a pattern recognition problem and statistical decoding can be used to find the most likely semantic representation given a model and an observed word sequence $W = (w_1 \cdots w_T)$. If assuming that the hidden data take the form of a semantic parse tree C then the model should be a push-down automata which can generate the pair $< W, C >$ through some canonical sequence of moves $D = (d_1 \cdots d_T)$. That is,

$$P(W,C) = \prod_{t=1}^{T} P(d_t | d_{t-1} \cdots d_1) \tag{2.18}$$

Decision sequences are usually steps in some top-down or bottom-up derivation of trees. For the general case of an unconstrained hierarchical model, D will consist of three types of probabilistic move:

1. popping semantic category labels off the stack;
2. pushing one or more non-terminal semantic category label onto the stack;
3. generating the next word.

In practice, conditional independence can be used to reduce the number of parameters needed to manageable proportions. As in conventional statistical language modelling, this

involves defining an equivalence function Φ which groups move sequences into equivalent classes. Thus, the final generic parsing model is:

$$P(W, C) = \prod_{t=1}^{T} P(d_t | \Phi(d_{t-1} \cdots d_1)) \tag{2.19}$$

The above is essentially the history-based model described in Section 2.2.2.3 where the probability of each parser action is conditioned on the history of previous actions in the parse or some partially built structure.

For the constrained case of the flat concept model, the stack is effectively depth one and $< W, C >$ is built by repeatedly popping one label off the stack, pushing one new label onto the stack and then generating the next word. This kind of model is unable to capture long distance dependencies. This inability of representing hierarchical structures can be overcome by allowing the state stack to grow without limit and more than one new semantic labels to be pushed onto the stack. This is essentially analogous to using stochastic phrase structure rules and extends the class of supported languages from *regular* to *context-free*. The HUM model is an early example of such an SCFG model which uses fully-annotated corpora to simplify the parameter estimation problem that is otherwise complex due to the recursive nature of hierarchical parse trees.

A general SCFG model is computationally expensive to train. However, computational tractability issues may be tackled by imposing certain constraints on the SCFG model itself. The HHMM model constrains the level of hierarchies or the state stack depth to be a bounded depth, it is nevertheless still complex as its state inference takes $O(T^3)$ time where T is the sequence length. Murphy converts an HHMM into a dynamic Bayesian networks such that the inference can be done using the junction tree algorithm which only takes $O(T)$ time empirically provided that the HHMM hierarchy depth and the number of states at each level of hierarchy are bounded to some relatively small values.

The weakness of non-lexicalized SCFG models such as HUM and HHMM can be avoided by associating a headword to each non-terminal in the parse tree. Examples of lexicalized SCFG models such as the immediate-head parsing model achieved 6% reduction in recall error and 5% reduction in precision error compared to a general non-lexicalized model when tested on Penn WSJ treebank data [57].

Chelba's SLM does not impose a constraint on the state stack depth, but it does constrain the pushing of at most one new tag (a POStag in this case, not a semantic tag) into the stack. As opposed to the conventional SCFG models where each parser action is only conditioned on the immediately preceeding non-terminal tag being expanded, a parser action in the SLM is conditioned on the previouly two exposed headwords. However,

Chelba's SLM has the limitation that it is not able to capture dependencies between non-headwords due to its headword percolation rules. For example, *less* and *than* as in *less people join the society this month than last month* where neither *less* nor *than* are headwords of this phrase.

Erdogan's structured semantic language model is similar to Chelba's SLM, but it assumes the availability of complete sentences and cannot operate left-to-right. Thus, unlike the SLM, it cannot be incorporated directly into the speech recognition process. Nevertheless, it can still be used to rescore N-best word hypotheses from a speech recognizer.

All of the previously reviewed hierarchical structured models require full-annotated treebank style training data for robust model parameter estimation. This is both time-consuming and difficult to acquire in practice. Robust training from unannotated or partially annotated corpora is therefore crucial for practical applications. This is one of the major issues that has been explored in the research work reported here.

All the parser models described so far apply constraints in one way or another to the general framework described in Equation 2.19 and other forms of constraint are equally possible. In particular, this thesis considers a constrained form of automata where the stack is finite depth and $< W, C >$ is built by repeatedly popping 0 to n labels off the stack, pushing exactly one new label onto the stack and then generating the next word. This constrained form of automata implements a right-branching parser which has some very convenient properties. It is left-to-right, it can be trained robustly using EM, and it has complexity $O(TQ^D)$ [1], yet it can still model hierarchical structure. The investigation of this model comprises a major part of this thesis.

2.3 Dialogue Act Decoding

After transforming a user's utterance into a semantic representation, the next step in a spoken language understanding system is to decode the user's dialogue acts based on the semantic concepts extracted and the current system's belief. Available approaches to this problem include Bayesian Networks (BNs) [58; 59] which take semantic concepts as inputs. Other methods operate directly on words instead of semantic concepts though the extension to semantic concepts is straightforward, these include vector-based methods [60; 61], call routing methods [62; 63; 64; 65], Transformation-Based Learning (TBL) [66; 67], and topic trees [68]

[1] Where T is the length of the sequence, D is the maximum stack depth, and Q is the maximum number of concepts (node labels) at each level of the stack.

2.3.1 Bayesian Networks

Bayesian Networks (BNs) [58; 59] take the topological form of a directed acyclic graph (CAG), where each link is directional, and there are no loops. A BN consists of nodes and their associated discrete states. The links between nodes represent conditional probabilities. BNs capture the conditional independence relationship in the sense that a node is independent of its ancestors given its parents. The advantage of using BNs is that they only specify necessary dependencies, which leads to a significant reduction in the cost of inference. The main disadvantage of BNs is that, in general, the complexity rises exponentially with the number of nodes [69]. Besides, creating algorithms to update probabilities in BNs is not trivial though it is now quite well understood.

BNs have been first applied for dialogue act classification by Pulman [70] and have also been used to infer information goals from users' free-text queries in information retrieval [71]. More recently, Meng *et. al* [58] proposed the modified Naive Bayes networks to infer users' dialogue acts or information goals in spoken dialogue applications and Keizer [59] used more general BNs to perform the same tasks for Dutch dialogues.

2.3.1.1 Naive Bayes

A method using modified Naive Bayes networks has been proposed [58; 72; 73; 74; 75] to identify the communication goal(s) of a user's information-seeking query out of a finite set of within-domain goals in spoken dialogue systems. The problem is formulated as N binary decisions, and each is performed by a BN. An example of a BN is shown in Figure 2.5. The arc from concept node 2 to node 3 captures the probabilistic dependency between two nodes.

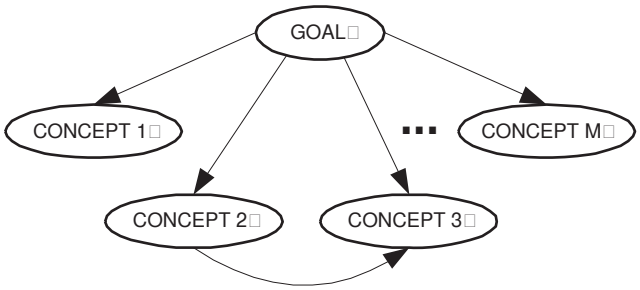


Figure 2.5: An example of a modified Naive Bayes Network.

A user's input query is first transformed into a sequence of semantic concepts by a semantic parser. These concepts serve as the input to the BNs for inferring the query's communication goal. The process of goal identification is as follows:

1. *Semantic tagging.*

   Training utterances are tagged with semantic concepts. Each training query is then represented by its annotated goal and a set of semantic concepts.

2. *Belief Network Training*

   (a) Semantic concept selection

   For a given goal G_i, only M semantic concepts $\{C_1, C_2, \cdots, C_M\}$ which have the highest Mutual Information (MI) and Information Gain (IG) are selected as the feature set for the ith goal. These form the input to the BN. MI (2.20) measures the degree of co-occurrence while IG (2.21) considers both the presence and absence of the concept and the goal [1].

$$MI(C_k, G_i) = P(C_k, G_i) \log \frac{P(C_k, G_i)}{P(C_k)P(G_i)} \tag{2.20}$$

$$IG(C_k, G_i) = \sum_{c=0,1} \sum_{g=0,1} P(C_k = c, G_i = g) \log \frac{P(C_k = c, G_i = g)}{P(C_k = c)P(G_i = g)} \tag{2.21}$$

   (b) Bayesian network learning

   For each BN, the goal prior probability $P(G_i)$ and the conditional probability of each semantic concept C_k given the goal G_i, $P(C_k|G_i)$, are learned by presenting each training query to the BN and using simple event counting. Dependencies between concept nodes are captured using Rissanen's minimum description length (MDL) principle [76].

3. *Goal identification.*

   Each trained BN is a distinct classifier for making the binary decision using a preset threshold 0.5 regarding a unique goal. For an input query which contains a set of concepts $C_1, C_2, \cdots, C_M$, a BN applies Bayesian inferencing (2.22) to derive the *a posteriori* probability $P(G_i|C_1, C_2, \cdots, C_M)$.

$$P^*(G_i = 1|C_1, C_2, \cdots, C_M) = P(G_i = 1) \prod_{k=1}^{M} \frac{P(C_k = c_k|G_i = 1)}{P(C_k = c_k)} \tag{2.22}$$

   where c_k can either be 1 or 0 depending on whether the kth concept should be present or absent in the network. The decision across all the BNs are combined to identify the output goal of the input query. The query can be labelled with

[1] In the equations decribed in this section, each concept variable C_k takes the value either 1 or 0, which is different from what has been described previouly where C_k takes certain semantic concept as a value. Thus, $P(C_k = 1)$ denotes the probability of the presence of kth concept as in Figure 2.5.

a goal if the corresponding BN gives the highest value of $P(G_i|C_1, C_2, \cdots, C_M)$. Alternatively, multiple goals can be identified by selecting the goals for which the BNs voted positive. In the case when all BNs vote negative, the input query is rejected as out-of-domain.

2.3.1.2 General Bayesian Networks

More general BNs have been used in [59] for dialogue act decoding. Each training utterance has been manually tagged using two layers of the DAMSL(Dialog Act Markup in Several Layers) multi-layer annotation scheme [25], which includes a layer of Forward-looking Functions (FF) and a layer of Backward-looking Functions (BF). Assuming that dialogues are between a server and a client, the dialogue context features selected as input to the BNs include the previous FF of the client, the previous BF of the server, the current FF decided by the client, and the sentence type, subject type and punctuation of current utterance.

Network structures are learned from the annotated training data using the K2 algorithm [77] and conditional probability distributions are learned using the *Maximum A Posterior* (MAP) technique as in [78].

2.3.2 Vector-Based Methods

Dialogue act decoding or goal detection can be considered as similar to topic identification in information retrieval. Users' utterances and corresponding dialogue acts are represented by vectors in a vector space. The detected dialogue act is the one whose corresponding vector is the closest to the vector representing the incoming query. Traditional methods which are widely used in information retrieval can be used to detect dialogue acts. Some of the methods described here include latent semantic analysis [60] and Kohonen's Self-Organizing Maps (SOM) [61].

2.3.2.1 Latent Semantic Analysis

Latent Semantics Indexing (LSI) [79] has been widely used in information retrieval. It uses a fully automatic statistical method based on singular value decomposition (SVD) to uncover the associations among terms in a large collection of texts, which in turn is used to construct a semantic or concept space wherein terms and documents that are closely associated are placed near one another. During retrieval, terms in a query are used to identify a point in the semantic space and documents in its neighborhood are returned to the user.

Inspired by these ideas, a similar technique has been applied for dialogue act decoding in a desktop command and control system [60; 80]. Here, latent semantic analysis (LSA) [81] is employed to map a user's utterance to a desired action. The algorithm works as follows:

- Let $\mathcal{V}$ be a vocabulary of size M, and $\mathcal{T}$ a collection of utterances used to invoke N actions. First construct a word-command $(M \times N)$ matrix W associated with $\mathcal{V}$ and $\mathcal{T}$. This matrix fully describes how often word w_i appeared in command d_j. Then perform the SVD of W as:

$$W \approx \hat{W} = U\ S\ V^T \tag{2.23}$$

  where U is the $(M \times R)$ left singular matrix with row vectors u_i $(1 \leq i \leq M)$, S is the $(R \times R)$ diagonal matrix of singular values, V is the $(N \times R)$ right singular matrix with row vectors v_j $(1 \leq j \leq N)$, $R \ll \min(M, N)$.

- As is well known, $U^T U = V^T V = I_R$, where I_R is the identity matrix of order R. Thus,

$$U^T W = SV^T \tag{2.24}$$

$$U^T d_j = S v_j^T = \chi_j \tag{2.25}$$

  That is, column vectors d_j are projected onto a lower dimensional subspace. χ_j characterizes the position of command d_j in this subspace.

- Mapping the transcription of a command onto an action is equivalent to finding the nearest χ_j in R-dimensional space given the command vector w_i.

One of the drawbacks of LSA is that it only works well for longer texts, its ability to discriminate different texts will be reduced if the average length of texts is short [82]. Therefore, the accuracy of dialogue act decoding using this method would not be high if the average length of users' utterances is short.

2.3.2.2 Self-Organizing Maps (SOM)

Dialogue act decoding may also be modelled using Self-Organizing Maps (SOM) [61], an unsupervised neural network method suitable for ordering and visualization of complex data sets. In an SOM, utterances are meaningfully organized to be topically ordered, i.e. utterances with similar topics are found near each other.

During training, dialogues are manually tagged with their corresponding goals and each continuous dialogue segment of several utterances that belongs to a particular goal

category forms a single document. Each document is then encoded as a vector using the widely adopted *term frequency×inverse document frequency (TFIDF)* method [83] in information retrieval. Such document vectors are then organized on an SOM which forms a kind of associative topic-semantic memory.

During goal identification, a user's query is encoded as a vector similar to the encoding of the training utterances. It is then mapped onto the SOM to retrieve the goal based on the fact that similar inputs, which are defined in terms of proximity of their descriptive vectors in the multidimensional input space, tend to be mapped close to each other on the SOM. It has been found that longer dialogue segments in the training data resulted in higher goal identification accuracy.

Similar work can also be found in [84] where the Learning Vector Quantization (LVQ) algorithm is first used to map the input vectors to a set of predefined target vectors, then the SOM algorithm is applied to output a two-dimensional map of the data items in which proximity of the data items is interpreted as similarity.

2.3.3 Call Routing Methods

A natural language call router/classifier takes recognized speech as input and determines where the call should be directed or whether the system should initiate a disambiguation dialogue. The first step of a call router does is to classify the incoming call into one of the pre-defined call types, which is quite similar to dialogue act detection. Some of the reported approaches include a vector-based information retrieval technique [85], probabilistic models with salient phrases [62; 63; 64], discriminative training [65] etc. As the vector-based technique similar to the one used in [85] has already been described in section 2.3.2, only the other two techniques will be reviewed here.

2.3.3.1 Probabilistic Models with Salient Phrases

AT&T's *How May I Help You* (HMIHY) [62; 63; 64] acts as a call router to recognize and understand user's speech to determine its call-type. It first automatically learns salient phrase and grammar fragments from a database of transcribed utterances labelled with associated machine actions based on the mutual information between words or phrases. Salience here is defined as the measurement of the information content of an event for a task [62]. It then recognizes these fragments in fluent speech by searching the output of a large vocabulary speech recognizer, whose language model is a stochastic finite state machine (SFSM) with the automatically-acquired phrases embedded within it. Finally, the recognized fragments are used to classify the call-type of an utterance based on the

peak of the *a posteriori* distribution $P(g_k|f)$, where g_k is the set of call actions and f is a fragment.

The procedure of the algorithm is summarized as follows:

1. *Extract salient phrase fragments.*

   (a) Acquire salient phrase fragment recursively based on the mutual information measure between two words or between a fragment and a word.

   $$I(f, w) = \log_2[P(w|f)/P(w)] \tag{2.26}$$

   where f is a fragment which contains one or more words, w is a word.

   (b) Compute the extra-linguistic associations via the salience measure of mutual information averaged over all call actions.

   $$S(f) = \sum_k P(g_k|f)I(f, g_k) \tag{2.27}$$

   where f is a fragment, g_k is the set of call-actions, and $S(f)$ is the fragment's salience measure.

2. *Combine salient phrase fragments into a grammar fragment.*
   For a certain call type, the first pass determines the salient words with the highest posterior distribution $P(g_k|f)$. The local context of this salient word is then evaluated based on syntactic and semantic associations. Syntactic association signifies the relationship between a fragment and the phrases following or proceeding this fragment, while semantic association refers to the relationship between a fragment in spoken language and the call-type corresponding to the speech. The same process is then repeated to construct fragments surrounding other salient words for this call type [86].

3. *Recognize fragments in speech.*
   The speech recognizer is constrained by a stochastic language model that approximates an n-gram model on variable-length phrase units, which are automatically acquired from the database based on their utility for minimizing the entropy of the training corpus.

4. *Classify an utterance to one of 15 call types.*
   For the recognized fragments f_i, select the call type with the maximum $P(g_k|f_i)$. If the peak value is less than some threshold, then the utterance is rejected.

2.3.3.2 Discriminative Training

In the vector-based information retrieval technique for call routing [85], a $n \times m$ routing matrix R is constructed where the rows represent n call types and the columns represent m terms (features). An input query is represented as an m-dimensional feature vector $\vec{x}$ and is classified based on the cosine similarity score with the model call type vectors encoded in the routing matrix, i.e.

$$calltype\ \hat{j} = \underset{j}{\text{argmax}}\ \cos\phi_j = \underset{j}{\text{argmax}}\ \frac{\vec{r_j} \cdot \vec{x}}{\| \vec{r_j} \| \| \vec{x} \|} \tag{2.28}$$

where $\vec{r_j}$ is the model document vector for call type j and $\vec{x}$ is the input query vector.

The above technique does not guarantee that the classification error rate will be minimized, discriminative training on vector-based models can be used to adjust model parameters in order to improve routing accuracy and robustness. A number of different discriminative techniques for vector-based routing were compared in [87], which included Generalized Probabilistic Descent (GPD), Corrective Training (CT), and Linear Discriminant Analysis (LDA). It was found that the GPD technique due to Kuo and Lee [65] was the most effective. Therefore, only the GPD technique proposed in [65] will be reviewed.

In Kuo and Lee's work, the *discriminant function* for call type j and observation vector $\vec{x}$ is first defined as the dot product of the model vector and the observation vector:

$$g_j(\vec{x}, R) = \vec{r_j} \cdot \vec{x} = \sum_{i=1}^{m} r_{ji} x_i \tag{2.29}$$

Given that the correct target call type for $\vec{x}$ is k, the *misclassification function* is defined as

$$d_k(\vec{x}, R) = -g_k(\vec{x}, R) + G_k(\vec{x}, R) \tag{2.30}$$

where

$$G_k(\vec{x}, R) = [\frac{1}{n-1} \sum_{j \neq k, 1 \leq j \leq n} g_j(\vec{x}, R)^{\eta}]^{\frac{1}{\eta}} \tag{2.31}$$

is the *anti-discriminant function* of the input $\vec{x}$ in call type k and $n-1$ is the number of competing call types. $d_k(\vec{x}, R) > 0$ implies misclassification. Equation 2.30 in fact converts a multi-dimensional decision function into a one-dimensional metric. The generalized probabilistic descent (GPD) algorithm [88] can then be used to optimize a non-decreasing function of this misclassification metric iteratively.

It was found that discriminative techniques for call routing produced very large reductions in the error rate on the training set but did not generalized well to the test set [87].

2.3.4 Transformation-Based Learning

Transformation-based Learning (TBL) [89] for dialogue act tagging was first introduced in [66] and subsequently applied by Araki in his semantic tagger [67] where users' dialogue acts are identified first and then used as features to determine the frame of a semantic representation.

TBL is a rule-based machine learning algorithm. Prior to learning, the initial tagged data was made from an unannotated corpus by using a bootstrapping method, where all the utterances are assigned the most frequent tag. Then rules are generated using a predefined set of rule templates by instantiating variables with certain features from the annotated data. Examples of such features include cue phrases, word n-grams, dialogue act cues[1], the sentence length, the previous dialogue act etc. The derived rules are applied to the unannotated data and the rule whose transformation results in the greatest improvement on the precision of newly tagged data is selected. This rule is added into the current rule set and the iteration is continued until no improvement is observed.

The advantage of TBL over other standard machine learning techniques is that it is error driven and is therefore optimizing directly the error rate rather than other presumably correlated metrics. However, one of the disadvantages is that the rule templates must be supplied in advance. If the templates are too specific, the model is limited to only consider certain features or feature combinations. On the other hand, if the templates are too general, the model would turn to be less efficient. Moreover, the most expensive step in TBL is the computation of the scores of the transformation rules and efficient algorithms are needed in order to reduce the computational complexity.

2.3.5 Topic Trees

Jokinen *et. al* [68] hand-coded user's goals (topics) as a topic tree for a particular dialogue corpus. The tree both constrains and enables prediction of what is likely to be talked about next, and provides a top-down approach to dialogue coherence. Each topic is represented by a vector whose individual element is the mutual information between a word and the topic. Topic shift probabilities are estimated using the CMU-CAM language modelling toolkit [90].

There are two ways to determine the topic of an input utterance. The first one is to predict a topic, called *predicted topic*, given the previous topic sequences based on the topic shift probabilities. The second one is to decide a topic, called *supported topic*, according to the total mutual information of each topic type and the words in the input utterance. For

[1] Dialogue act cues are word substrings that appear frequently in dialogue and provide useful clues to help determine the appropriate dialogue acts.

each topic type g_i, the amounts of mutual information $mi(g_i; w_j)$ are summed up where w_j is one of the words occurred in the input utterance. Then the topic types are ranked in descending order and the top one is selected. If all the words are unknown or out of domain, then the predicted topic is chosen, otherwise the supported topic is selected. The accuracy of topic identification is about 75% using this algorithm.

2.3.6 Discussion

Meng *et al.* treats dialogue act detection as an inference problem and uses Bayesian networks (BNs) to solve it. That is, given all available concept evidence $\vec{C}$ about an utterance, the purpose is to find its dialogue act or information goal that has the highest posterior probability $P(G|\vec{C})$:

$$\hat{G} = \underset{G}{\text{argmax}}\ P(G|\vec{C}) \tag{2.32}$$

Thus, the BN approach essentially provides soft decisions and it is straightforward to generate the top N dialogue acts. That is, multiple dialogue acts may be inferred from BNs.

When viewing dialogue act detection as an information retrieval (IR) problem, all the traditional vector-based models used for IR such as LSA, SOM etc may also be used. However, the vector-based approaches are not able to handle utterances which simultaneously encode more than one dialogue act.

The call routing method used in HMIHY identifies salient phrases first, which is essentially a generalization of word spotting. The detected salient phrase fragments are then used to classify users' utterances into one of the 15 predefined call types. Thus, this method takes recognized words from the speech recognizer instead of semantic concepts as input. It does not make use of the knowledge provided by the semantic parser, which is usually important to dialogue act detection especially when there might be some possible speech recognition errors and the semantic module would still be able to derive the correct overall meaning of the sentence.

Other approaches require significant human effort such as a set of rule templates have to be defined in advance for the TBL and the topic hierarchies also need to be defined in the topic tree approach.

After reviewing the pros and cons of various approaches to dialogue act decoding, Belief Networks (BNs) appears to be the most promising one due to the following reasons. First it provides a soft decision and thus multiple dialogue acts can be detected. Second, various knowledge sources such as the semantic concepts extracted, the system's beliefs, users' beliefs etc can be integrated into one probability model and the most probable

dialogue act can be found through belief updating. In addition, methods exist for network topology and parameter learning. Inference algorithms on BNs are also well-developed and can be applied easily. Therefore, automatic learning using the BNs defines another major objective of this research.

Strictly speaking, evaluating BNs is in general NP-hard [91]. Therefore, many restricted classes of BNs have been proposed by making independence assumptions between certain network nodes such that the network evaluation can be solved efficiently. So there is a trade-off between capturing all conditioning context and efficiently evaluating BNs. One restricted class of BNs is Naive Bayes classifier where all attribute nodes are assumed conditionally independent given the class node. Evaluation of this kind of networks can be solved in linear time. Obviously, the above assumption is unrealistic. One way to tackle this problem is to add dependency links between attributes and it results in the Tree-Augmented Naive Bayes networks (TAN) [18]. It was reported that TAN maintains the robustness and computational complexity of Naive Bayes, and at the same time displays better accuracy [18]. Thus, the TAN networks are explored for dialogue act detection in this research and a detailed discussion is presented in Chapter 4.

2.4 Parser Model Adaptation

Statistical model adaptation has been studied extensively by researchers working on acoustic modelling for automatic speech recognition (ASR), language modelling, PCFG adaptation to a novel domain etc. The two most common approaches used in adaptation for continuous probability distributions such as in the area of ASR are Bayesian adaptation, which uses a maximum *a posteriori* (MAP) probability criteria [92], and transformation-based adaptation such as maximum likelihood linear regression (MLLR) [93], which uses a maximum likelihood (ML) criteria. The adaptation methods used for discrete probability distributions include MAP adaptation applied in n-gram language models [94] and lexicalized PCFG models [95], Markov transform [96] and householder transform [97] used in statistical semantic parser adaptation. This section reviews some of the adaptation methods used for discrete probability distributions.

2.4.1 Maximum a Posteriori (MAP)

In the maximum *a posteriori* (MAP) framework described in [92], the model parameter vector θ is assumed to be a random vector taking values in the space Θ and estimated from the sample $\mathbf{x}$ with probability density function (pdf) $f(\cdot|\theta)$. Given the prior pdf

of θ which is denoted as g, the the MAP estimate, θ_{MAP}, is defined as the mode of the posterior pdf of θ denoted as $g(\cdot|x)$, i.e.

$$\theta_{MAP} = \underset{\theta}{\text{argmax}}\ g(\theta|x) = \underset{\theta}{\text{argmax}}\ f(x|\theta)g(\theta) \tag{2.33}$$

In the MAP formulation, three key issues to be addressed are the choice of the prior distribution family, the specification of the parameters for the prior densities, and the evaluation of the maximum a posterior. In the case of n-gram model adaptation [94], the estimation of probabilities for a discrete distribution over words is analogous to the estimation of the distribution across mixture components within a mixture density in ASR. Therefore, a Dirichlet density which is the conjugate prior for the prior distribution of the K-mixture component weights $\omega_1, \omega_2, \cdots, \omega_K$ can be used

$$g(\omega_1, \omega_2, \cdots, \omega_K|\nu_1, \nu_2, \cdots, \nu_K) \propto \prod_{i=1}^{K} \omega_i^{\nu_i - 1} \tag{2.34}$$

where $\nu_i > 0$ are the parameters of the Dirichlet distribution. With such a prior, if the expected counts for the ith component is denoted as c_i, the mode of the posterior distribution is obtained as

$$\hat{\omega}_i = \frac{(\nu_i - 1) + c_i}{\sum_{k=1}^{K}(\nu_k - 1) + \sum_{k=1}^{K} c_k} \quad 1 \leq i \leq K \tag{2.35}$$

In a PCFG model [95], assume a set of production rules have the form $A \rightarrow \gamma$, where A is a non-terminal symbol and γ is, in general, a string consisting of non-terminal or terminal symbols. Let γ_i denote the ith expansion of A, $c(A \rightarrow \gamma_i)$ the expected adaptation count, and $\tilde{P}(\gamma_i|A)$ the probability estimate for the production $A \rightarrow \gamma_i$ according to the out-of-domain model, then the parameters of the prior distribution for A are chosen as

$$\nu_i^A = \tau_a \tilde{P}(\gamma_i|A) + 1 \quad 1 \leq i \leq K \tag{2.36}$$

where τ_A is the left-hand side dependent prior weighting parameter. This choice of prior parameters defines the MAP estimate of the probability of expansion τ_i from the left-hand side A as

$$\tilde{P}(\gamma_i|A) = \frac{\tau_A \tilde{P}(\gamma_i|A) + c(A \rightarrow \gamma_i)}{\tau_A + \sum_{k=1}^{K} c(A \rightarrow \gamma_k)} \quad 1 \leq i \leq K \tag{2.37}$$

An over-parameterization problem may arise if each left-hand side A has its own prior distribution. This however can be avoided by adopting parameter tying approach. Two methods of parameter tying, counting merging and model interpolation, can be used. If τ_A is chosen as

$$\tau_A = \tilde{c}(A)\frac{\alpha}{\beta} \tag{2.38}$$

the MAP adaption reduced to count merging, scaling the out-of-domain counts with a factor α and the in-domain counts with a factor β

$$\hat{P}(\gamma_i|A) = \frac{\alpha\tilde{c}(A \to \gamma_i) + \beta\bar{c}(A \to \gamma_i)}{\alpha\tilde{c}(A) + \beta\bar{c}(A)} \tag{2.39}$$

where $\tilde{P}$ and $\tilde{c}$ denote the probabilities and counts from the out-of-domain model, and $\bar{P}$ and $\bar{c}$ denote the probabilities and counts from the adaptation model (i.e. in-domain).

If τ_A is chosen as

$$\tau_A = \begin{cases} \bar{c}(A)\frac{\lambda}{1-\lambda},\ 0 < \lambda < 1 & \text{if } \bar{c}(A) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.40}$$

the MAP adaptation reduces to model interpolation using interpolated parameter λ

$$\hat{P}(\gamma_i|A) = \lambda\tilde{P}(\gamma_i|A) + (1-\lambda)\bar{P}(\gamma_i|A) \tag{2.41}$$

Roark *et al.* [95] show that the supervised and unsupervised MAP adaptation gives 2.5% and 5% gains respectively on F-measure parsing accuracy when tested on the WSJ corpus using the model originally trained on the Brown corpus.

2.4.2 Transformation-Based Approaches

Transformation-based approaches such as MLLR applies a general transform on some clusters of model parameters. Since all the model parameters belonging to the same cluster are transformed simultaneously, such approaches are quite effective when only a small amount of adaptation data is available. However, the main drawback is a rapid saturation of the performance improvement as the amount of adaptation data increases [98]. Here, two transformation-based approaches, the Markov transform and the Householder transform used on parser models, will be described.

2.4.2.1 Markov Transform

Luo *et al.* [96] proposed to map an initial parser model to a new model by a Markov matrix where the transform matrix is obtained by maximizing the log likelihood of the parses of test data encoded using the model before adaptation. Assume the initial model $\mathcal{M}_0$ consists of M probability mass functions (pmf), i.e.

$$\mathcal{M}_0 = P_e \ :\ P_e \text{ is a pmf; } \mathrm{e} = 1, 2, \cdots, \mathrm{M} \tag{2.42}$$

Let K be the dimension of each (row vector) P_e. The ith component of P_e is written as $P_e(i)$. Consider a $K \times K$ Markov matrix $Q = [q_{ij}]$ which is subject to the constraints

$$\sum_{j=1}^{K} q_{ij} = 1 \quad \text{where } i = 1, 2, \cdots, K \tag{2.43}$$

$$q_{ij} \geq 0 \qquad \text{where } i, j = 1, 2, \cdots, K \tag{2.44}$$

The transform is defined as

$$\hat{P}_e = P_e Q \tag{2.45}$$

where both $\hat{P}_e$ and P_e are row vectors. It can be easily verified that the transformed $\hat{P}_e$ is still a valid pmf since

$$\sum_{j=1}^{K} \hat{P}_e(j) = \sum_{j=1}^{K} \sum_{i=1}^{K} P_e(i) q_{ij} = \sum_{i} P_e(i) \sum_{j} q_{ij} = 1 \tag{2.46}$$

Let $C_e(i)$ be the count for the ith component of P_e collected using data $\mathcal{C}_1$ from a new domain, Q can then be optimized by maximizing the log likelihood of parses of data $\mathcal{C}_1$ measured by the transformed model $\mathcal{M}_1 = \hat{P}_e(\cdot)$, i.e.

$$\hat{Q} = \underset{Q}{\text{argmax}}\ L(Q) \tag{2.47}$$

$$= \underset{Q}{\text{argmax}} \sum_{e=1}^{M} \sum_{j=1}^{K} C_e(j) \log \hat{P}_e(j) \tag{2.48}$$

$$= \underset{Q}{\text{argmax}} \sum_{e=1}^{M} \sum_{j=1}^{K} C_e(j) \log(\sum_{i=1}^{K} P_e(i) q_{ij}) \tag{2.49}$$

Since the value of K is typically small and constraints in equation 2.43 and 2.44 are independent, each row of Q can be optimized sequentially.

This technique has been applied to IBM's semantic classer in air travel domain and a 23% to 31% relative reduction of parsing errors is obtained.

2.4.2.2 Householder Transform

A Markov transform requires $K^2 - K$ free parameters for each transform, where K is the number of values to be predicted in the model. When K reaches several hundred, the estimation of transform parameters becomes computationally complex. In view of this, a special orthogonal transform, the Householder transform, is proposed in [97] for statistical parser adaptation, which only requires $K - 1$ free parameters to be predicted per transform.

The Householder matrix in Euclidean space is defined as

$$H = I - 2ww^T \tag{2.50}$$

where I is an identity matrix, w is a unit vector, and w^T is the transpose of w. The Householder matrix is orthogonal since $H^T H = I$. The Householder transform, also called a *reflector*, is defined as

$$f(x) = Hx \tag{2.51}$$

where x is a vector in Euclidean space. It in fact reflects a vector x along a hyperplane perpendicular to w.

For statistical parser adaptation, a pmf is first mapped from simplex to unit sphere by taking the square root of each component of the pmf, then the Householder transform is applied on the unit sphere, and finally, the transformed pmf is mapped back to simplex by taking the square of each component of the pmf. If similar notations as those described in section 2.4.2.1 are used, let $\hat{P}_e$ be the new pmf after transformation, the proposed transform is then

$$\hat{P}_e(i) = (\sqrt{P_e(i)} - 2w(i)(w^T\sqrt{P_e}))^2, \quad 1 \leq i \leq K \tag{2.52}$$

As in [96], parameters in the Householder transform are optimized by maximizing the probability of parses of adaptation data. Instead of a global Markov transform used in [96], multiple transforms are used in [97]. A bottom-up classification algorithm is applied to classify M pmfs in the initial statistical model $\mathcal{M}_0$ into L classes $\{\tau_i : i = 1, 2, \cdots, L\}$, where $L \leq M$ and $\{\tau_i\}$ form a partition of pmfs in $\mathcal{M}_0$. The classification algorithm is described as

1. Let each pmf P_e such that $\sum_k C_e(k) > 0$ be a class, and initialize the centroid of each class with P_e where the centroid of a class is defined as arithmetic average of pmfs in the class.

2. For each class whose count is smaller than δ, merge it with its closest class. Go to Step 4 if no such class can be found.

3. Recompute the centroid of the new class. Go to Step 2.

4. For each P_e such that $\sum_k C_e(k) = 0$, classify it to the class closest to P_e.

The distance measure used in the algorithm is L_2 norm between two centroids, or between a pmf and a centroid, depending on whether the distance in question is between two classes or between a pmf and a class. δ is the minimum count for each transform class.

Given L classes $\{\tau_i : i = 1, 2, \cdots, L\}$, distributions in τ_i share a Householder transform, parameterized by $w^{(i)}$. Therefore, the totality of parameters of Householder transforms is written as $W = \{w^{(l)} : l = 1, 2, \cdots, L\}$. Accordingly, the log likelihood of the parses of the adaptation data is

$$L(W) = \sum_{l=1}^{L}\sum_{e\in\tau_l}\sum_{k=1}^{K} C_e(k)\log\hat{P}_e(k; w^{(l)}) \tag{2.53}$$

$$= \sum_{l=1}^{L}\sum_{e\in\tau_l}\sum_{k=1}^{K} C_e(k)\log(\sqrt{P_e(k)} - 2w^{(l)}(k)\sqrt{P_e}^T w^{(l)})^2 \tag{2.54}$$

Each $w^{(l)}$ can be optimized independently subject to the constraint

$$\| w^{(l)} \|_2 = 1, \quad l = 1, 2, \cdots, L \tag{2.55}$$

Tests have also been conducted on the classer of IBM AirTravel system and a 21% to 24% relative reduction of parsing errors is obtained, which is slightly worse than the results obtained using a Markov transform as described in Section 2.4.2.1.

2.4.3 Discussion

MAP adaptation has been successfully applied in the area of speech recognition and has recently been adopted in n-gram model adaptation and PCFG adaptation. In MAP estimation, the parameters of the model are considered to be random variables themselves with a known distribution (the prior). The prior distribution and the maximum likelihood distribution based on the in-domain (adaptation) observations then gives a posterior distribution over the parameters. As the amount of the adaptation data increases, the contribution of the prior becomes negligible and the posterior distribution is dominated by the maximum likelihood estimate. Since MAP estimation only updates the parameters of models which are observed in the adaptation data, in general, fairly large amounts of adaptation data are required.

In Luo's work [96], a linear transform, the Markov transform, has been proposed to globally update the model parameters of a statistical parser. However, computational complexity issues arise when the number of values to be predicted for each pmf becomes moderately large. Also because of inherent constraints (i.e. each row has to sum to 1) on Markov matrices, diagonal matrices can not be used, as could be done in acoustic model adaptation using MLLR. A special orthogonal Householder transform has been applied in statistical parser adaptation [97] which in fact absorbs the constraints imposed on the Markov transform because of the orthogonality property of the Householder matrix. The number of free parameters to be predicted per transform is therefore reduced to $K - 1$ as opposed to $K^2 - K$ free parameters in the Markov transform. However, because the number of free parameters is small, a global Householder transform is unlikely to be good for all pmfs. Therefore, multiple transforms have to be used which require pmfs to be classified into L classes first.

Though performance improvement has been reported using both the Markov transform and the Householder transform, there is no explicit specification of the amount of adaptation training data used in Luo's papers. Also, the optimization procedure is not straightforward and it is not easy to derive transform parameters.

Since MAP adaptation has been applied successfully in PCFG parsers, it has been investigated for the HVS model parser adaptation in this research. Transformation-based approaches have not been taken into consideration as they involve nonlinear programming for this particular application of HVS adaptation, which is difficult to derive close form solutions. As one of the special forms of MAP adaptation is in fact interpolation between the in-domain and out-of-domain models, attention has also been paid to nonlinear interpolation between the baseline model and the model trained solely from the adaptation training data and this has been studied as well. The detailed work on HVS parser model adaptation is discussed in Section 5.4.2.

Chapter 3

Semantic Parsing using the Hidden Vector State Model

This chapter describes the Hidden Vector State (HVS) model. First the overall structure of the HVS model is outlined, followed by the definition of the model parameters. Then, procedures by which the model can be trained using minimally annotated training data are presented. Finally, experimental results using the HVS model are compared with whose obtained from the baseline finite-state tagger (FST) model.

3.1 Overview of the HVS model

Consider the semantic parse tree shown in Fig. 3.1, the semantic information relating to each word is completely described by the sequence of semantic concept labels extending from the preterminal node to the root node. If these semantic concept labels are stored as a single vector, then the parse tree can be transformed into a sequence of vector states as shown in the lower portion of Fig. 3.1. For example, the word `Dallas` is decribed by the semantic vector `[CITY, TOLOC, RETURN, SS]`. Viewing each vector state as a hidden variable, the whole parse tree can be converted into a first order vector state Markov model, this is the *Hidden Vector State (HVS)* model.

Each vector state is in fact equivalent to a snapshot of the stack in a push-down automaton. Indeed, given some maximum depth of the parse tree, any PCFG formalism can be converted to a first-order vector state Markov model. If viewing each vector state as a stack, then state transitions may be factored into a stack shift by n positions followed by a push of one or more new preterminal semantic concepts relating to the next input word. If such operations are unrestricted, then the state space will grow exponentially and the same computational tractability issues of hierarchical HMMs are incurred. However, by imposing constraints on the stack operations, the state space can be reduced to a manageable size. Possible constraints to introduce are limiting the

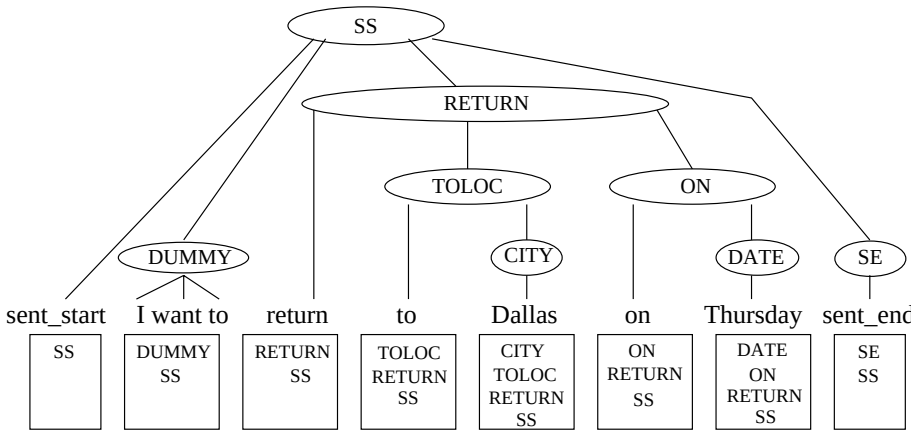


Figure 3.1: Example of a parse tree and its vector state equivalent.

maximum stack depth and only allowing one new preterminal semantic concept to be pushed onto the stack for each new input word. These constraints ensure that the size of the underlying probability tables are linear in stack depth, number of concept labels, and vocabulary size. Such constraints effectively limit the class of supported languages to be right-branching. Although left-branching structures do exist in English, the majority of sentences can be represented as right-branching structures. In addition, right-branching structures are generally preferred because they reduce the working memory needed to represent a sentence [99].

Note also that although any parse tree can be converted into a sequence of vector states, it is not a one-to-one mapping and ambiguities may arise. However for the semantic parsing task defined in this thesis, what are interested is not the exact parse tree to be recovered, but are the concept/value pairs to be extracted. Even if there are ambiguities, the extracted concept/value pair will be the same. For example, assuming A and B are semantic concept labels and x and y are words, the partial parse trees B(A(x) A(y)) and B(A(x y)) would share a common HVS representation and hence would yield the same concept/values B.A=x and B.A=y. If the entities x and y were actually distinct, then the preterminal labels would have to be made unique.

3.2 Definition of the HVS model

The joint probability $P(N, \mathbf{C}, W)$ of a series of stack shift operations N, a concept vector sequence $\mathbf{C}$, and a word sequence W can be decomposed as follows

$$P(N, \mathbf{C}, W) = \prod_{t=1}^{T} P(n_t | W_1^{t-1}, \mathbf{C}_1^{t-1}) P(c_t[1] | W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t) P(w_t | W_1^{t-1}, \mathbf{C}_1^{t}) \qquad (3.1)$$

where

- $\mathbf{C}_1^t$ denotes a sequence of vector states $\mathbf{c}_1..\mathbf{c}_t$. $\mathbf{c}_t$ at word position t is a vector of D_t semantic concept labels (tags), i.e. $\mathbf{c}_t = [c_t[1], c_t[2], .., c_t[D_t]]$ where $c_t[1]$ is the preterminal concept immediately dominating the word w_t and $c_t[D_t]$ is the root concept (SS in Fig. 3.1),
- $W_1^{t-1}\mathbf{C}_1^{t-1}$ denotes the previous semantic parse up to position $t-1$,
- n_t is the vector stack shift operation and takes values in the range $0, .., D_{t-1}$,
- $c_t[1] = c_{w_t}$ is the new preterminal semantic tag assigned to word w_t at word position t.

In terms of the generic model given by equation 2.18, each move has now been factored into 3 terms: choice of how many elements to pop from the stack, the new concept label to push onto the stack and the word to generate given the new vector state.

The stack transition from $t-1$ to t given preterminal semantic concept tag c_{w_t} for word w_t is

$$c_t[1] = c_{w_t} \tag{3.2}$$

$$c_t[2..D_t] = c_{t-1}[(n_t+1)..D_{t-1}] \tag{3.3}$$

$$D_t = D_{t-1} + 1 - n_t \tag{3.4}$$

Thus n_t defines the number of semantic tags which will be popped off the stack before pushing on c_{w_t}. The case $n_t = 0$ corresponds to growing the stack by one element i.e. entering a new semantic tag. The case $n_t = 1$ corresponds to simply replacing the preterminal at word position $t-1$ by c_{w_t} at word position t, the rest of the stack being unchanged. The case $n_t > 1$ corresponds to shifting the stack i.e. popping off one or more semantic tags.

In the version of the HVS model discussed in this thesis, equation 3.1 is approximated by

$$P(n_t|W_1^{t-1}, \mathbf{C}_1^{t-1}) \approx P(n_t|\mathbf{c}_{t-1}) \tag{3.5}$$

$$P(c_t[1]|W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t) \approx P(c_t[1]|c_t[2..D_t]) \tag{3.6}$$

$$P(w_t|W_1^{t-1}, \mathbf{C}_1^{t}) \approx P(w_t|\mathbf{c}_t) \tag{3.7}$$

An alternative approximation to equation 3.6 also investigated here is:

$$P(c_t[1]|W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t) \approx P(c_t[1]|\mathbf{c}_{t-1}) \tag{3.8}$$

where the preterminal semantic tag $c_t[1]$ is now conditioned on the whole of the preceding state vector rather than just that part which remains after the stack has been popped by

n_t elements. In this thesis, the system using equation 3.6 is referred to as HVS-Partial and the system using equation 3.8 as HVS-Full. Note that if the maximum stack size is set to 1 and $n_t = 1$, the HVS-Full model becomes equivalent to a conventional flat-concept bigram model.

3.3 Training the HVS model

This section discusses training the HVS model to perform hierarchical semantic parsing. It first describes the preprocessing steps which are needed to add lexical features to the training data and augment the abstract annotations, then it briefly describes model initialization, and parameter re-estimation.

3.3.1 Training assumptions

Before training a statistical semantic parsing model from unannotated data, it is natural to ask what kinds of *a priori* knowledge can be easily provided by the dialogue designer. There are two obvious possibilities:

- *A set of domain specific lexical classes.* For example, in an air travel domain, it is possible to group all airline names into one single class `AIRLINE_NAME`. Such domain specific classes can normally be extracted automatically from the application domain database schema.
- *Abstract semantic annotation for each utterance.* Such an annotation need only list a set of valid semantic concepts and the dominance relationships between them without considering the actual realized concept sequence or attempting to identify explicit word/concept pairs.

The provision of abstract annotations implies that the dialogue designer must define the semantics that are encoded in each training utterance but need not provide an utterance level parse. It effectively defines the required input-output mapping whilst avoiding the need for expensive tree-bank style annotations. For example, in an air travel domain, a dialogue system designer may define the following hierarchical semantic relationships:

- `FLIGHT(FROMLOC(CITY) TOLOC(CITY))`
- `AIRLINE(FROMLOC(CITY) TOLOC(CITY) DEPART(DAY_NAME))`
- `GROUND_SERVICE(CITY_NAME)`
- `RETURN(TOLOC(CITY) ON(DATE))`

- ...

Having defined such a set of hierarchical semantic relationships, annotation is simply a method of associating the appropriate semantics with each training utterance and does not require any linguistic skills. For example, when building a system from scratch, a dialogue designer can take each possible abstract schema in turn and give examples of corresponding natural language forms, as in

`RETURN(TOLOC(CITY(X)) ON(DATE(D)))` →

1. I would like to return to X on D.
2. I wanna return on D to X.
3. I want to return to X on D.

Alternatively, if a corpus of representative user utterances are already available, perhaps obtained via Wizard-of-Oz style data collection, then each utterance can easily be tagged with the appropriate abstract annotation as in

"I want to return on Thursday to Dallas"

← `RETURN(TOLOC(CITY(Dallas)) ON(DATE(Thursday)))`

In the training data used for the evaluations reported later in this thesis, corpora of real user utterances were used. Annotation was done by extracting abstract semantics from the accompanied SQL reference queries or the available parse results by a rule-based semantic parser which is essentially similar to the annotation method described in the latter case.

Note finally that no assumptions are made regarding access to any syntactic information, neither explicitly via grammar rules nor implicitly via syntactic treebank data.

3.3.2 Preprocessing

As mentioned in section 3.3.1, two kinds of *a priori* knowledge are assumed to be available or can be provided by dialogue system designers: a set of domain specific lexical classes and abstract annotations for each training utterance.

The first step needed to train the HVS model is to replace all class members by their corresponding class names. Where there is ambiguity, the class covering the largest span of words is replaced first. Where a word or phrase may occur in more than one class, the first class encountered is chosen arbitrarily. Better solutions to this problem clearly exist, but in practice the problem is quite rare and it does not signficantly affect performance.

Recalling again the example illustrated in section 3.3.1, in the case of the HVS model, the abstract annotation

```
RETURN(TOLOC(CITY(Dallas)) ON(DATE(Thursday)))
```

corresponding to the utterance "I want to return on Thursday to Dallas" would be expanded to the flattened concept sequence

```
RETURN  RETURN+TOLOC RETURN+TOLOC+CITY(Dallas)  RETURN+ON
RETURN+ON+DATE(Thursday)
```

In order to cater for irrelevant input words, a DUMMY tag is allowed everywhere in preterminal positions. In order to accommodate this, the vector state sequence is finally expanded to

```
RETURN  RETURN+DUMMY  RETURN+TOLOC  RETURN+TOLOC+DUMMY
RETURN+TOLOC+CITY(Dallas)  RETURN+TOLOC+CITY(Dallas)+DUMMY
RETURN+ON  RETURN+ON+DUMMY  RETURN+ON+DATE(Thursday)
RETURN+ON+DATE(Thursday)+DUMMY
```

Note that this final set of vector states only provides the set of valid semantic vector states that can appear in the parse results of the current utterance. As explained further below, this set is used as a constraint in the EM re-estimation algorithm. It does not define the actual vector state transition sequence. Further note that the total number of distinct vector states required to use the HVS model for a particular application can be enumerated directly from this expanded vector state list.

The system only allows the DUMMY tag to appear in preterminal positions, therefore, for consecutive irrelevant word inputs, the model will stay in the same vector state. Only when a relevant word input is observed will the DUMMY tag together with zero or more preceding semantic tags be popped off from the previous vector state stack and a new preterminal tag will be pushed into the stack accordingly. For a more complex domain, it may be useful to allow a DUMMY tag anywhere in a vector state, for example, RETURN+DUMMY+TOLOC may also be considered as valid. The experimental results using both ways to augment the vector states by the DUMMY state are compared in Section 3.6.3.

3.3.3 Parameter initialization

Equations 3.5 to 3.7 define the three main components of the HVS model, namely, the stack shift operation, the push of a new preterminal semantic tag, and the selection of a new word. Thus, each vector state is associated with three sets of probabilities: a vector of stack shift probabilities, a vector of semantic tag probabilities, and a vector of output probabilities. The cardinalities of these three sets of probabilities are determined by the maximum vector state stack depth, the number of preterminal semantic tags, and the vocabulary size, respectively. Hence, the total information required to define an HVS model is:

- number of distinct vector states
- number of preterminal semantic tags
- maximum vector state stack depth
- vocabulary size
- for each vector state
  - state name
  - stack shift operation probability vector
  - tag transition probability vectors (push of a new preterminal tag)
  - output probability vector

Having acquired all of the information required for a particular application domain, the prototype topology of an HVS model can be defined. One example of such a prototype definition is shown in Fig. 3.2 which uses a format similar to that described for HMM definitions in the HTK Toolkit [100].

The first line of the definition contains a macro type ~o which specifies global features of the HVS model, such as the total number of distinct vector states, number of preterminal semantic tags, maximum stack depth, vocabulary size etc. Another macro ~h indicates an HVS definition whose name is specified by the following string ''hvsStk4''. The HVS definition is bracketed by the symbol <BeginHVS> and <EndHVS>. Parameters associated with each vector state are enumerated within the HVS definition and are listed following the keyword <State>. For example, a stack shift operation probability vector is introduced by the keyword <StackOp>, a tag transition probability vector is introduced by the keyword <TagTrans> for HVS-Partial or <TagFullTrans> for HVS-Full, and finally, an output probability vector is introduced by the keyword <Output>.

It should be noted that the actual number of semantic tags kept in a vector state is $n+1$ for the stack depth n as the root tag SS exists in all vector states. Therefore, though the maximum stack depth is 4 in Fig. 3.2, the vector size of stack shift probabilities is 5 as can be seen from $0.200 * 5$. Here, to allow efficient coding, successive repeated values are represented as a single value plus a repeat count in the form of an asterisk followed by an integer multiplier, e.g. $0.200 * 5$ represents the probability 0.200 repeated by 5 times.

A so-called *flat start* is used whereby all model parameters are initially made identical. These parameters are then iteratively refined using the EM based re-estimation procedure described in the next section.

```
~o
<NumStates> 2799
<NumTermTags> 85
<MaxStackDepth> 4
<VocabSize> 611
~h "hvsStk4"
<BeginHVS>
    <State> 1
       <StateName> SS
       <StackOp> 0.200*5
       <TagTrans> 0.015
       <TagFullTrans> 0.012*85
       <Output>
          0.000*255 0.000*225 1.000 0.000*130
    <State> 2
       <StateName> SS+DUMMY
       <StackOp> 0.200*5
       <TagTrans> 0.015
       <TagFullTrans> 0.012*85
       <Output>
          0.002*255 0.002*255 0.002*101
    <State> 3
       <StateName> SS+FLIGHT
       <StackOp> 0.200*5
       <TagTrans> 0.015
       <TagFullTrans> 0.012*85
       <Output>
          0.002*255 0.002*255 0.002*101
       ...
<EndHVS>
```

Figure 3.2. Definition for an HVS model.

3.3.4 Parameter re-estimation

There are no explicit word level annotations in the training corpora, hence parameter estimation based on event counts cannot be used and forward-backward estimation must be applied instead. The same situation applies to parameter estimation on the finite-state model which will be discussed in Section 3.5.1.3.

Let the complete set of model parameters be denoted by λ, EM-based parameter estimation aims to maximize the expectation of $L(\lambda) = \log P(N, \mathbf{C}, W|\lambda)$ given the observed

data and current estimates. To do this, define the auxiliary Q function as:

$$\begin{aligned} Q(\lambda|\lambda^k) &= \mathrm{E}\left[\log P(N, \mathbf{C}, W|\lambda)|W, \lambda^k\right] \\ &= \sum_{\mathbf{C},N} P(N, \mathbf{C}|W, \lambda) \log P(N, \mathbf{C}, W|\lambda^k) \end{aligned} \tag{3.9}$$

The approximation of $P(N, \mathbf{C}, W)$ given by equations 3.5 to 3.7 is

$$P(N, \mathbf{C}, W) = \prod_{t=1}^{T} P(n_t|\mathbf{c}_{t-1}) P(c_t[1]|c_t[2..D_t]) P(w_t|\mathbf{c}_t) \tag{3.10}$$

Substituting equation 3.10 into 3.9 and differentiating leads to the following re-estimation formulae

$$\hat{P}(n|\mathbf{c}') = \frac{\sum_t P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)}{\sum_t P(\mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)} \tag{3.11}$$

$$\hat{P}(c[1]|c[2..D]) = \frac{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda)}{\sum_t P(c_t[2..D] = c[2..D]|W, \lambda)} \tag{3.12}$$

$$\hat{P}(w|\mathbf{c}) = \frac{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda)\delta(w_t = w)}{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda)} \tag{3.13}$$

where $\delta(w_t = w)$ is one iff the word at time t is w, otherwise it is zero.

For the HVS-Full model, equation 3.12 becomes

$$\hat{P}(c[1]|\mathbf{c}') = \frac{\sum_t P(c_t[1] = c[1], \mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)}{\sum_t P(\mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)} \tag{3.14}$$

The key components of equation 3.11 to 3.13 are the likelihoods $P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)$, $P(c_t[2..D] = c[2..D]|W, \lambda)$ and $P(\mathbf{c}_t = \mathbf{c}|W, \lambda)$. These can be efficiently calculated using the forward-backward algorithm. First define the forward probability α as

$$\alpha_{\mathbf{c}}(t) = P(w_1, w_2, ..., w_t, \mathbf{c}_t = \mathbf{c}|\lambda) \tag{3.15}$$

Similarly, define the backward probability β as

$$\beta_{\mathbf{c}}(t) = P(w_{t+1}, w_{t+2}, ..., w_T|\mathbf{c}_t = \mathbf{c}, \lambda) \tag{3.16}$$

Then, omitting the conditioning on W and λ for clarity, the required likelihoods are

$$P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}') = \alpha_{\mathbf{c}'}(t-1) \cdot P(n_t = n|\mathbf{c}_{t-1} = \mathbf{c}') \cdot \sum_{\{\mathbf{c}|\mathbf{c}' \Rightarrow \mathbf{c}\}} P(c[1]|c[2..D]) P(w_t|\mathbf{c}) \beta_{\mathbf{c}}(t) \tag{3.17}$$

$$P(c_t[2..D] = c[2..D]) = \sum_{\{\mathbf{c}|c_t[2..D] = c[2..D]\}} \alpha_{\mathbf{c}}(t)\beta_{\mathbf{c}}(t) \tag{3.18}$$

$$P(\mathbf{c}_t = \mathbf{c}) = \alpha_{\mathbf{c}}(t)\beta_{\mathbf{c}}(t) \tag{3.19}$$

Finally, the forward and backward probabilities are defined recursively as follows

$$\alpha_{\mathbf{c}}(t) = \sum_{\{\mathbf{c}'|\mathbf{c}'\Rightarrow\mathbf{c}\}} \alpha_{\mathbf{c}'}(t-1)P(\mathbf{c}|\mathbf{c}')P(w_t|\mathbf{c}) \tag{3.20}$$

$$\beta_{\mathbf{c}}(t) = \sum_{\{\mathbf{c}''|\mathbf{c}\Rightarrow\mathbf{c}''\}} P(\mathbf{c}''|\mathbf{c})P(w_{t+1}|\mathbf{c}'')\beta_{\mathbf{c}''}(t+1) \tag{3.21}$$

where

$$P(\mathbf{c}|\mathbf{c}') = P(n^*|\mathbf{c}')P(c[1]|c[2..D]) \tag{3.22}$$

Here, n^* is the value of stack shift needed to map $\mathbf{c}'$ into $\mathbf{c}$.

The notation $\mathbf{c}' \Rightarrow \mathbf{c}$ denotes all valid or legal derivations from any state vector $\mathbf{c}'$ to the state vector $\mathbf{c}$. This is the place where we apply the constraints on the dominance relationships between semantic concepts as described in section 3.3.1 of the main text.

For the HVS-Full model, equation 3.12 becomes

$$\hat{P}(c[1]|\mathbf{c}') = \frac{\sum_t P(c_t[1] = c[1], \mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)}{\sum_t P(\mathbf{c}_{t-1} = \mathbf{c}'|W, \lambda)} \tag{3.23}$$

where the likelihood $P(c_t[1] = c[1], \mathbf{c}_{t-1} = \mathbf{c}')$ is calculated as

$$P(c_t[1] = c[1], \mathbf{c}_{t-1} = \mathbf{c}') = \sum_{n=0}^{D} \alpha_{\mathbf{c}'}(t-1)P(n_t = n|\mathbf{c}_{t-1} = \mathbf{c}')P(c[1]|\mathbf{c}')P(w_t|\mathbf{c})\beta_{\mathbf{c}}(t) \tag{3.24}$$

In all of the above formulae, a single training utterance is assumed. As in the case of regular HMMs, the extension to multiple utterances is straightforward.

During training, two constraints are applied in the estimation process:

1. For each utterance, a state transition is only allowed if both incoming and outgoing states can be found in the corresponding semantic annotation.

2. If the observed word is a class name (such as `CITY`), then only semantic concepts (states) which contain this class name can be associated with the word (eg `TOLOC.CITY`, `FROMLOC.CITY` but not `FROMLOC`). In addition, in order to cater for irrelevant words, the `DUMMY` tag is allowed everywhere. That is, state transitions from or to the `DUMMY` state are always allowed.

The following example illustrates how these two constraints are applied. Consider again the annotation for the utterance "I want to return on Thursday to Dallas" which was in abstract form

```
RETURN  RETURN+DUMMY  RETURN+CITY(X)  RETURN+CITY(X)+DUMMY
RETURN+DATE(D)  RETURN+DATE(D)+DUMMY
```

The transition from `RETURN+CITY(X)` to `RETURN` is allowed since both states can be found in the semantic annotation. However, the transition from `RETURN` to `FLIGHT` is not allowed as `FLIGHT` is not listed in the semantic annotation. Also, for the lexical item `X` in the training utterance, the only valid vector state is `RETURN+CITY(X)` since `X` has to be bound with the preterminal tag `CITY`.

3.4 Implementation

The hierarchical semantic concepts are represented by a tree structure, an example of such a tree is shown in Figure 3.3. As all abstract annotations begin with an SS tag (sentence start marker), all the vector states defined by a set of semantic concepts share the same root node. A distinct vector state can be traversed from the root node to any of the intermediate nodes or leaf nodes. For example, the vector states are SS, SS+FLIGHT, SS+FLIGHT+FROMLOC, and SS+FLIGHT+FROMLOC+CITY when doing left-most travering on the semantic concept tree in Figure 3.3.

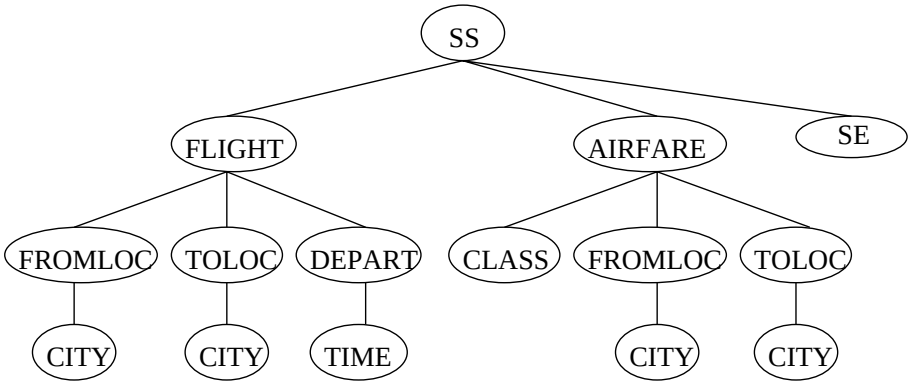


Figure 3.3: An example of hierarchical semantic concept tree.

Assume the number of distinct semantic concepts is N and the maximum vector state stack depth is D, then the fully populated HVS model state space is defined by N^D. Usually, N takes the order of 10^2 and D will be at least 3 or 4 to be useful. The complete state space is therefore very large. In practice, only those vector states that appear in the training data are stored. The probabilities of unseen states are then computed using back-off or smoothing. In the implementation here, the vector states are stored in linked lists which are illustrated in Figure 3.4.

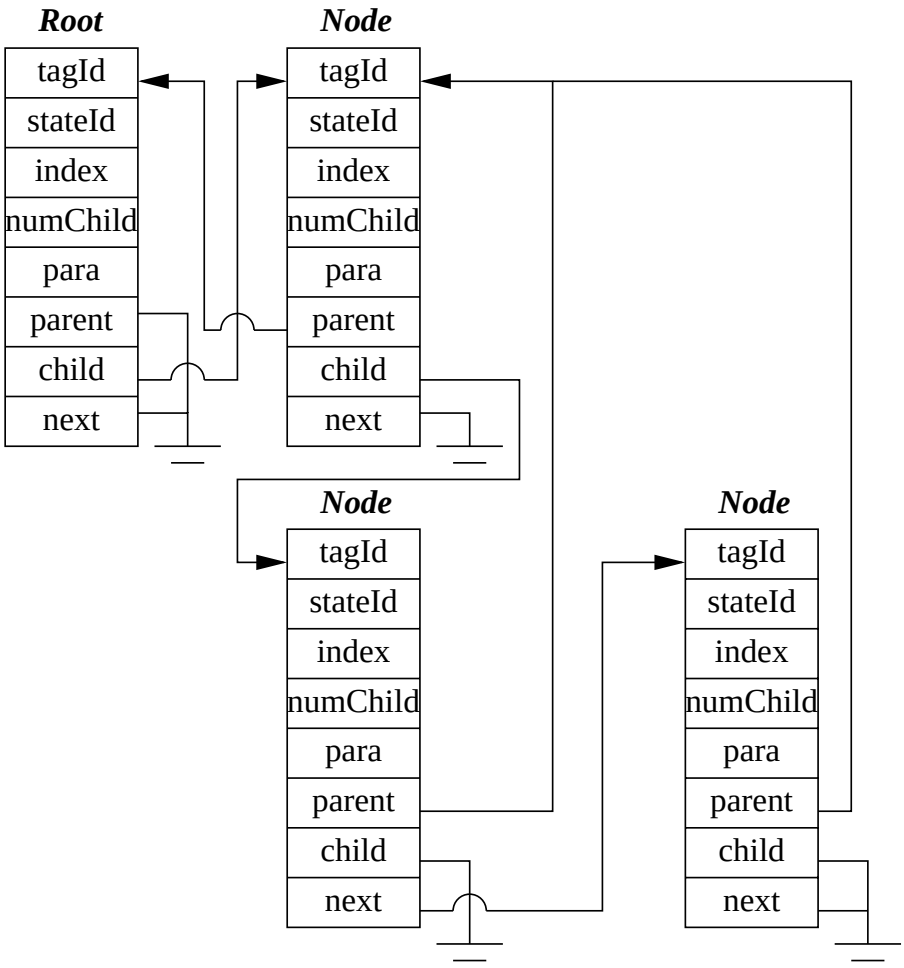


Figure 3.4: Linked lists of vector states.

Each node has the same data structure with the root points to the node containing SS. The tagId stores the semantic concept name of the current node, examples include FROMLOC, CITY, TIME etc. The stateId stores the vector state name which contains all the semantic concept names traversing from the root to the current node, for example, the stateId of the left-most leaf node in Figure 3.3 would store "SS+FLIGHT+FROMLOC+CITY". The index and termIdx are symbolic representations of the current vector state and semantic concept respectively and normally take integer values. The numChild field stores the total number of immediate child concepts. For example, in Figure 3.3, numChild will be 3 for the SS node. There are three pointers defined in each node. The parent pointer points to the immediate predecessor of the current node, the child pointers points to the first child in the lower level of the semantic concept tree, and finally, the next pointer links to other nodes in the same tree level. For example, for the FLIGHT node in Figure 3.3, its parent points to the SS node, child points to the FROMLOC node, and next points to the

AIRFARE node.

There is also a para field in each node. It points to a data structure that stores parameters to be estimated during EM training, which are the stack shift operation probability, the tag transition probability, and the output probability as specified in Figure 3.2.

3.5 Experimental evaluation

In order to assess the performance of the HVS model, a finite-state tagger (FST) has been built as a baseline for comparison. Section 3.5.1 describes the structure, parameter estimation, and training procedures of the FST model. Section 3.5.2 presents the experimental setup on the ATIS corpus and the DARPA Communicator data.

3.5.1 Baseline - finite-state semantic model

This section briefly describes a general finite-state tagger (FST) of the type used in CHRONUS [32] mentioned in section 2.2.1.

3.5.1.1 Overview of the FST model

An FST model treats the generation of a spoken sentence as an HMM-like process whose hidden states correspond to semantic concepts and outputs correspond to individual words in the utterance. Fig. 3.5 shows an example of a parse C output using an FST model. Each word is tagged with a single discrete semantic concept label and the utterance is enclosed by sentence start and end markers, `sent_start` and `sent_end`, to enable the prediction of the first and last word in the utterance. The corresponding semantic tags for them are `SS` and `SE` respectively.

Figure 3.5: An example of finite-state tagger parse result.

Note that if required, the output of the FST can be trivially converted to a well-formed parse tree simply by adding a sentence tag dominating all FST state tags.

3.5.1.2 Definition of the FST model

For a general FST model,

1. assume a sentence is represented by a word sequence $W = (w_1 \cdots w_T)$ where $w_t \in \mathcal{V}$, $\mathcal{V}$ denotes the vocabulary;

2. assume each word w_t in a sentence is tagged with one semantic concept label c_t, the corresponding tag sequence is $C = (c_1 \cdots c_T)$ where $c_t \in \mathcal{T}$, $\mathcal{T}$ denotes the semantic concept space;

3. the FST model can then compute the required joint distribution $P(W, C)$ as follows where each move d_t in the generic form of equation 2.18 is realised by first generating a concept label c_t given the concept history and then generating the corresponding word w_t given c_t and the word history:

$$P(W|C)P(C) = \prod_{t=1}^{T} P(w_t|w_{t-1} \cdots w_1, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \cdots c_1) \tag{3.25}$$

   In practice, following equation 2.19, the history is truncated to be of finite length:

$$P(W|C)P(C) \approx \prod_{t=1}^{T} P(w_t|w_{t-1} \cdots w_{t-n+1}, c_t) \prod_{t=1}^{T} P(c_t|c_{t-1} \cdots c_{t-m+1}) \tag{3.26}$$

If $n = 1$ and $m = 2$, this becomes a conventional first order Markov model with words modelled as a unigram conditioned on the semantic concept c_t and state transitions given by concept bigram probabilities.

3.5.1.3 Training the FST model

Similar to the training of the HVS model, the training procedure for an FST model also consists of three steps: *preprocessing*, *parameter initialization*, and *parameter re-estimation*.

The *preprocessing* step first replaces class members found in the training utterances with their corresponding class names. The abstract annotation associated with each utterance is then expanded to form a sequence of concept labels. Note that the abstract annotations for the FST model do not provide any hierarchical dominance relationships, they simply list out valid semantic concepts that can appear in the parse results. Furthermore, as in the example below, the order of the concept labels is not necessarily monotonic relative to the actual word order.

For example, the abstract annotation

```
RETURN(TOLOC(CITY(Dallas)) ON(DATE(Thursday)))
```

mentioned in section 3.3.1 corresponding to the utterance "I want to return on Thursday to Dallas" would be expanded to the concept sequence [1]

[1] Note that in practice compound semantic concepts such as TOLOC.CITY are used instead of the atomic semantic tags TOLOC and CITY assumed by the pure model illustrated in Fig. 3.5. These compounds are limited to those which can be derived directly from the database schema and their use is a heuristic needed to make the FST model work acceptably well in practical systems. The HVS model needs no such heuristic.

```
RETURN TOLOC TOLOC.CITY ON ON.DATE
```

The parameters of the FST model include output vectors for each state and a state transition matrix. The size of the semantic concept space determines the total number of distinct states whilst the vocabulary size determines the output vector size. The *parameter initialization* step simply initializes all of the parameters to be identical.

Finally, the *parameter re-estimation* step uses the forward-backward algorithm to estimate parameters. Note that this is different from CHRONUS where maximum likelihood estimation based on relative frequency counts on a fully annotated ATIS training set was used. There are no explicit word level annotations in the training corpora, hence as mentioned in Section 3.3.4, forward-backward estimation must be applied instead of parameter estimation based on event counts. Two constraints are applied during the estimation process. First, for each utterance, a state transition is only allowed if both incoming and outgoing states can be found in the corresponding semantic annotation. Second, if the observed word is a class name (such as `CITY`), then only semantic concepts (states) which contain this class name can be associated with the word (eg `TOLOC.CITY`, `FROMLOC.CITY` but not `FROMLOC`). In addition, in order to cater for irrelevant words, the `DUMMY` tag is allowed everywhere. That is, state transitions from or to the `DUMMY` state are always allowed.

3.5.2 Experimental Setup

Experiments have been conducted on a relatively simple corpus - ATIS [20] which contains air travel information data, and a more complex corpus - DARPA Communicator Travel Data [21] which contains not only air travel related data but also information on hotel reservation, car rental etc. The ATIS training set consists of 4978 utterances selected from the Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora whilst the ATIS test set contains both the ATIS-3 NOV93 and DEC94 datasets. The DARPA Communicator Travel Data are available to the public as open source download. The data contain utterance transcriptions and the semantic parse results from the Phoenix parser [101]. These parse results were hand corrected and then used to generate both the abstract training annotations and the test set reference results for the experiments. Table 3.1 gives the overall statistics of both ATIS and DARPA data.

In order to evaluate the performance of the model, a reference frame structure was derived for every test set utterance consisting of slot/value pairs. An example of such a reference frame is:

Table 3.1: Statistics of the ATIS and DARPA Communicator Data

	No. of Utterances	
Dataset	*Training Set*	*Test Set*
ATIS	4978	448 (NOV93)
		445 (DEC94)
DARPA Communicator Data	12702	1178

```
Show me flights from Boston to New York.
Frame: FLIGHT
Slots: FROMLOC.CITY = Boston
       TOLOC.CITY = New York
```

Performance was then measured in terms of F-measure on slot/value pairs [102], which combines the precision (P) and recall (R) values with equal weight and is defined as

$$F = \frac{2PR}{P+R} \tag{3.27}$$

Various smoothing techniques have been tested and it was found that linear discounting for transition probabilities and Good-Turing for output probabilities yielded the best result for the FST model, whereas Witten-Bell for vector state stack operation probabilities, push of a new preterminal tag, and output probabilities achieve the highest F-measure score for the HVS model [33; 103].

3.5.2.1 ATIS

A set of 30 domain-specific lexical classes were extracted from the ATIS-3 database as listed in Table 3.2 which contains the complete set of lexical classes extracted from both ATIS and DARPA Communicator corpora. The training data was then preprocessed to replace class members by their corresponding class names as described in sections 3.3.1 and 3.5.1.3. Abstract semantics for each training utterance were derived semi-automatically from the SQL queries provided in ATIS-3.

After the parse results have been generated for the test sets, post-processing is required to extract relevant slot/value pairs and convert them into a format compatible with the reference frames. This post-processing depends on a pre-defined list of semantic tags to be ignored, examples of which include the sentence start/end markers, the `DUMMY` tag, the tags about departure and arrival indicators such as `FROMLOC, TOLOC, DEPART, ARRIVE` etc. The extraction of slot/value pairs from parse results then follows the rules below:

Table 3.2: Lexical classes extracted from ATIS and DARPA Communicator data

Corpus	Lexical Classes
ATIS	aircraft_code, airline_code, airline_name, airport_code, airport_name, city_code, city_name, class_type, cost_relative, country_name, day_name, day_number, days_code, fare_basis_code, flight_mod, flight_stop, flight_time, manufacturer, meal_code, meal_description, month_name, period_of_day, restriction_code, round_trip, state_code, state_name, time, today_relative, transport_type, year
DARPA	airline_name, airport_code, airport_name, city_name, continent_name, country_name, day_name, day_number, hotel_name, month_name, period_of_day, rental_company, rental_type, round_trip, state_name, time, today_relative, year

- ignore the vector state if its preterminal tag is in the tag ignore list;
- if a preterminal tag is about location or date/time, then search the indicators of departure or arrival in its corresponding vector state in a bottom-up manner, extract the first encountered indicator tag;
- for all other vector states, only extract preterminal tags, e.g., for the word `flight` which is tagged as `SS+FLIGHT`, the extracted slot/value pair is `FLIGHT=flight`.

Note that in the slot/value pairs extraction rules described above, essentially only the semantic tag ignore list needs to be built manually, which is relatively straightforward as the number of preterminal semantic tags for any particular task is normally limited. For example, in the experimental work reported here, there are only 85 preterminal tags for the ATIS corpus and 65 preterminal tags for the DARPA Communicator data. Hence, it is easy to identify those tags to be ignored.

Taking the HVS parse result shown in Fig. 3.8 as an example, the vector state associated with the word `Denver` is `SS+FLIGHT+ARRIVE+FROMLOC+CITY`, the preterminal tag `CITY` indicates that `Denver` is a location and the vector state has two location indicators `ARRIVE` and `FROMLOC`. However, only `FROMLOC` will be extracted as it is the first indicator found by the system when the vector state is scanned from pre-terminal tag back up towards the root tag. Therefore, the slot/value pair extracted will be `FROMLOC.CITY = Denver`. The full list of slots extracted in this example would be:

```
FLIGHT = flight
ARRIVE.CITY = Burbank
FROMLOC.CITY = Denver
ARRIVE.DAY_NAME = saturday
```

3.5.2.2 DARPA Communicator Travel Data

The DARPA Communicator Travel Data were collected in 461 days and consist of 2211 dialogues or 38408 utterances in total. From these, 46 days were randomly selected for use as test set data and the remainder were used for training. As the data provided contain only utterances from the user's side of the conversation, there exist some seemingly meaningless utterances such as "No, thank you", "Yes, please" etc., which were classified as "Respond" class in the reference annotations generated by the Phoenix parser. These utterances plus short backchannel utterances for which "No Parse" could be found have no defined semantics outside of an interactive dialogue context and hence have been excluded. The statistics of the DARPA Communicator Travel Data are shown in Table 3.3. After cleaning up the data, the training set consists of 12702 utterances while the test set contains 1178 utterances.

Table 3.3: Statistics of the DARPA Communicator Travel Data

Criteria	*Training Set*	*Test Set*
Collection dates	415	46
Total utterance number	35531	2877
Utt. number after removing "No Parse"	29947	2442
Utt. number after further removing "Respond"	13060	1192
Utt. number after further removing "no slots extracted"	12702	1178

A set of 18 domain-specific lexical classes were extracted from the DARPA Communicator database. As mentioned earlier, the abstract annotations used for training and the reference annotations needed for testing were derived by hand correcting the Phoenix parse results. It should be emphasised here that the use of the Phoenix parser for generating the abstract training annotations was purely a matter of convenience. If the parser had not been available, it would have been straightforward to manually generate the abstract annotations using a simple annotation tool. It would certainly have taken far less time to do this, than it would have taken to provide detailed treebank annotations.

3.6 Experimental results

The performance of the HVS model was evaluated using both the ATIS and the DARPA Communicator corpora. The performance of the FST model on the same data was also evaluated to provide a baseline for comparison. The following sections present the evaluation results in detail.

3.6.1 Evaluation of the FST baseline system

The FST model was implemented according to equation 3.25 and experiments were conducted to find the best values for n and m where $n \in (1, 2)$ and $m \in (2, 3)$. The results listed in Table 3.4 indicate that the first order FST model ($n = 1, m = 2$) gives the best performance results on both the ATIS and DARPA Communicator Travel Task corpora, this model was therefore chosen as the baseline system for subsequent experiments.

Table 3.4: Performance Comparison of FST model on various language modelling techniques on ATIS and DARPA Communicator data

Lang. Model		**ATIS**			**DARPA Communicator**		
States	*Outputs*	*Recall*	*Precision*	*F-measure*	*Recall*	*Precision*	*F-measure*
bigram	unigram	86.71%	84.84%	85.77%	82.94%	82.34%	82.64%
bigram	bigram	74.72%	67.55%	70.95%	68.52%	77.39%	72.68%
trigram	unigram	62.15%	64.33%	62.59%	71.70%	59.90%	65.27%
trigram	bigram	61.32%	56.51%	58.81%	49.68%	46.69%	48.14%

3.6.2 Comparison of the HVS model with the FST model

Experiments have been conducted using the FST model, the HVS-Partial model, and the HVS-Full model and the results are listed in Table 3.5. Both HVS models outperform the FST model, with HVS-Partial improving on the FST model by 4.1% for a simple corpus like ATIS and 6.6% for the more complex DARPA Communicator Travel corpus.

The DARPA Communicator data contains a large number of short utterances due to short responses to system queries. For example, if the system asks *"What time do you want to leave Denver"*, the user may simply answer *"In the morning"*. Among 1178 test utterances, 847 utterances have length less than five words, while 8516 out of 12702 training utterances contain only four or less words. Table 3.6 gives results partitioned by

Table 3.5: Performance comparison of FST, HVS-Partial and HVS-Full models on ATIS and DARPA data

	ATIS			**DARPA Communicator**		
Measurement	*FST*	*HVS (Partial)*	*HVS (Full)*	*FST*	*HVS (Partial)*	*HVS (Full)*
Recall	86.71%	90.21%	85.99%	82.94%	87.31%	84.30%
Precision	84.84%	92.00%	88.08%	82.34%	88.84%	89.26%
F-measure	85.77%	91.10%	87.02%	82.64%	88.07%	86.71%

utterance length. These results show that the performance of the FST and HVS model do degrade with utterance length. The HVS-Partial model outperforms the FST model by 4.9% for short utterances and 9.6% for long utterances. From the above, we can conclude that the HVS model always outperforms the FST model and the improvement increases with more complex data and longer sentences.

Table 3.6: Performance comparison of FST, HVS-Partial and HVS-Full models on long and short utterances in DARPA Communicator corpus

	Utterance length <= 4 (847)			**Utterance length > 4 (331)**		
Measurement	*FST*	*HVS (Partial)*	*HVS (Full)*	*FST*	*HVS (Partial)*	*HVS (Full)*
Recall	88.42%	92.01%	90.07%	74.16%	79.74%	75.09%
Precision	91.57%	96.78%	96.63%	69.31%	77.40%	78.09%
F-measure	89.97%	94.33%	93.24%	71.65%	78.56%	76.56%

3.6.3 Comparison of the HVS models with `DUMMY` insertion

As mentioned in Section 3.3.2 that all the vector states in the abstract annotations are augmented by the `DUMMY` tag prior to training. The `DUMMY` tag may be appended at the end of each vector state or inserted between any two semantic tags in one vector state. An example of the former is `SS+FROMLOC+CITY_NAME+DUMMY` whilst an example of the latter is `SS+DUMMY+FROMLOC+DUMMY+CITY_NAME`. It is obvious that inserting the `DUMMY` tag everywhere in the vector states makes the HVS model more complicated as the state space would be enlarged. However, parse accuracy might be improved as more search path hypotheses would be available and the chances of getting the correct parse tree might be increased. Thus, there is a trade-off between the model complexity and the parse accuracy. Experiments have been conducted to compare the results of these two kinds of augmentation of the `DUMMY` tag, which are denoted as DUMMY-tail and DUMMY-all in Table 3.7.

It can be observed from Table 3.7 that there is no significant difference for the ATIS corpus. However, the model trained using DUMMY-all performs better than that using DUMMY-tail for the DARPA data. Further experiments have been conducted on the long and short utterances in the DARPA data and the results are shown in Table 3.8. The HVS model trained using either DUMMY-tail or DUMMY-all performs almost the same for the short utterances. But the model trained using DUMMY-all gives much better results for the long utterances due to significantly increased precision. Error analysis shows that the

Table 3.7: Performance comparison of HVS models on ATIS and DARPA data using different augmentation of the DUMMY tag

	ATIS		DARPA Communicator	
Measurement	*DUMMY-tail*	*DUMMY-all*	*DUMMY-tail*	*DUMMY-all*
Recall	90.21%	89.66%	87.31%	88.03%
Precision	92.00%	91.85%	88.84%	92.76%
F-measure	91.10%	90.76%	88.07%	90.34%

DUMMY-all model is able to filter out irrelavent input words more naturally. For example, for the utterance `I don't want to leave Durham`, the parse results given by the DUMMY-tail and DUMMY-all models are illustrated in Figure 3.6 and 3.7 respectively.

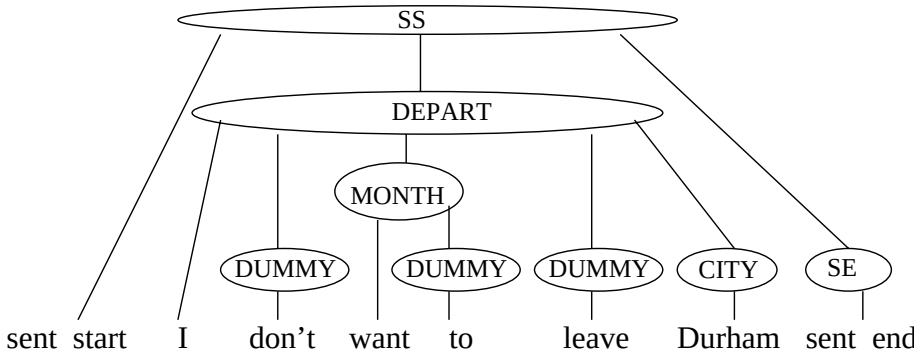


Figure 3.6: Parse results given by the DUMMY-tail model.

The extracted slot/value pairs for these two models are

DUMMY-tail→

`DEPART.MONTH_NAME = want, DEPART.CITY_NAME = Durham`

DUMMY-all→

`NEGATIVE = don't, DEPART.CITY_NAME = Durham`

For this utterance, the F-measure value given by the DUMMY-tail model and the DUMMY-all model is 0.5 and 1 respectively.

3.6.4 Comparative parse examples generated by the FST and HVS models

The ability of the HVS model to represent hierarchical structure is illustrated in the parse examples given in Fig. 3.8 which have been extracted from the ATIS test results. It is clear that `saturday` has the strongest correlation with the phrase `arrives in` and should be interpreted as `ARRIVE_DATE`. However, such long distance dependency requires at least 5-grams to capture, which is beyond the range of the FST. As a consequence, the FST model

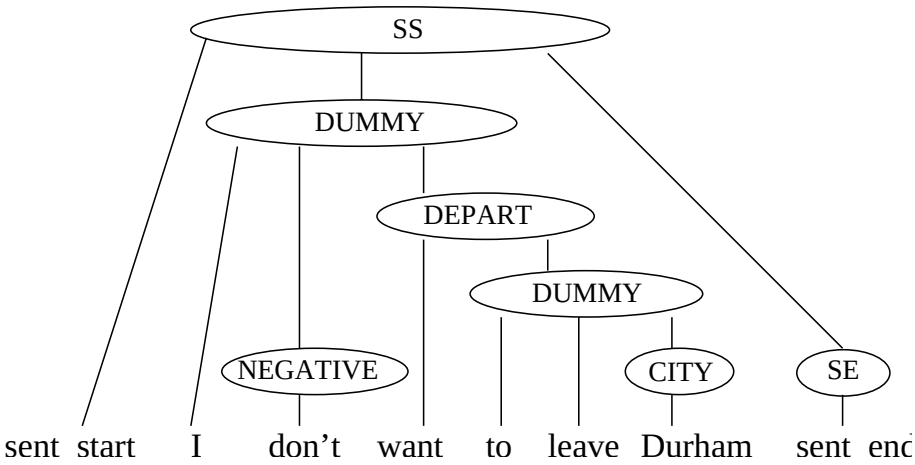


Figure 3.7: Parse results given by the DUMMY-all model.

Table 3.8: Performance comparison of HVS models on long and short utterances in DARPA Communicator corpus using different augmentation of the DUMMY tag

	Utterance length <= 4 (847)		**Utterance length > 4 (331)**	
Measurement	*DUMMY-tail*	*DUMMY-all*	*DUMMY-tail*	*DUMMY-all*
Recall	92.01%	91.78%	79.74%	81.96%
Precision	96.78%	96.85%	77.40%	86.38%
F-measure	94.33%	94.25%	78.56%	84.11%

erroneously tagged `saturday` as `DEPARTURE_DATE` since it follows `FROMLOC.ON`. In contrast, the HVS model was able to reproduce the hierarchical structure of the utterance. It carried forward the `ARRIVE` context in the stack and thereby properly interpreted `saturday` as an `ARRIVE_DATE`.

3.6.5 Optimal vector stack depth

The stack depth in the HVS models determines the number of previous semantic tags which can be encoded as historical context. Figure 3.9 shows the variation of system performance as a function of the maximum stack depth where solid lines represent the F-measure of the ATIS test set and dashed lines represent the F-measure for the DARPA test data[1]. The HVS-Partial model always outperforms the HVS-Full model for both ATIS corpus and DARPA travel data. The optimal stack depth is 4 for both HVS models and this is consistent with the hierarchical complexity of the abstract semantic concepts defined in the training utterances.

[1]Note here the actual number of semantic tags kept in a vector state is $n+1$ for the stack depth n as the root tag *SS* exists in all vector states

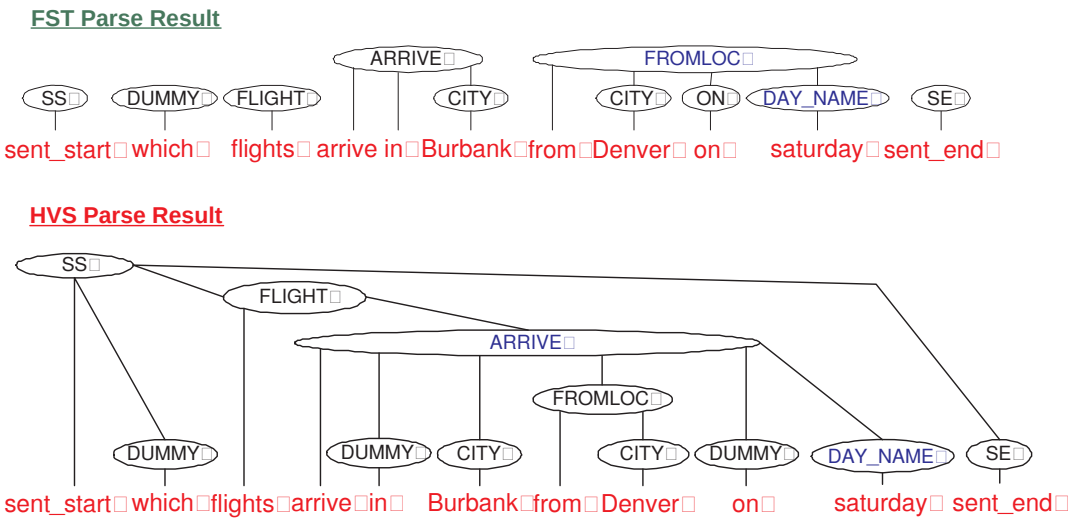


Figure 3.8: Comparison of parses generated by the FST and HVS models.

As mentioned in section 3.2, theoretically, the HVS-Full model with stack depth 1 should be analogous to the FST model with bigram state transitions. However, it can be seen that in fact the F-measure of HVS-Full is 0.38 and 0.58 for ATIS and DARPA data respectively, whereas the F-measure values for the FST model are 0.86 and 0.83. The relatively improved performance of the FST model is a consequence of using compound semantic concepts in the FST model (eg `TOLOC.CITY` instead of the atomic semantic tags assumed by the pure model). This use of compound tags is a heuristic needed to make the FST model work acceptably well in practical systems. The HVS model needs no such a heuristic.

3.6.6 Model complexity

Another interesting issue is the number of free parameters in each model. This is related to the total number of distinct states, the number of preterminal tags, and the vocabulary size. Table 3.9 gives the relevant statistics for the FST model and HVS models of stack depth 4. Although the state space for the HVS model is large relative to the FST model, the possible transitions between any two states are constrained by the maximum depth of the vector state stack and the condition that only one new semantic tag can be added per input word. Note also that although the FST has far fewer parameters than the HVS model, it cannot be claimed that it is being unnecessarily handicapped in the comparison with the HVS model since, as shown in section 3.6.1, the simple expedient of increasing the history lengths in the FST model to take advantage of more free parameters does not yield any improvement in performance.

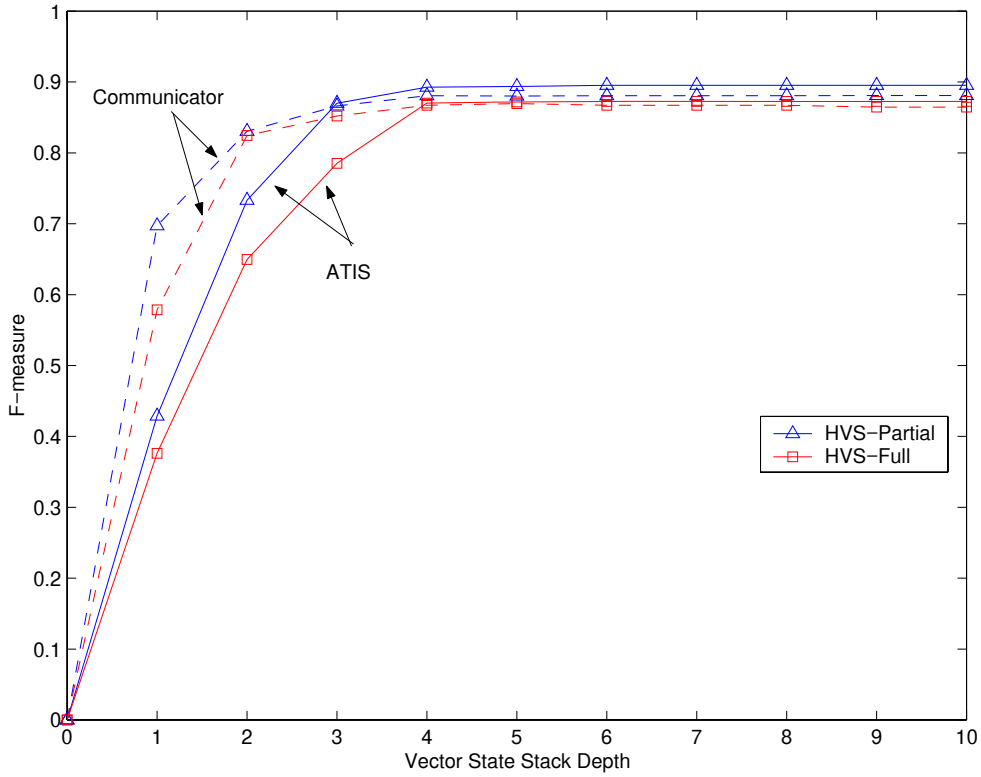


Figure 3.9: Slot F-measure vs stack depth.

3.7 Conclusions

The major problem of building a statistical model for semantic processing is how to collect large quantities of training data in order to reliably estimate model parameters. In general, fully annotated tree-bank data are not available for most applications. An ideal situation would be where the initial training data could be easily provided by the dialogue designer and where the semantic parsing model could be trained directly from unannotated data in a constrained way whilst at the same time being able to capture the underlying hierarchical

Table 3.9: Statistics of model parameters.

	ATIS			**DARPA Communicator**		
Model Parameters	*FST*	*HVS (Partial)*	*HVS (Full)*	*FST*	*HVS (Partial)*	*HVS (Full)*
No. of states	120	2799	2799	111	705	705
No. of preterminal tags	120	85	85	111	65	65
Vocabulary size	611	611	611	612	612	612
Total parameters	87720	1726983	1962099	80253	435690	480810

semantic structures. This is the motivation underlying the interest in the Hidden Vector State (HVS) model.

In this chapter, a baseline Finite State Tagger (FST) model and two variants of the HVS model have been built and tested on the ATIS and DARPA Communicator Travel Data. The experimental results have shown that these models can be trained directly from the abstract semantics without the use of word-level annotations. However, unlike the FST, the HVS model is able to capture hierarchical structure in the semantics. Furthermore it can do this without incurring the computational tractability issues inherent in fully recursive models such as the hierarchical HMM.

Chapter 4

Dialogue Act Decoding using Tree-Augmented Naive Bayes Networks

Bayesian networks or belief networks (BNs) can be viewed as directed probabilistic graphical models where nodes represent random variables, and arcs represent conditional independence assumptions. In addition to the graph structure or network topology, network parameters need to be specified by a Conditional Probability Distribution (CPD) at each node. If the variables are discrete, this CPD can be represented as a table (CPT), which lists the probability that the child node takes on each of its different values for each combination of values of its parents.

As discussed in Chapter 2, evaluating BNs is in general NP-hard [91] and therefore there is a trade-off between capturing all conditioning context and efficiently evaluating BNs. One restricted class of BNs is Naive Bayes classifier where all attribute nodes are assumed conditionally independent given the class node. Evaluation can then be solved in linear time. A natural extension of Naive Bayes Networks, Tree-Augmented Naive Bayes networks (TAN) [18], relax this conditional independence assumption by allowing dependency links between attributes. It was reported that TAN maintains the robustness and computational complexity of Naive Bayes, and at the same time displays better accuracy [18]. Thus, the TAN networks are explored for dialogue act detection in this research.

The basic Naive Bayes classifier learns from training data the conditional probability of each semantic concept C_i given the goal G_u[1], $P(C_i|G_u)$. Classification is done by picking the goal with the highest posterior probability of G_u given the particular instance of

[1]Capital letters such as C, G are used for variable names, and lowercase letters such as c, g are used to denote specific value taken by those variables. Sets of variables are denoted by boldface capital letters such as $\mathbf{C}, \mathbf{G}$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters $\mathbf{c}, \mathbf{g}$.

concepts $C_1 \cdots C_n$, $P(G_u|C_1 \cdots C_n)$. The strong independence assumption made in Naive Bayes networks is that all the concepts C_i are conditionally independent given the value of the goal G_u. TAN networks relax this independence assumption by allowing dependencies between concepts. They are however still a restricted family of Bayesian networks since the goal variable has no parents and each concept has as parent the goal variable and at most one other concept. An example of such a network is given in Figure 4.1 where each concept may have one augmenting edge pointing to it.

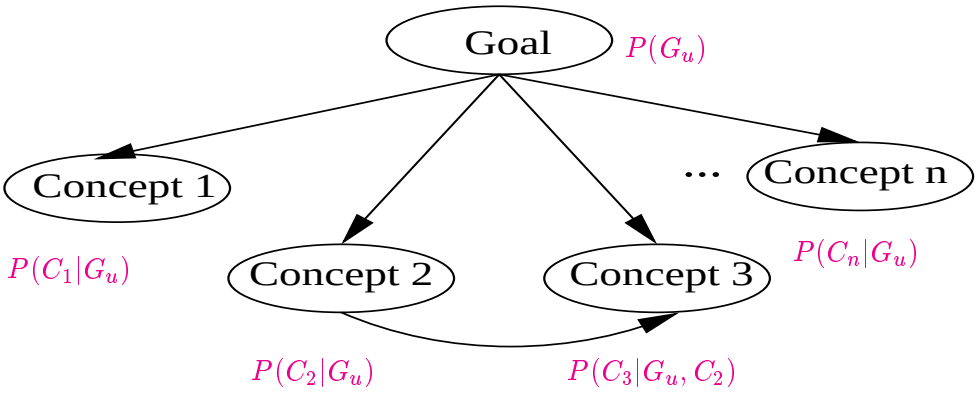


Figure 4.1: Example of a Tree-Augmented Naive Bayes Network.

In the dialog act decoder discussed here, one TAN was used for each goal and the semantic concepts which serve as input to each TAN were selected based on the mutual information (MI) between the goal and the concept. Dependencies between concepts were then added based on the conditional mutual information (CMI) between concepts given the goal.

The rest of the chapter is organized as follows. Training of the TAN networks is described in section 4.1, followed by a discussion of inference procedures in section 4.2. Finally, the experimental setup and evaluation results using TAN networks for dialogue act decoding are described in sections 4.3 and 4.4 respectively.

4.1 Training of TAN Networks

A TAN network can be fully specified by its topology and CPD at each node. Training therefore encompasses these two aspects, topology learning and parameter learning. The following describes each of them in detail.

4.1.1 Topology Learning

1. Select the top n semantic concepts based on the MI value between the concept C_x and the goal G_u, $MI_{(C_x;G_u)}$, where

$$MI_{(C_x;G_u)} = \sum_{C_x,G_u} P(C_x, G_u) \cdot \log \frac{P(C_x|G_u)}{P(G_u)} \tag{4.1}$$

   The optimal value of n is determined experimentally as will be shown in section 4.4.1.

2. Compute $CMI_{(C_x;C_y|G_u)}$ between each pair of concepts, $x \neq y$.

$$CMI_{(C_x;C_y|G_u)} = \sum_{C_x,C_y,G_u} P(C_x, C_y, G_u) \cdot \log \frac{P(C_x, C_y|G_u)}{P(C_x|G_u)P(C_y|G_u)} \tag{4.2}$$

3. Build a fully connected undirected graph in which the vertices are the concepts $C_1, \cdots, C_n$. Annotate the weight of an edge connecting C_x to C_y by $CMI_{(C_x;C_y|G_u)}$.

4. Build a maximum weighted spanning tree which is a spanning tree of a weighted connected graph such that the sum of the weights of the edges in the tree is maximum.

5. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

6. Construct a TAN model by adding a vertex labelled by G_u and adding an arc from G_u to each C_x.

Figure. 4.2. Procedure for building a TAN network.

The procedure for building a TAN network for a goal G_u is based on the well-known Chow-Liu algorithm [104] except that instead of using the mutual information (MI) between two concepts, conditional mutual information (CMI) between concepts given the goal variable is used. The procedure is summarized in Figure 4.2.

In the TAN approach proposed by Friedman *et al.*, it is assumed that all the attribute nodes (or concept nodes here) are fully connected. This may force dependencies between random variables in the network which are in fact statistically independent and thus degrades TAN network performance. Therefore, such a constraint is relaxed here. There are many ways to relax this constraint. One simple approach is only keeping the dependency

links between concept nodes if the CMI between them is greater than certain threshold. A more formal approach is to add dependency links if they result in an increase of the quality measure of the network topology [105], which is described in Figure 4.3.

1. Compute $CMI_{(C_x;C_y|G_u)}$ between each pair of concepts $\{C_x, C_y\}$, $x \neq y$, add undirected edges between C_x and C_y to a graph, and annotate the weight of the edge by $CMI_{(C_x;C_y|G_u)}$.
2. Build a maximum weighted spanning tree A.
3. Sort the edges in the tree A in descending order according to the weights.
4. Construct a network B by adding the goal vertex G_u and its corresponding concepts vertices $\{C_1, C_2, \cdots, C_n\}$. Also add an arc from G_u to each C_x in the network B.
5. For each sorted edge in A, add it to the network B. Compute the network quality measure score for B given a training data set $\mathcal{D}$, $Q(B, \mathcal{D})$. If this score increases compared to the one computed for the previous network topology, then keep the edge; else, remove this edge.

Figure. 4.3. Procedure of adding dependency links.

Let $\mathcal{D}$ be training data and S be a Bayesian network structure, a quality value that is function of the posteriori probability distribution $P(S|\mathcal{D})$ can be assigned to each network. Such values can then be used to rank Bayesian network structures, this is Bayesian quality measure. Quite a number of methods for Bayesian quality measure have been proposed [106; 107; 108]. The Bayesian Dirichlet (BD) metric [107] has been used here because it has the lowest numerical complexity [105].

$$Q(S^h, \mathcal{D}) = \log \mathrm{P}(\mathrm{S^h}) + \sum_{\mathrm{i=1}}^{\mathrm{n}} [\sum_{\mathrm{j=1}}^{\mathrm{q_i}} [\log \frac{\Gamma(\alpha_{\mathrm{ij}})}{\Gamma(\alpha_{\mathrm{ij}} + \mathrm{N_{ij}})} + \sum_{\mathrm{k=1}}^{\mathrm{r_i}} \log \frac{\Gamma(\alpha_{\mathrm{ijk}} + \mathrm{N_{ijk}})}{\Gamma(\alpha_{\mathrm{ijk}})}]] \tag{4.3}$$

where

- S^h denotes the hypothesis that the training data $\mathcal{D}$ is generated by network structure S;
- n is the total number of training cases, q_i is the number of parent configurations of variable C_i and r_i is the number of configurations of variable C_i, i.e. $q_i = \prod_{C_j \in \Pi_{C_j}} r_j$ where Π_{C_i} denotes the parents of C_j;

- N_{ijk} is the number of cases in $\mathcal{D}$ where the random variable C_i is in configuration k and its parents, Π_{C_i}, are in configuration j, N_{ij} is the number of cases in $\mathcal{D}$ where parents of the random variable C_i are in configuration j, i.e. $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$;

- α_{ijk} is the parameter of the associated Dirichlet prior of θ_{ijk}, the distribution of the variable C_i in configuration k and its parents, Π_{C_i} in configuration j, $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ and $\alpha_{ijk} > 0$;

- Γ is the gamma function

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \mathrm{dt} \tag{4.4}$$

  Lonczos' approximation of function $\log \Gamma(z)$ [109] is used in the implementation here.

In practice, the prior network structure probability $P(S^h)$ is unknown and assumed to be constant over all possible network structures S^h.

As mentioned in section 2.3.1.1, Meng's work used the minimum description length (MDL) metric to learn the network topology. The MDL scoring function is in fact an approximation of the BD metric. As the number of training cases tends to infinity, MDL would give the same score as the BD metric, assuming Dirichlet distribution with uniform priors on network structures [110].

4.1.2 Parameter Learning

Consider a finite set $\mathbf{U} = C_1, \cdots, C_n, G$, where the variables $C_1, \cdots, C_n$ are the semantic concepts and G is the goal variable. Formally, a Bayesian network for $\mathbf{U}$ is a pair $B = \langle S, \Theta \rangle$, where S is a *directed acyclic graph (DAG)* (such as figure 4.1) whose vertices correspond to the variables, $C_1, \cdots, C_n, G$, and whose edges represent direct dependencies between the variables, Θ represents the set of parameters that quantify the network. It contains a parameter $\theta_{c_i|\Pi_{c_i}} = P_B(c_i|\Pi_{c_i})$ for each possible value c_i of C_i and Π_{c_i} of Π_{C_i}, where Π_{C_i} denotes the set of parents of C_i in $\mathbf{U}$. Since the goal variable G is the root, $\Pi_G = \oslash$. A Bayesian network B defines a unique joint probability distribution over $\mathbf{U}$ given by

$$P_B(C_1, \cdots, C_n, G) = P(G) \prod_{i=1}^{n} P_B(C_i|\Pi_{C_i}) = P(G) \prod_{i=1}^{n} \theta_{C_i|\Pi_{C_i}} \tag{4.5}$$

The TAN parameters can be estimated as $\theta(c|\Pi_c) = \hat{P_{\mathcal{D}}}(c|\Pi_c)$ where $\hat{P_{\mathcal{D}}}(\cdot)$ denotes the *empirical distribution* defined by frequencies of events in a training data set $\mathcal{D}$, namely, $\hat{P_{\mathcal{D}}}(A) = \frac{1}{N} \sum_j \delta_A(\mathbf{u}_j)$ for each event $A \subseteq \mathbf{u}$, where $\delta_A(\mathbf{u}) = 1$ if $\mathbf{u} \in A$ and $\delta_A(\mathbf{u}) = 0$ otherwise. Here, N is the total number of training instances. It has been shown in [18] that

the above estimation leads to "overfitting" the model and is unreliable especially in small data sets due to many unseen events. Friedman *et al.* proposed to estimate parameters by incorporating the Dirichlet probability distribution which in fact performs a smooth operation [111]. In the context of learning Bayesian networks, a different Dirichlet prior may be used for each distribution of a random variable C_i given a particular value of its parents. Thus,

$$\theta^s(c|\Pi_c) = \frac{N \cdot \hat{P}_{\mathcal{D}}(\Pi_c)}{N \cdot \hat{P}_{\mathcal{D}}(\Pi_c) + N^0_{c|\Pi_c}} \cdot \hat{P}_{\mathcal{D}}(c|\Pi_c) + \frac{N^0_{c|\Pi_c}}{N \cdot \hat{P}_{\mathcal{D}}(\Pi_c) + N^0_{c|\Pi_c}} \cdot \hat{P}_{\mathcal{D}}(c) \tag{4.6}$$

where N is the total number of training instances, $N^0_{c|\Pi_c}$ is the confidence associated with the prior estimate of $P(c|\Pi_c)$. This application of Dirichlet priors mainly affects the estimation in those parts of the conditional probability table that are rarely seen in the training data. In the application here, $N^0_{c|\Pi_c}$ is set to be the same for all probability distributions and its optimal value is determined experimentally.

4.2 Inference in TAN networks

A method called *Probability Propagation in Trees of Clusters (PPTC)* [112; 113] is used to compute the probability $P(G_u = g|\mathbf{C} = \mathbf{c})$ where g is a value of a variable G_u and $\mathbf{c}$ is an assignment of values to a set of variable $\mathbf{C}$. PPTC works in two steps, first converting a TAN network into a secondary structure called a junction tree, then computing the probabilities of interest by operating on the junction tree.

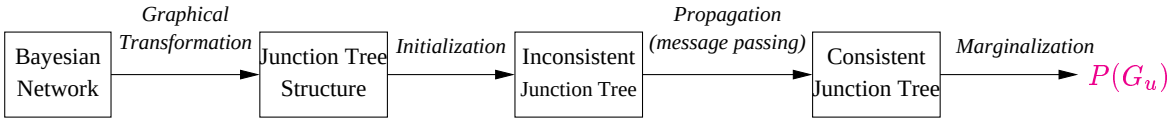


Figure 4.4: Procedure of the PPTC algorithm.

Figure 4.4 gives the overall procedure of the PPTC algorithm. It contains the following steps:

1. *Graphical Transformation.* Transform the *directed acyclic graph (DAG)* of a Baysian network into a junction tree.

2. *Initialization.* Quantify the junction tree with belief potentials, where the potential $\Phi_{\mathbf{C}}$ of a set of variables $\mathbf{C}$ is defined as a function that maps each instantiation $\mathbf{c}$ into a nonnegative real number $\Phi_{\mathbf{C}}(\mathbf{c}) \rightarrow \mathcal{R}$. The result is an inconsistent junction tree.

3. *Global Propagation.* Perform message passing on the junction tree potentials such that all the potentials are locally consistent, which results in a consistent junction tree.

4. *Marginalization.* From the consistent junction tree, compute $P(G_u)$ for each goal G_u.

Each of the above steps are explained in detail in Appendix A.

4.3 Experimental Setup

Experiments have been conducted on both the ATIS corpus and the DARPA Communicator Travel Data. Two sets of TAN networks were therefore trained on them separately. The trained TAN networks are stored in model definition files as illustrated in Appendix B.

4.3.1 ATIS

For the ATIS corpus, altogether 4978 training utterances were selected from the Class A (context-independent) training data in the ATIS-2 and ATIS-3 corpora and both the ATIS-3 NOV93 and DEC94 datasets were used as the test sets. Each training utterance contains a goal and a set of semantic concepts. For example,

```
show flights between toronto and san francisco
Goal    : FLIGHT
Concepts: FLIGHT FROMLOC.CITY_NAME TOLOC.CITY_NAME
```

Performance was measured in terms of goal detection accuracy, which is calculated as the number of correctly detected goals divided by the total number of utterance goals. The semantic concepts of each utterance were derived semi-automatically from the SQL queries provided in ATIS. There are in total 68 distinct concepts. 16 goals were defined by enumerating the key attribute to be fetched from each SQL query, which are listed in Table 4.1.

4.3.2 DARPA Communicator Travel Data

Similar to what has been described in section 3.5.2.2, the DARPA Communicator training set contains 12702 utterances while the test set consists of 1178 utterances. The Phoenix parse result for each utterance was transformed into a frame structure which consists of a goal and a number of semantic concepts and these frames were then manually checked and corrected. There are in total 105 distinct concepts. The Phoenix parser defines only

six topics or goals for the DARPA Communiator data: *Air, Car, Hotel, Query, Respond, Return*. The goal *Air* is rather too broad so this was further divided into sub-classes such as *Date_Time, Location, Origin, Destination, Depart_Time, Return_Time* etc. After this refinement, 16 goals in total were defined which are listed in Table 4.1.

Data Set	*Goals*
ATIS	abbreviation, aircraft, airfare, airline, airport, airport_distance, city, class_service, flight, flight_distance, flight_no, ground_service, location, meal, quantity, restriction
DARPA	air, airline, airport, arrive_time, car, city, date_time, depart, depart_time, destination, fare, flight_time, hotel, origin, return_time, stop

Table 4.1: Goals defined for ATIS and DARPA Communicator Data.

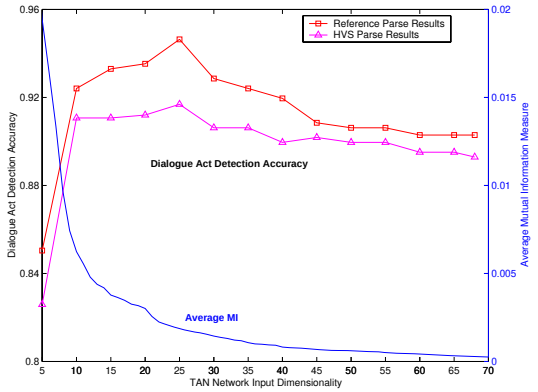
4.4 Experimental Results

This section presents experimental results for both the ATIS and DARPA Communicator corpora using TAN networks for dialogue act decoding. Naive Bayes networks and neural networks have also been trained and evaluated on the same data to provide baselines for performance comparison.

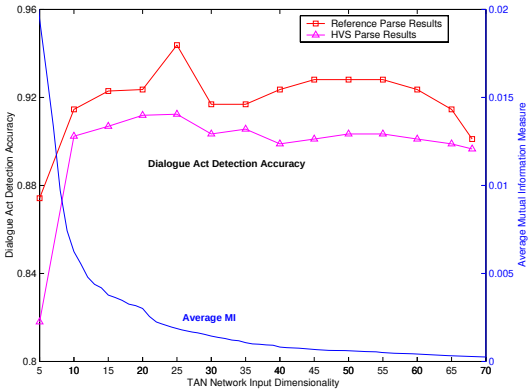
4.4.1 Optimal Number of Semantic Concepts

As noted earlier in section 4.1, the input concepts to the TAN networks are selected based on the MI value between each concept and the goal of the corresponding TAN network. The number of input concepts was determined experimentally by varying the number of concepts to be selected for each goal. Figure 4.5 shows the dialogue act detection accuracy versus the number of semantic concepts selected as the input to the TAN networks for both reference parse results and HVS model parse results. The semantic concepts extracted from the reference parse results are assumed to be all correct. The HVS parse results were generated from the HVS parsing model described in Chapter 3. The parsing errors which is defined as 1-F-measure are 9.7% and 8.1% for the ATIS NOV93 and DEC94 test sets respectively. Not surprisingly, the dialogue act detection accuracy on the reference parse results is always better than that obtained from the HVS parse results. It can be observed from Figure 4.5 (a) and (b) that the optimal number of concepts is 25 for both NOV93 and DEC94 test sets. Increasing the number of concepts degrades the system performance due to overtraining. The average MI value converges to zero when the number of input concepts is 25 or greater.

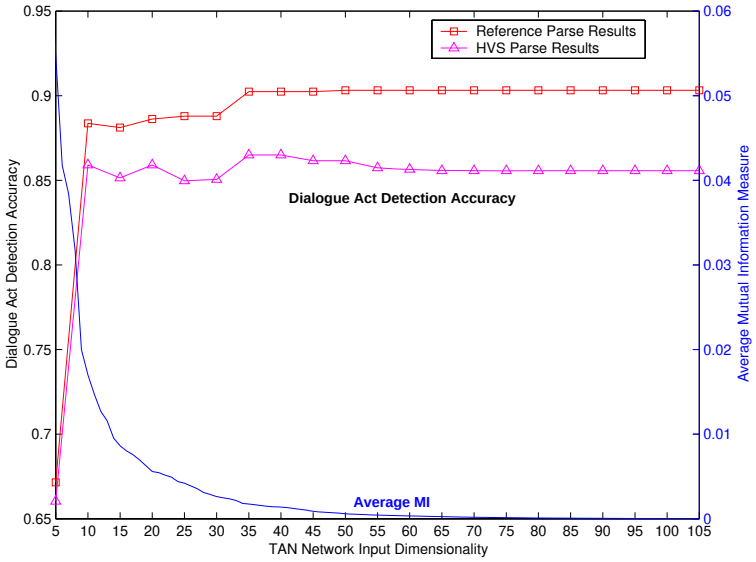
For the DARPA Communicator data, the HVS parsing error is 11.9%. Figure 4.5 (c) shows that goal detection accuracy increases with increased TAN input dimensionality but



(a) ATIS NOV93 Test Set



(b) ATIS DEC94 Test Set



(c) DARPA Communicator Data

Figure 4.5: DA Detection Accuracy vs TAN Input Dimensionality.

then flattens off after the top 35 concepts have been included. Figure 4.5 (c) also shows that the average MI value converges to zero when the number of input concepts is 35 or greater. Hence, this number was chosen for all of the experiments reported subsequently on the DARPA Communicator data.

4.4.2 Optimal Value of Dirichlet Prior

Another setting to be determined is the value of the Dirichlet prior, i.e. the term $N^0_{c|\Pi_c}$ in Equation 4.6. As the same Dirichlet prior will be used through all probabilistic distributions, the subscript of $N^0_{c|\Pi_c}$ may be dropped and the Dirichlet prior can be simply denoted as N^0. A number of different values have been tried and the dialogue act detection accuracy vs different settings of the Dirichlet prior is plotted in Figure 4.6. It can be observed that the best performance is obtained when N^0 is set to 5 for both the ATIS NOV93 and DEC94 test sets. For the DARPA Communicator data, the choice of N^0 did not affect the dialogue act detection accuracy much. One main reason is that the total number of training instances in the Communicator data is 12702, which is much larger than the number of training cases used in ATIS, 4978. When the number of training instances is large, the bias caused by the Dirichlet prior would essentially disappear. Recall from section 4.1.2 that the application of Dirichlet prior mainly affects the estimation of the events that are rarely seen in the training data. Nevertheless, the detection accuracy at N^0 set to 10 is slightly better than that obtained with the smaller values of N^0, therefore, this value is chosen for all the subsequent experiments.

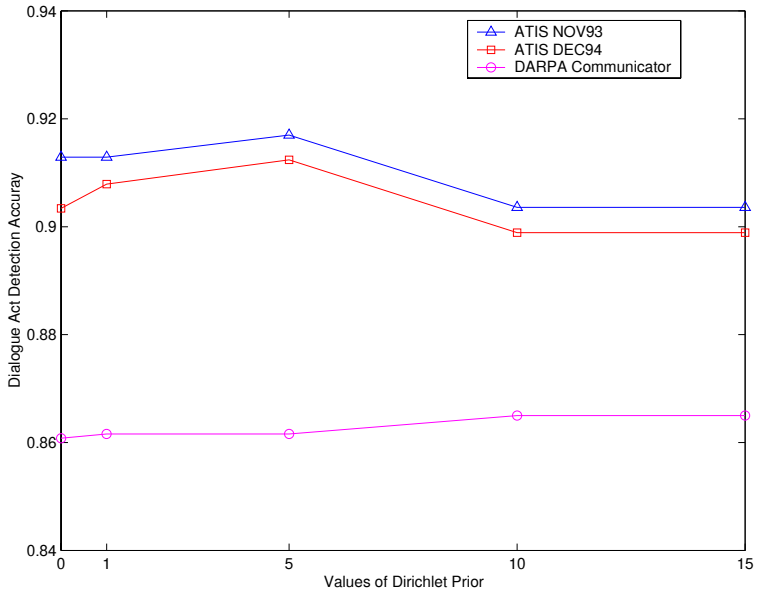


Figure 4.6: DA Detection Accuracy vs Dirichlet Prior Value.

4.4.3 Comparison with Naive Bayes

TAN networks without dependencies between concepts are just Naive Bayes networks. Experiments were conducted to observe the effect of adding dependency links between semantic concepts. Figure 4.7 shows the TAN errors versus the Naive Bayes errors by varying the input dimensionalities of the networks. Each point represents a network input dimension setting, with its x coordinate being the goal detection error using the TAN networks and y coordinate being the goal detection error according to the Naive Bayes networks. The diagonal line in the figure denotes the locations where TAN and Naive Bayes produce the same errors. Thus points above the diagonal line mean TAN outperforms Naive Bayes. It can be observed that most points are above the diagonal line in Figure 4.7 (a) and (b). That is, under most circumstances, TAN outperforms Naive Bayes for both the NOV93 and DEC94 test sets.

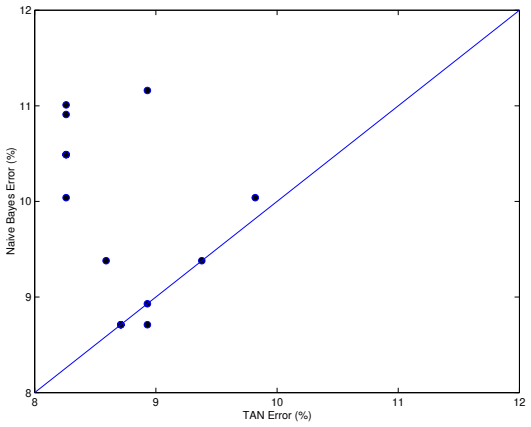
Similar experiments were also conducted on the DARPA Communicator data. Figure 4.7 (c) shows that all the dots are above the diagonal line which means that TAN always outperforms Naive Bayes for the DARPA Communicator test set.

4.4.4 Comparison with Fully-Connected TAN Networks

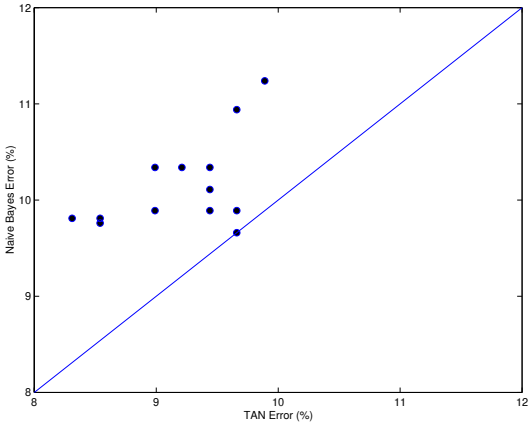
As mentioned earlier, all the attribute nodes in the TAN networks proposed in [18] are fully connected, such a constraint is relaxed in the work here where not all concept pairs have dependency links between them. Experiments have been conducted to compare the performance of the TAN networks built according to the procedures proposed in section 4.1.1 with the fully-connected TAN networks as in [18]. Figure 4.8 shows that generally the TAN networks without full-connected dependency links between concept nodes perform better than the fully-connected TAN networks for both the ATIS and DARPA Communicator test sets. This confirms that some concept nodes are indeed only weakly coupled and the dependency links between them should be removed to achieve better goal detection accuracy.

4.4.5 Comparison with Neural Networks

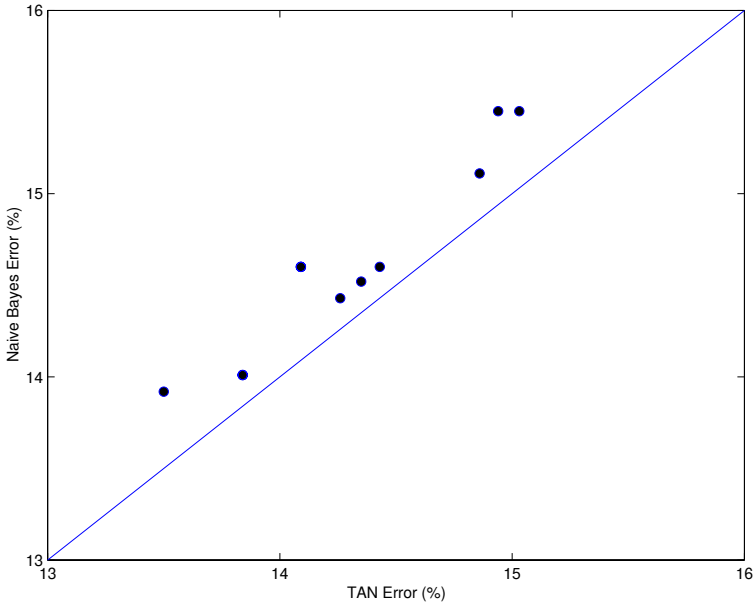
Neural networks have been applied successfully in acoustic modelling [114], language modelling [115], and language understanding [116]. From the perspective of pattern recognition, neural networks can be regarded as an extension of Bayesian inference. Indeed, Bayesian methods have been applied to neural networks learning [117]. Thus, neural networks and Bayesian networks share certain similarities. It is therefore interesting to compare their performance when applied in dialogue act detection.



(a) ATIS NOV93 Test Set

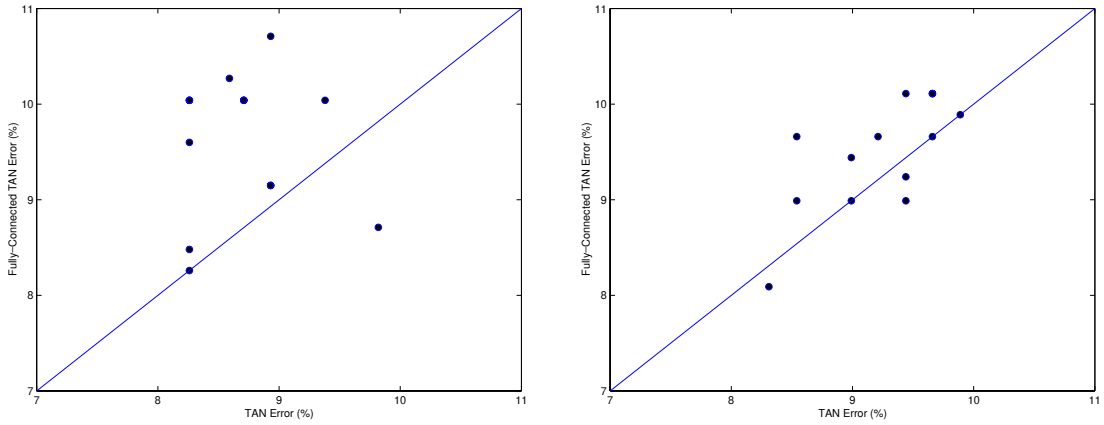


(b) ATIS DEC94 Test Set



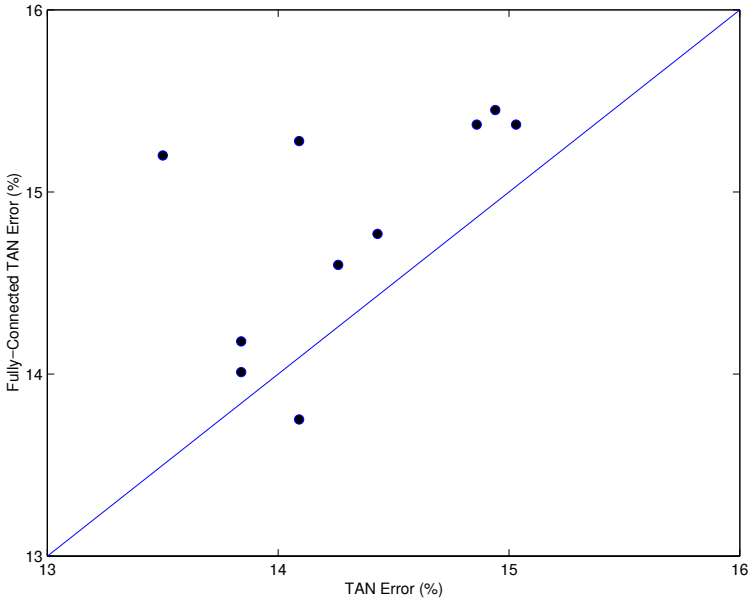
(c) DARPA Communicator Data

Figure 4.7: Comparison of TAN vs Naive Bayes on DA Detection Error by Varying Network Input Dimensionality (Section 4.4.3).



(a) ATIS NOV93 Test Set

(b) ATIS DEC94 Test Set



(c) DARPA Communicator Data

Figure 4.8: Comparison of TAN vs Fully-Connected TAN on DA Detection Error by Varying Network Input Dimensionality (Section 4.4.4).

In the experimental work reported here, dialogue act detection was performed by a single multilayer perceptron (MLP) neural network. This is in contrast to the setup used with Bayesian networks where an individual network was used for each possible goal. The number of input units to the MLP was determined by the number of distinct semantic concepts whilst the number of output units was set by the number of dialogue acts or goals defined. For example, there are altogether 110 semantic concepts and 16 dialogue acts defined for the ATIS task, therefore, the number of input and output units to the MLP is 110 and 16 respectively. An example of an MLP network is shown in Figure 4.9 where σ denotes a sigmoid function.

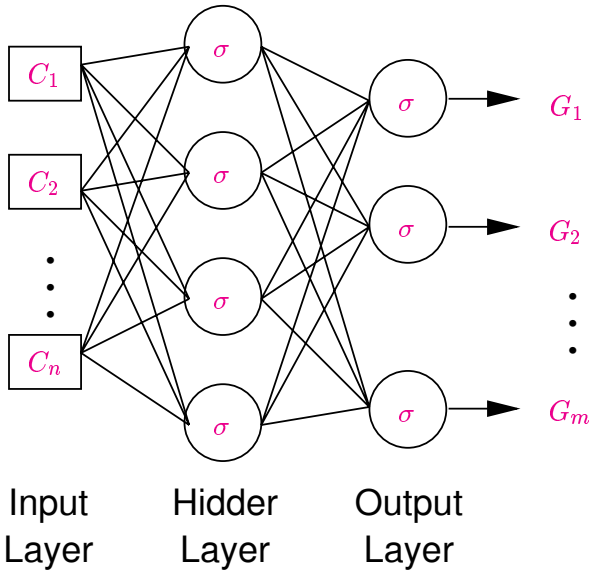


Figure 4.9: Example of a Multilayer Perceptron Neural Network.

For any extracted semantic concept sequence, the appearance of a semantic concept will set its corresponding input unit to be 1. For those semantic concepts that do not appear in the sequence, their input units will remain at 0. Training of the MLP network was conducted using the neural network software package, Stuttgart Neural Network Simulator (SNNS) [118] which supports a variety of learning algorithms. The original training data was divided into two parts, 90% of the data forms the training set while the remaining 10% forms the validation set. Various network topologies were tried such as one hidden layer with different numbers of units (e.g. 2, 4, 8, 16, 32, 64), or two hidden layers with different combinations of units in each layer (e.g. 2-2, 4-2, 4-4, 8-2, 8-4 etc). Learning algorithms tested include the backpropagation algorithm with or without momentum term and the quickprop algorithm. Learning parameters such as the learning rate and the momentum term were tuned to give the best configuration. For each training scenario, the validation data was used to stop the learning process if its minimum square error (MSE) saturated.

It was found that for ATIS the best results on the validation data were obtained using an MLP with one hidden layer of 16 units, trained using the standard backpropagation algorithm with the learning rate 0.2. For the DARPA Communicator data, an MLP with one hidden layer of 8 units, trained using backpropagation with momentum term, with learning rate 0.3 and momentum term 0.3 gave the best results on the validation data.

The dialogue act detection error of the MLP networks when tested on the ATIS NOV93, DEC94 test sets and the DARPA Communicator data test set is shown in Table 4.2. For comparison purposes, the test results on the same test sets using Naive Bayes, fully-connected TAN and TAN networks are also given. It can be observed that the MLP performs worse than the TAN for both the ATIS test sets. However, it gives the same result as the TAN for the DARPA Communicator test set.

Algorithm	*ATIS NOV93*	*ATIS DEC94*	*DARPA Test Data*
Naive Bayes	91.1%	90.2%	86.1%
Fully-Connected TAN	90.9%	91.4%	83.9%
MLP	88.1%	86.9%	86.5%
TAN	91.7%	91.2%	86.5%

Table 4.2: Dialogue Act Detection Accuracy using different algorithms.

4.4.6 Additional Context for DA Detection

All the above experiments on goal detection were based solely on the extracted semantic concepts of the current utterance without considering the historical context of the dialogues. The DARPA Communicator data consists of a set of dialogues within which each utterance is a part of one conversation. Experiments were therefore conducted on the DARPA Communicator test set to measure the impact of including the historical context. Table 4.3 gives the results of including the semantic concepts of the immediately preceeding utterance, the previously detected goal or both. It can be observed that by including the semantic concepts or both goal and concepts from the previous utterance, more noise seems to have been introduced and thus goal detection accuracy degrades. This is partly due to the propagation of the previous semantic tagging errors. The analysis done here was limited to only the user side of the dialogue. It is likely that larger improvements in goal detection accuracy could be gained if the Bayesian networks were conditioned on the previous system output but this information was unforturnately not accessible for the data sets used here.

Context	*HVS Parse Results*	*Reference Parse Results*
Current concepts	86.5%	90.2%
+ prev. concepts	83.5%	86.8%
+ prev. goal	86.3%	90.0%
+ both	83.4%	86.6%

Table 4.3: Goal Detection Accuracy based on Various Contexts.

4.5 Conclusion

This chapter has discussed dialogue act detection using Tree-Augmented Naive Bayes networks (TANs), which are a natural extension to Naive Bayes networks achieved by adding dependencies between concepts or attributes which serve as inputs to the networks. Experiments have been conducted on the ATIS and DARPA Communicator corpora. Dialogue act detection accuracy rates of 91.7% and 91.2% were obtained for the ATIS-3 NOV93 and DEC94 test sets respectively, which are better than the earlier results obtained using the similar Bayesian network methods for goal detection where 87.3% accuracy was obtained for the ATIS NOV93 test set [1] [74].

Peformance has been compared on the effectiveness of concept dependencies in TAN and Naive Bayes networks. It has been found that simple models such as Naive Bayes classifiers perform surprisingly well on both the ATIS and the Communicator data. Capturing dependencies between semantic concepts improves dialogue act detection accuracy slightly. However, fully-connected TAN networks gave worse detection accuracy as more noise was introduced to the networks. This is more significant for the Communicator data. The above suggests that for efficiency purposes, especially in real-time spoken dialogue applications, Naive Bayes networks would be good enough to give acceptable dialogue act detection accuracy.

A neural network approach, multilayer perceptron (MLP) neural network, has also been investigated in this thesis where a composite model was used for all the dialogue acts defined, which is in contrast to Bayesian network approaches where individual network was used for each dialogue act. Though both approaches aim to create models which are well-matched to the data, they seem to occupy opposite extremes of data modelling spectrum. The idea behind MLP training is to find a single set of weights for the network that maximize the fit to the training data, perhaps modified by some sort of weight

[1] It was claimed in [74] that 91.5% goal detection accuracy was obtained for the NOV93 test set. However, the way that goal detection accuracy was calculated was slightly different in that the utterances with out-of-domain goals in the test set were excluded whereas they were included in the calculation of the goal detection accuracy reported here. Besides, there are only 11 goals defined in Meng's work, while the total number of goals defined in the work reported here is 16 for the ATIS task.

penalty to prevent overfitting. It can be interpreted as variations to maximum likelihood estimation. The individual relations between the input variables and the output variables are not developed by engineering judgment so that the model tends to be a black box or input/output table without analytical basis. Bayesian network learning is based on a different view in which probability is used to represent uncertainty about the relationship being learned. Prior knowledge about what the true relationship might be can be expresssed in a probability distribution over the network weights that define this relationship. After presenting training data to the Bayesian networks, probabilities are redistributed in the form of posterior distributions over network weights. Therefore, unlike neural networks which are prone to overfitting, Bayesian networks automatically suppress the tendency to discover spurious structure in data [117]. Experimental results show that if there exist a large amount of training data, the neural network approach may give the same results as the Bayesian network approaches.

Finally, it would be interesting to speculate on performance with more complex tasks. The experimental results reported in this chapter only focussed on individual utterance dialogue act detection. It might be possible to incoporate top level dialogue information such as context of the previous utterance, discourse information, intonation etc, and predict topics of the whole dialogue session. But that would require the availability of history context.

Chapter 5

Integrated Spoken Language Understanding System

This chapter presents a purely data-driven spoken language understanding (SLU) system. It consists of three major components, a speech recognizer, a semantic parser, and a dialog act decoder. Section 5.1 briefly describes the general framework of a statistical SLU system and Section 5.2 summarizes the training and evaluation procedures used. The experimental setup and evaluation results are then presented in section 5.3. Section 5.4 discusses in detail robustness issues of the SLU system and two aspects of SLU system performance are investigated: noise robustness and adaptability to different applications. Finally, section 5.5 concludes this chapter.

5.1 Spoken Language Understanding

Spoken language understanding (SLU) can be broadly viewed as a pattern recognition problem. It aims to interpret the meanings of users' utterances and respond reasonably to what users have said. A typical architecture of an SLU system is given in Figure 5.1, which consists of a speech recognizer, a semantic parser, and a dialog act decoder. The user's input acoustic signal A is first translated into a word string W by the speech recognizer. Such word strings are then mapped into a set of semantic concepts C by the semantic parser. The dialog act decoder infers the user's intention or goals G_u based on the semantic concepts extracted and the current dialogue context. Finally, the deduced information may be passed to the dialogue manager to decide appropriate actions to take in response to the user's query.

Within a statistical framework, the SLU problem can be factored into three stages. First recognize the underlying word string W from each input acoustic signal A, i.e.

$$\hat{W} = \underset{W}{\operatorname{argmax}}\ P(W|A) = \underset{W}{\operatorname{argmax}}\ P(A|W)P(W) \tag{5.1}$$

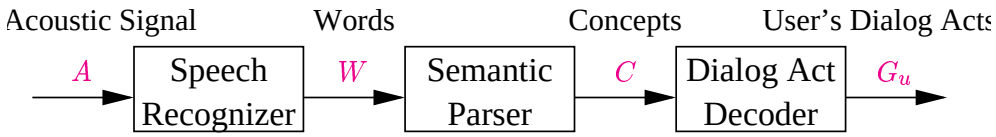


Figure 5.1: Typical structure of a spoken language understanding system.

then map the recognized word string $\hat{W}$ into a set of semantic concepts C

$$\hat{C} = \underset{C}{\text{argmax}}\ P(C|\hat{W}) \tag{5.2}$$

and finally determine the user's dialog acts or goals by solving

$$\hat{G_u} = \underset{G_u}{\text{argmax}}\ P(G_u|\hat{C}) \tag{5.3}$$

In the system described here, each of these stages is modelled separately. A standard HTK-based [19] Hidden Markov Model (HMM) recognizer is used for recognition, the Hidden Vector State (HVS) model for semantic parsing [119], and Tree-Augmented Naive Bayes networks (TAN) [18] for dialog act decoding. The speech recognizer was built using the HTK toolkit [19]. It uses 14 component Gaussian mixture state-clustered cross-word triphone HMMs augmented by using heteroscedastic linear discriminant analysis (HLDA) [120]. Incremental speaker adaptation based on the maximum likelihood linear regression (MLLR) method [93] was performed during the test with updating being performed in batches of five utterances per speaker. The semantic parser component was built using the *Hidden Vector State (HVS)* model [119] as described in Chapter 3. The dialog act decoder was implemented using the Tree-Augmented Naive Bayes (TAN) algorithm [18] as described in Chapter 4.

It should, however, be noted that sequential decoding is suboptimal in the sense that the solution of each stage depends on the exact solution of the previous stage. In order to reduce the effect of this approximation, it is possible to retain a word lattice or N-best word hypotheses instead of the single best string $\hat{W}$ as the output of the speech recognizer. The semantic parse results may then be incorporated with the output from the speech recognizer to rescore the N-best list since it provides additional knowledge to the recognizer. This is considered further in section 5.3. Similarly, it is possible to retain the N-best parse results from the semantic parser and leave the selection of the best hypothesis until the dialog act decoding stage. However, in practice, no gain was found for this and hence it is not pursued further here.

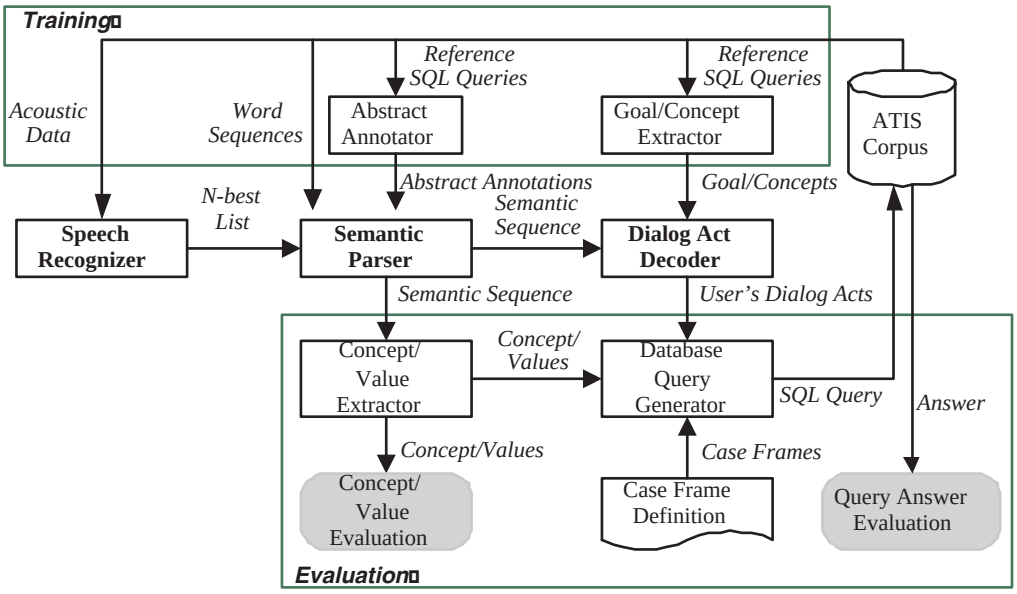


Figure 5.2: Procedures on ATIS training and evaluation.

5.2 System Training and Evaluation

Figure 5.2 shows the organization of the SLS system for both training and evaluation. The ATIS training data contain the acoustic speech signal, word transcription and reference SQL query for each utterance. Each of the three major components, the speech recognizer, the semantic parser, and the dialog act decoder are trained separately. The acoustic speech signal is modelled by extracting 39 features every 10ms: 12 cepstra, energy, and their first and second derivatives. This data is then used to train the speaker-independent, continuous speech recognizer. The semantic parser is trained using the word transcriptions from the ATIS corpus combined with their abstract semantics extracted automatically from the reference SQL queries provided in the corpus. The parser is trained on this data using constrained EM as described in Chapter 3. It is straightforward to identify the main topic or goal and the key semantic concepts of each utterance from the corresponding reference SQL query and this information is used to train the dialog act decoder.

During testing, the N-best lists from the speech recognizer are passed to the semantic parser to generate semantic concept sequences. Parse scores from the semantic parser are combined with the total acoustic and language model likelihoods from the speech recognizer and used to rescore the N-best list. Meaningful semantic concept/value pairs are then extracted from the resulting best hypothesis and the user's goals are inferred by the dialog act decoder from the semantic concept sequences generated. These extracted concept/value pairs and inferred goals are then fed into the SQL query generator to form an SQL query in order to fetch answers from the ATIS database. An example of the

case frame definition file which is used by the query generator to produce SQL queries is illustrated in Appendix C.

Performance is measured at both the component and the system level. For the former, the recognizer is evaluated by word error rate, the parser by concept slot retrieval rate using an F-measure metric [102], and the dialog act decoder by detection rate. The overall system performance is measured using the standard NIST "query answer" rate.

5.3 Experiments

Experiments have been conducted using the ATIS corpus and the ATIS-3 NOV93 and DEC94 evaluation data were selected as test sets. As mentioned earlier, utterances in the ATIS corpus are divided into three categories, context-independent (A), context-dependent (D), or unanswerable (X). The experimental results reported in this chapter focus on category A utterances only unless otherwise specified.

Altogether 22316 spontaneous utterances recorded using Sennheiser microphone from ATIS-2 and ATIS-3 are used for acoustic model training. This includes the ATIS-2 FEB92 and NOV92 test sets in addition to the ATIS-2 and ATIS-3 training sets. The language model was trained on 23096 ATIS spontaneous utterances with vocabulary size 1644. It consists of a word trigram and a word trigram interpolated with a class-based trigram. The latter has 60 classes derived automatically using the Kneser-Ney clustering procedure [121]. The perplexity tested on the joint ATIS-3 NOV93 and DEC94 test sets is 16.5 and 15.5 for the word trigram alone and the interpolated model respectively.

The N-best word hypotheses generated from the speech recognizer were fed into the semantic parser to output semantic concept sequences. Given an acoustic speech signal A, translated into a word sequence W, and parsed into a semantic concept sequence C, the parse scores are combined with the total acoustic and language model likelihoods according to equation 5.4.

$$\begin{aligned} \hat{C}, \hat{W} &\approx \underset{C,W \in L_N}{\operatorname{argmax}} P(A|W)P(W)P(C|W) \\ &\approx \underset{C,W \in L_N}{\operatorname{argmax}} P(A|W)P(W)^{\gamma}P(C|W)^{\alpha} \end{aligned} \tag{5.4}$$

where $P(A|W)$ is the acoustic probability from the first pass, $P(W)$ is the language modelling likelihood, $P(C|W)$ is the semantic parse score, L_N denotes the N-best list, α is a semantic parse scale factor and γ is a grammar scale factor.

For the dialog act decoder, 16 dialog acts or goals were defined in the ATIS domain with each goal corresponding to one TAN. The top 25 semantic concepts ranked by MI were used as input to each TAN.

The SQL query generator module was tested on the reference parse results of ATIS-3 NOV93 and DEC94 test sets. 5 out of 448 utterances from NOV93 test set and 3 out of 445 utterances from DEC94 test set did not return the correct answers, which gives the utterance understanding error rate 1.1% and 0.7% respectively. The analysis of the results shows that one context-dependent utterance has been misclassified as category A (context-independent) in each of these two test sets and the rest are too complicated for the SQL query generator to handle properly. Even though these eight utterances failed to return correct answers, they were however retained in all the subsequent experiments.

5.3.1 Performance of Individual Components

Experiments were first conducted to evaluate individual components of the SLU system. Table 5.1 gives the results in word error rate (WER) for the speech recognizer by imposing different refinement techniques on the context-dependent (category A) test sets as well as the full test sets (A+D+X). The baseline was built using a word bigram language model (LM), then the HMM models were refined based on the HLDA technique. Subsequently, during testing, incremental adaptation was performed and bigram word lattices were generated, which were then expanded to word trigram lattices by applying the word trigram LM. Alternatively, the class-based and the word trigram LM was used to transform word bigram lattices to interpolated word/class trigram lattices.

	Category A		*Category A+D+X*	
Criteria	*NOV93*	*DEC94*	*NOV93*	*DEC94*
word bigram	6.1	5.5	7.3	6.0
+HLDA	5.8	4.9	6.8	5.4
+adaptation	4.9	4.3	5.7	4.1
+word trigram	4.1	3.1	4.8	3.6
+class-based trigram	4.1	3.0	4.8	3.4

Table 5.1: Test results for the speech recognizer (%WER).

The semantic parser was tested using both text input (reference transcriptions) and spoken input (recognizer output). The F-measure scores together with recall and precision values are reported in Table 5.2.

	NOV93		*DEC94*	
Measurement	*Text*	*Spoken*	*Text*	*Spoken*
Recall	89.2%	87.6%	91.3%	89.7%
Precision	91.4%	90.4%	92.6%	91.4%
F-measure	90.3%	89.0%	91.9%	90.5%

Table 5.2: Test results for the semantic parser.

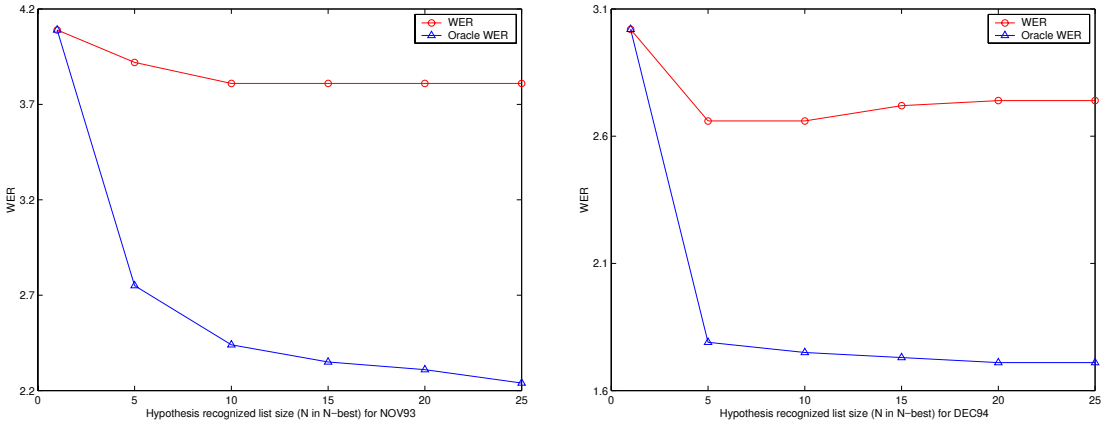
For the dialog act decoder, the goal detection accuracy based on the parse results of both text input and spoken input is shown in Table 5.3.

Parser Input	*NOV93*	*DEC94*
Text input	91.7%	91.2%
Spoken input	91.5%	90.8%

Table 5.3: Test results for the dialog act decoder.

5.3.2 Optimal N-best List

During the integrated system test, experiments were first conducted to determine the best possible performance in WER obtainable from the N-best lists output by the speech recognizer. This was done by picking the hypothesis with the lowest WER from each list for N ranging from 1 to 1000. As the system gave the same performance for all values of N greater than 25, only the results with values of N ranging from 1 to 25 are reported in Figure 5.3. It can be observed that $N = 10$ gives the optimal WER and subsequent experiments were therefore conducted on 10-best lists only. Increasing the value of N degrades the system performance slightly. This is due to noise introduced by the lower ranks of N-best lists. The oracle WER of different N-best lists are also given to indicate the range of improvements possible by incorporating more knowledge sources. It should be mentioned that for each N-best list, various settings of the semantic scale factor α as defined in Equation 5.4 has been tested and it was found that the best performance was obtained when the semantic scale factor was set to 10 for all the N-best lists.



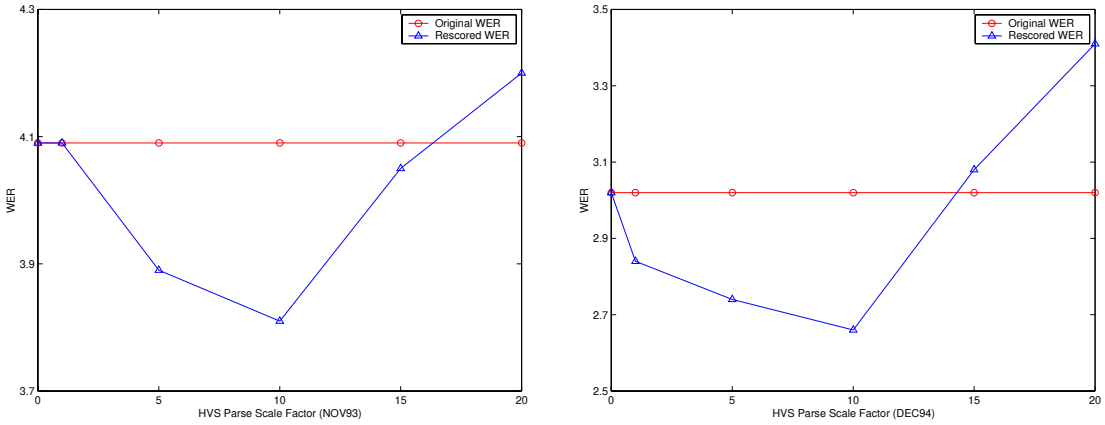
(a) ATIS NOV93 Test Set

(b) ATIS DEC94 Test Set

Figure 5.3: Values of N (as in N-best list) vs WER.

5.3.3 Optimal Semantic Parse Scale Factor

Figure 5.4 shows the WER obtained for rescored 10-best word hypotheses when the semantic parse scale factor α as defined in Equation 5.4 is varied. The optimal value for α is 10 as the lowest WER is obtained at this point for both NOV93 and DEC94 test sets. Increasing α value degrades the system performance since the semantic parse scores tend to dominate the rescored results.



(a) ATIS NOV93 Test Set (b) ATIS DEC94 Test Set

Figure 5.4: Scale of semantic parse score vs WER.

5.3.4 End-to-End Evaluation Results

The end-to-end evaluation results on both natural language understanding (NL) and spoken language understanding (SLS) evaluations are shown in Table 5.4. F-measure evaluates the extraction of concept/value pairs in terms of recall and precision, while answer error rate measures the minimum / maximum answers from the ATIS database using the NIST scoring package. The latter is the standard scoring metric used by DARPA ATIS SLU systems. For the NL test, the reference transcriptions is used as input to the semantic parser instead of the recognized output. The SLS(1) results were obtained by taking the best word hypothesis directly from the speech recognizer, while the SLS(10) results were obtained by taking the best word hypothesis from the rescored 10-best list after incorporating semantic parse scores.

It can be observed from Table 5.4 that by incorporating semantic knowledge via rescoring, the WER has been reduced by 7.3% and 10.0% relative for the NOV93 and DEC94 test set respectively. The relative reduction in answer error rate is 12.0% and 9.4%.

	NOV93			*DEC94*		
	WER	*F-measure*	*Answer Error*	*WER*	*F-measure*	*Answer Error*
NL	-	90.3%	12.3%	-	91.9%	8.5%
SLS(1)	4.1	89.0%	18.3%	3.0	90.5%	13.9%
SLS(10)	3.8	89.3%	16.1%	2.7	90.6%	12.6%

Table 5.4: NOV93 and DEC94 NL and SLS test results.

Figure 5.5 compares the performance of the system built here (denoted as cu) with the systems developed by the DARPA ATIS programme participants. The upper portion of Figure 5.5 gives the NL answer error rates of various systems on the NOV93 and DEC94 test sets, while the lower portion of the figure gives the SLS answer error rates on the same test sets. It can be observed that the SLU system discussed here is indeed comparable to the original DARPA ATIS SLU systems which relied on either hand-crafted semantic grammar rules or fully-annotated training corpora to extract semantic information.

5.4 Robustness Issues

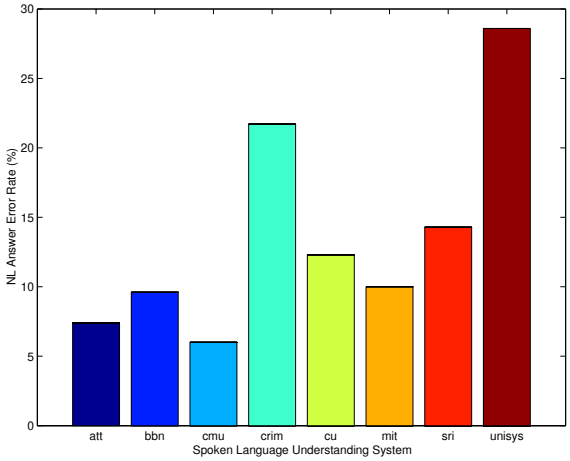
Robustness is a key requirement in spoken language understanding (SLU) systems. Human speech is often ungrammatical and ill-formed, and there will frequently be a mismatch between training and test data. This section discusses robustness and adaptation issues in the SLU system.

Formally speaking, the robustness of language (recognition, parsing, etc.) is a measure of the ability of human speakers to communicate despite incomplete information, ambiguity, and the constant element of surprise [122]. In this section, two aspects of SLU system performance are investigated: noise robustness and adaptability to different applications. For the former, an SLU system should maintain acceptable performance when given noisy input speech data. This requires, the understanding components of the SLU system to be able to correctly interpret the meaning of an utterance even when faced with recognition errors. For the latter, the SLU system should be readily adaptable to a different application using a relatively small set (e.g. less than 100) of adaptation utterances.

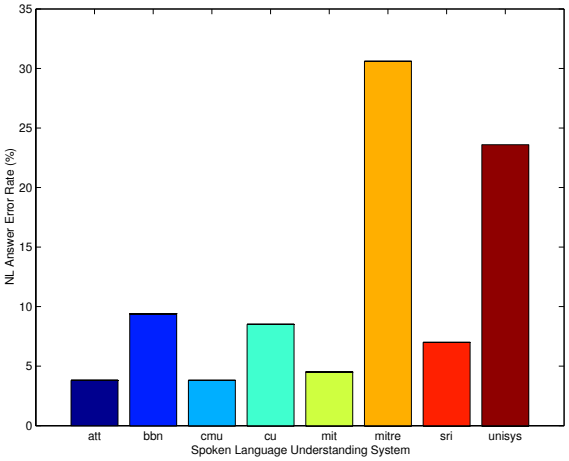
5.4.1 Noise Robustness

To test noise robustness, the system has been tested on data from the ATIS domain which has been artificially corrupted with varying levels of additive car noise from the NOISEX-92 [123] database. In order to obtain different SNRs, the noise was scaled accordingly before adding to the speech signal.

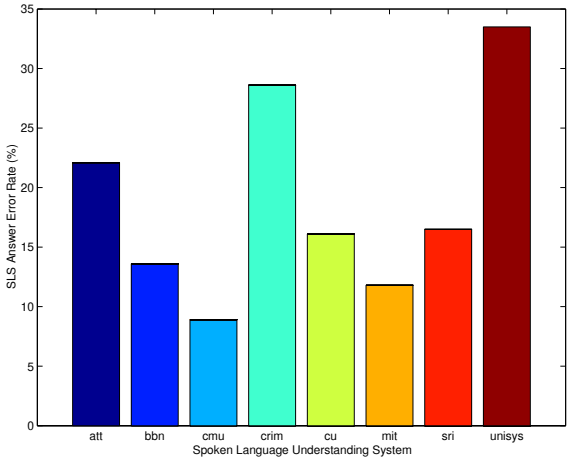
Robust spoken language understanding components should be able to compensate for the weakness of the speech recognizer. That is, ideally they should be capable of generating



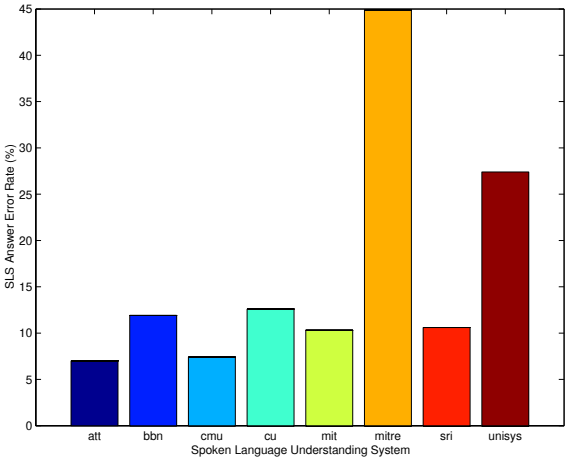
(a) NOV93 NL Answer Error Rate



(c) DEC94 NL Answer Error Rate



(b) NOV93 SLS Answer Error Rate



(d) DEC94 SLS Answer Error Rate

Figure 5.5: SLU Systems Performance Comparison.

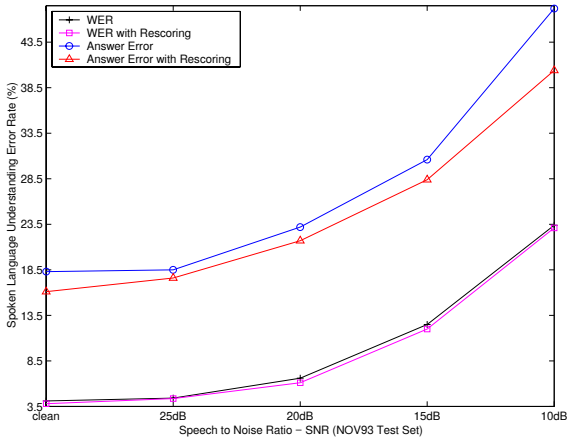
the correct meaning of an utterance even if it is recognized wrongly by a speech recognizer. At minimum, the performance of the understanding components should degrade gracefully as recognition accuracy degrades.

Figure 5.6 gives the system performance on the corrupted test data with additive noise ranging from 25dB to 10dB SNR. The label "clean" in the X-axis denotes the original clean speech data without additive noise. Note that the recognition results on the corrupted test data were obtained directly using the original clean speech HMM models without retraining for the noisy conditions. The upper portion of Figure 5.6 shows the end-to-end performance in terms of query answer error rate for the NOV93 and DEC94 test sets. For easy reference, WER is also shown. The individual component performance, F-measure for the HVS semantic parser and dialogue act (DA) detection accuracy for the DA decoder, are illustrated in the lower portion of Figure 5.6. For each test set, the performance on the rescored word hypotheses is given as well. This incorporates the semantic parse scores into the acoustic and language modelling likelihoods to rescore the 10-best word lists from the speech recognizer.

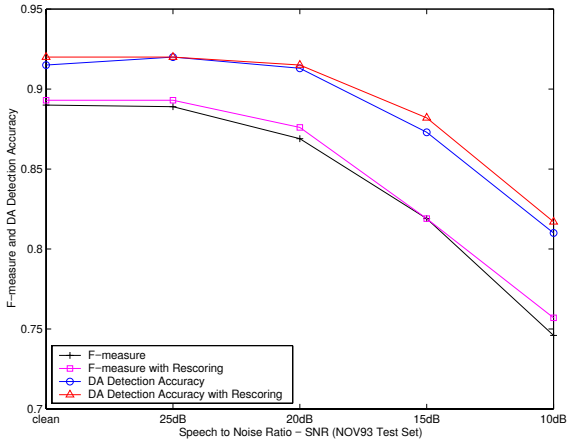
It can be observed that the system gives fairly stable performance at high SNRs and then the recognition accuracy degrades rapidly in the presence of increasing noise. At 20dB SNR, the WER for the NOV93 test set increases by 1.6 times relative to clean whilst the query answer error rate increases by only 1.3 times. On decreasing the SNR to 15dB, the system performance degrades significantly. The WER increases by 3.1 times relative to clean but the query answer error rate increases by only 1.7 times. Similar figures were obtained for the DEC94 test set.

The above suggests that the end-to-end performance measured in terms of answer error rate degrades more slowly compared to the recognizer WER as the noise level increases. This demonstrates that the statistically-based understanding components of the SLU system, the semantic parser and the dialogue act decoder, are relatively robust to degrading recognition performance.

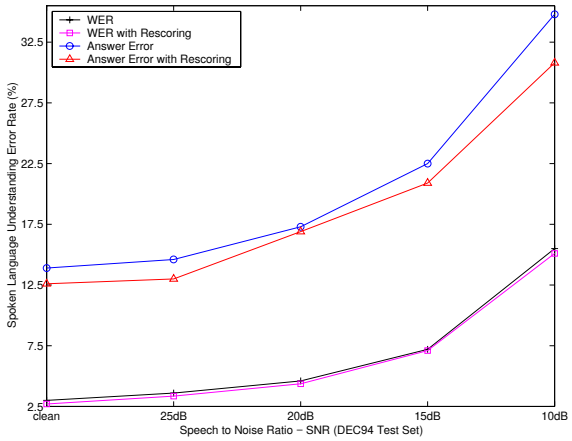
Regarding the individual component performance, the dialogue act detection accuracy appears to be less sensitive to decreasing SNR. This is probably a consequence of the fact that the Bayesian networks are set up to respond to only the presence or absence of semantic concepts or slots, regardless of the actual values assigned to them. In another words, the performance of the dialogue act decoder is not affected by the mis-recognition of individual words, but only by a failure to detect the presence of a semantic concept. It can also be observed from Figure 5.6 that the F-measure needs to be better than 85% in orderto achieve acceptable end-to-end performance.



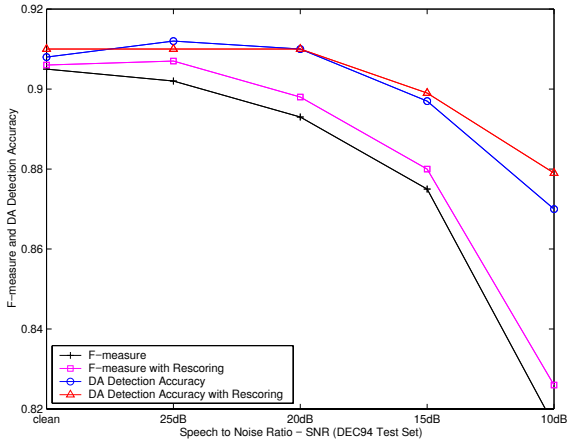
(a) NOV93 End-to-End Performance



(b) NOV93 Component Performance



(c) DEC94 End-to-End Performance



(d) DEC94 Component Performance

Figure 5.6: SLU system performance vs SNR.

5.4.2 Adaptation to New Applications

As discussed in Section 2.4, MAP adaptation and non-linear interpolation have been applied to HVS parser model adaptation. The following discusses each of these two approaches in detail and presents the experimental results in the end.

5.4.2.1 MAP Adaptation

Bayesian adaptation reestimates model parameters directly using adaptation data. It can be implemented via maximum *a posteriori* (MAP) estimation. Assuming that model parameters are denoted by Θ, then given observation samples Y, the MAP estimate is obtained as

$$\Theta_{MAP} = \underset{\Theta}{\text{argmax}} P(\Theta|Y) = \underset{\Theta}{\text{argmax}} P(Y|\Theta)P(\Theta) \tag{5.5}$$

where $P(Y|\Theta)$ is the likelihood of the adaptation data Y and model parameters Θ are random vectors described by their probabilistic mass function (pmf) $P(\Theta)$, also called the prior distribution.

In the case of HVS model adaptation, the objective is to estimate probabilities of discrete distributions over vector state stack shift operations and output word generation. Assuming that they can be modelled under the multinomial distribution, for mathematical tractability, the conjugate prior, the Dirichlet density, is normally used. Assume a parser model $P(W, C)$ for a word sequence W and semantic concept sequence C exists with J component distributions P_j each of dimension K, then given some adaptation data W_l, the MAP estimate of the kth component of P_j, $\hat{P}_j(k)$, is

$$\hat{P}_j(k) = \frac{\sigma_j}{\sigma_j + \tau}\tilde{P}_j(k) + \frac{\tau}{\sigma_j + \tau}P_j(k) \tag{5.6}$$

where $\sigma_j = \sum_{k=1}^{K} \sigma_j(k)$ in which $\sigma_j(k)$ is defined as the total count of the events associated with the kth component of P_j summed across the decoding of all adaptation utterances W_l, τ is the prior weighting parameter, $P_j(k)$ is the probability of the original unadapted model, and $\tilde{P}_j(k)$ is the empirical distribution of the adaptation data, which is defined as

$$\tilde{P}_j(k) = \frac{\sigma_j(k)}{\sum_{i=1}^{K} \sigma_j(i)} \tag{5.7}$$

As discussed in Chapter 3, the HVS model consists of three types of probabilistic move. The MAP adaptation technique can be applied to the HVS model by adapting each of these three component distributions individually.

5.4.2.2 Log-Linear Interpolation

Log-linear interpolation has been applied to language model adaptation and has been shown to be equivalent to a constrained minimum Kullback-Leibler distance optimisation problem[124].

Following the notation introduced in section 5.4.2.1, where $P_j(k)$ is the probability of the original unadapted model, and $\tilde{P}_j(k)$ is the empirical distribution of the adaptation data, denote the final adapted model probability as $\hat{P}_j(k)$. It is assumed that the Kullback-Leibler distance of the adapted model to the unadapted and empirically determined model is

$$D(\hat{P}_j(k) \parallel P_j(k)) = d_1 \tag{5.8}$$

$$D(\hat{P}_j(k) \parallel \tilde{P}_j(k)) = d_2 \tag{5.9}$$

Given an additional model probability $\bar{P}_j(k)$ whose distance to $\hat{P}_j(k)$ should be kept small, and introducing Lagrange multipliers $\lambda_1^{'}$ and $\lambda_2^{'}$ to ensure that constraints 5.8 and 5.9 are satisfied, yields

$$\mathcal{D} = D(\hat{P}_j(k) \parallel \bar{P}_j(k)) + \lambda_1^{'}(D(\hat{P}_j(k) \parallel P_j(k)) - d_1) + \lambda_2^{'}(D(\hat{P}_j(k) \parallel \tilde{P}_j(k)) - d_2) \tag{5.10}$$

Minimizing $\mathcal{D}$ with respect to $\hat{P}_j(k)$ yields the required distribution.

With some manipulation and redefinition of the Lagrange Multipliers, it can be shown that

$$\hat{P}_j(k) = \frac{1}{Z_\lambda} P_j(k)^{\lambda_1} \tilde{P}_j(k)^{\lambda_2} \tag{5.11}$$

where $\bar{P}_j(k)$ has been assumed to be a uniform distribution which is then absorbed into the normalization term Z_λ.

The computation of Z_λ is very expensive and can usually be dropped without significant loss in performance [125]. For the other parameters, λ_1 and λ_2, the generalized iterative scaling algorithm or the simplex method can be employed to estimate their optimal settings.

5.4.2.3 Experiments

To test the portability of the statistical parser, the initial experiments reported here are focussed on assessing the adaptability of the HVS model when it is tested in a domain which covers broadly similar concepts, but comprises rather different speaking styles. To this end, the flight information subset of the DARPA Communicator Travel task has been used as the target domain [21]. By limiting the test in this way, it is ensured that the

dimensionalities of the HVS model parameters remain the same and no new semantic concepts are introduced by the adaptation training data.

The baseline HVS parser was trained on the ATIS corpus using 4978 utterances selected from the context-independent (Class A) training data in the ATIS-2 and ATIS-3 corpora. The vocabulary size of the ATIS training corpus is 611 and there are altogether 110 semantic concepts defined. The parser model was then adapted using utterances relating to flight reservation from the DARPA Communicator data. Although the latter bears similarities to the ATIS data, it contains utterances of a different style and is often more complex. For example, Communicator contains utterances on multiple flight legs, information which is not available in ATIS.

To compare the adapted ATIS parser with an in-domain Communicator parser, a HVS model was trained from scratch using 10682 Communicator training utterances. The vocabulary size of the in-domain Communicator training data is 505 and a total of 99 semantic concepts have been defined. For all tests, a set of 1017 Communicator test utterances was used.

Table 5.5 lists the recall, precision, and F-measure results obtained when tested on the 1017 utterance DARPA Communicator test set. The baseline is the unadapted HVS parser trained on the ATIS corpus only. The in-domain results are obtained using the HVS parser trained solely on the 10682 DARPA training data. The other rows of the table give the parser performance using MAP and log-linear interpolation based adaptation of the baseline model using 50 randomly selected adaptation utterances.

System	*Recall*	*Precision*	*F-measure*
Baseline	79.81%	87.14%	83.31%
In-domain	87.18%	91.89%	89.47%
MAP	86.74%	91.07%	88.85%
Log-Linear	86.25%	92.35%	89.20%

Table 5.5: Performance comparison of adaptation using MAP or log-linear interpolation.

Since a reference database is not available for the DARPA Communicator task, it is not possible to conduct the end-to-end performance evaluation as in Section 5.4.1. However, the experimental results in Section 5.4.1 indicate that the F-measure needs to exceed 85% to give acceptable end-to-end performance. Therefore, it can be inferred from Table 5.5 that the unadapted ATIS parser model would perform very badly in the new Communicator application whereas the adapted models would give performance close to that of a fully trained in-domain model.

Figure 5.7 shows the parser performance versus the number of adaptation utterances used. It can be observed that when there are only a few adaptation utterances, MAP

adaptation performs significantly better than log-linear interpolation. However above 25 adaptation utterances, the converse is true. The parser performance saturates when the number of adaptation utterances reaches 50 for both techniques and the best performance overall is given by the parser adapted using log-linear interpolation. The performance of both models however degrades when the number of adaptation utterances exceeds 100, possibly due to model overtraining. For this particular application, it can be concluded that just 50 adaptation utterances would be sufficient to adapt the baseline model to give comparable results to the in-domain Communicator model.

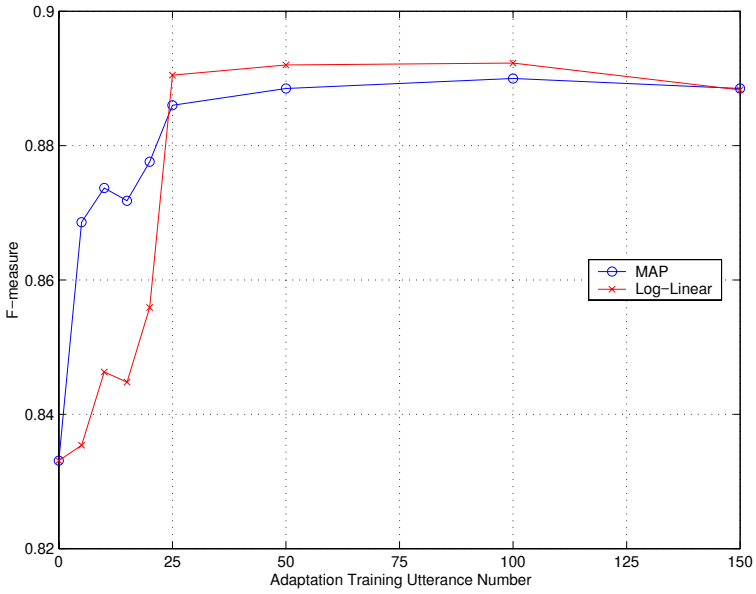


Figure 5.7: F-measure vs amount of adaptation training data.

5.5 Conclusions

This chapter has discussed a purely data-driven spoken language understanding system. Its three major components, the speech recognizer, the semantic parser, and the dialog act decoder, are trained directly from corpus data. In particular, its two understanding components, the semantic parser and the dialog act decoder, are trained without the use of explicit semantic grammar rules or fully-annotated treebank style data.

The evaluation results on the ATIS corpus show that the SLU system discussed here is comparable to the original DARPA ATIS SLU systems which relied on either hand-crafted semantic grammar rules or fully-annotated training corpora to extract semantic information, but it can be built at much lower cost. It has also been confirmed, as done by others [126; 127; 128; 129], that semantic knowledge extracted by a parser can be applied

to rescore N-best word hypotheses from the speech recognizer to improve both WER and overall end-to-end performance.

One of the major claims motivating the design of this data-driven statistical SLU system is that its fully-statistical framework makes it intrinsically robust and readily adaptable to new applications. This claim has been investigated experimentally via two sets of experiments using a system trained on the ATIS corpus.

In the first set of experiments, the acoustic test data was corrupted with varying levels of additive car noise. It was found that although the addition of noise had a substantial effect on the word error rate, its relative influence on both the semantic parser slot/value retrieval rate and the dialogue act detection accuracy was somewhat less. Most importantly of all, there was no catastrophic failure point at which the system effectively stops working, a situation not uncommon in current rule-based systems.

In the second set of experiments, the ability of the semantic decoder component to be adapted to another application was investigated. In order to limit the issues to parameter mismatch problems, the new application chosen (Communicator) covered essentially the same set of concepts but was a rather different corpus with different user speaking styles and different syntactic forms. Overall, it was found that moving a system trained on ATIS to this new application resulted in a 6% absolute drop in F-measure on concept accuracy (i.e. a 62% relative increase in parser error) and by extrapolation with the results in the ATIS domain, it may be inferred that this would make the non-adapted system essentially unusable in the new application. However, when adaptation was applied using only 50 adaptation sentences, the loss of concept accuracy was mostly restored. Specifically, using log-linear adaptation, the out-of-domain F-measure of 83.3% was restored to 89.2% which is close to the in-domain F-measure of 89.5%.

Although these tests are preliminary and are based on off-line corpora, the results do give positive support to the initial claim made for statistically-based spoken language systems, i.e. that they are robust and they are readily adaptable to new or changing applications.

Chapter 6

Conclusions and Future Work

The development of spoken dialogue systems rely heavily on computational linguists to define semantic grammar rules, to produce templates for appropriate response generation, to build the lexicon etc. Reducing the need for manual processing and automating the whole process as much as possible are thus of particular interest. The research work here has focused on statistical learning approaches for building a purely data-driven spoken language understanding (SLU) system whose three major components, the speech recognizer, the semantic parser, and the dialogue act decoder, are trained entirely from data. The system is comparable to existing SLU systems which rely on either hand-crafted semantic grammar rules or statistical models trained on fully-annotated training corpora but it has greatly reduced build cost.

This chapter first summarizes the research work reported in this thesis, then draws conclusions on the research conducted, and finally identifies several potential areas for future work.

6.1 Summary

The research work conducted are summarized as follows:

- Proposed and implemented a hierarchical semantic parser model, *Hidden Vector State (HVS)* model, which is a compromise between the flat-concept model and the fully recursive model. It is motivated by the hypothesis that a suitably constrained hierarchical model may be trainable without treebank data whilst simultaneously retaining sufficient ability to capture the hierarchical structure needed to robustly extract task domain semantics. One possible constraint is to restrict the underlying generation process to be strictly right-branching [130]. Such a constrained hierarchical model can be conveniently implemented using an HVS model which extends

the *flat-concept* HMM model by expanding each state to encode the stack of a push-down automaton. This allows the model to efficiently encode hierarchical context, but because stack operations are highly constrained it avoids the tractability issues associated with full context-free stochastic models such as the hierarchical HMM. As shown in Chapter 3, such a model is indeed trainable using only lightly annotated data and furthermore, it offers considerable performance gains compared to the flat concept model.

- Tree-Augmented Naive Bayes networks (TANs) [18], which are a natural extension to Naive Bayes networks achieved by adding dependencies between concepts or attributes which serve as inputs to the networks, have been successfully applied for classification tasks. This TAN algorithm was slightly modified and used for dialogue act detection in this research work. Inference from TANs was based on the Probability Propagation in Trees of Clusters (PPTC) algorithm [112]. Peformance has been compared on the effectiveness of concept dependencies in TAN, Naive Bayes networks, and fully-connected TAN. It has been found that TAN slightly outperforms the Naive Bayes approach whilst fully-connected TAN performs worse than TAN due to more noise being introduced to networks. To provide a contrast, the use of a multilayer perceptron (MLP) neural network for dialogue act detection has also been investigated in this thesis. In this case, a composite model was used for all the dialogue acts defined, which is in contrast to Bayesian network approaches where an individual network was used for each dialogue act. The MLP performs worse than TAN for the ATIS corpus, but gave similar dialogue act detection accuracy for the DARPA Communicator Data.

- An intergrated spoken language understanding system has been built, which consists of a standard HTK-based [19] Hidden Markov Model (HMM) recognizer for recognition, the HVS model for semantic parsing, and Tree-Augmented Naive Bayes networks (TAN) for dialog act decoding. In order to conduct an end-to-end performance evaluation in a way similar to that used in the DARPA sponsored SLU program for the ATIS task, an SQL query generator has also been implemented to generate SQL queries to fetch answers from the database. Experimental results show that the system is comparable to the original DARPA ATIS SLU systems but with greatly reduced build cost.

- Robustness issues in the SLU system have been discussed. To test robustness, the system has been tested on data from the ATIS domain which has been artificially corrupted with varying levels of additive noise. Although the speech recognition

performance degraded steadily, the system did not fail catastrophically. Indeed, the rate at which the end-to-end performance of the complete system degraded was significantly slower than that of the actual recognition component. In a second set of experiments, the ability to rapidly adapt the core understanding component of the system to a different domain has been tested. Using only a small amount of training data, experiments have shown that a semantic parser based on the HVS model originally trained on the ATIS corpus can be straightforwardly adapted to the somewhat different DARPA Communicator task using standard adaptation algorithms such as MAP [92; 94; 95] and log-linear interpolation [124]. The results presented support the claim that an SLU system which is statistically-based and trained entirely from data is intrinsically robust and can be readily adapted to new applications.

6.2 Conclusions

The existing approaches to spoken language understanding can be broadly classified into two categories, rule-based or statistically-based. The rule-based approaches require handcrafted semantic grammar rules to be defined, and the statistically-based approaches normally require fully-annotated training corpora in order to reliably estimate model parameters. The former needs heavy manual processing, whilst the latter is often difficult to obtain in practical applications. This thesis has proposed statistical approaches to building a purely data-driven SLU system that are comparable to existing systems but with low build cost. In particular, it has focused on statistical approaches to semantic parsing and dialogue act decoding, which are two major processes in an SLU system.

For semantic parsing, an ideal situation would be where the initial training data could be easily provided by the dialogue designer and where the semantic parsing model could be trained directly from unannotated data in a constrained way whilst at the same time being able to capture the underlying hierarchical semantic structures. With this motivation, this thesis proposed a Hidden Vector State (HVS) model which can be trained directly from abstract semantics without the use of word-level annotations. The HVS model is able to capture hierarchical structure in the semantics without incurring the computational tractability issues inherent in fully recursive models such as the hierarchical HMM.

For dialogue act decoding, existing work using Bayesian networks are based on the Naive Bayes approach and dependency links between semantic concepts are added according to the Minimum Description Length (MDL) principle [58; 59; 74]. This thesis has studied the extension of modified Tree-Augmented Naive Bayes networks (TANs) which have been found to give slightly improved performance.

The evaluation results on the integrated SLU system suggest that it is possible to build an SLU system without the need of linguistic knowledge or expensive annotation. Such a system can still give similar performance to the existing SLU systems that rely on either hand-crafted grammar rules or statistical models that are trained on fully-annotated training corpus data. Moreover, as the system is purely data-driven, it is robust to noise and can be readily ported or adapted to new applications.

6.3 Future Work

The current research work may be extended in several ways as discussed below.

6.3.1 Variants of the HVS model

The current version of the HVS model only allows one new semantic tag to be pushed into a vector state stack per input word. The effect of doing so is to limit the class of supported language to be right branching. It may not be sufficient to cover other more demanding domains such as Chinese, which is a left-recursive language. In such cases, this constraint needs to be relaxed such that more than one new semantic tags can be pushed into a vector state stack for each input word. However, the state space may then grow exponentially. Thus, efficient state pruning techniques need to be explored.

6.3.2 Online learning of the HVS model

Another important topic for future work is to port the HVS model to a real dialogue application and explore the possibilities for on-line learning. For example, if the mapping from an input utterance to semantic interpretation was initially wrong but was then repaired during the subsequent dialogue, it may be possible to use the corrected semantics to update the HVS model online.

6.3.3 Adaptation of the HVS model

The HVS model adaptation reported in this thesis was limited to domains with similar semantic concepts. Introduction of new semantic concepts would vary the dimensionalities of the HVS model parameters and make the adaptation problem more complicated. However, this may be overcome through parameter tying of semantic vector states that share similar dominance relationships.

6.3.4 Extension of TAN Networks for Dialogue Act Decoding

In this thesis, the modified TAN algorithm has been applied for dialogue act decoding. It is essentially an extension of Naive Bayes networks in which each semantic concept (attribute) node may have one additonal dependency link pointing to it besides the dependency link from the dialogue act or goal node. One obvious way to relax this constraint is to allow more than one dependency links between concept nodes. Another way to extend the TAN algorithm is to allow system beliefs or dialogue histories to be encoded into the network. More powerful methods of dialogue act decoding might exploit networks of networks.

6.3.5 A Real Spoken Dialogue System

Though the integrated spoken language understanding system reported in this thesis can take speech signals as input, they however still process users' utterances offline. It will be interested to port the understanding components into a real spoken dialogue system such that users can speak directly to the system and appropriate responses can be generated and fedback to the users. Many practical issues may then arise, such as response generation (whether to confirm or clarify what users said) and the system optimization needed to perform tasks in real-time etc.

References

[1] ARPA, *Proceedings of the spoken language systems technology workshop*, Morgan Kaufmann Publishers, Inc, Austin, Texas, Jan 1995. 1.1.2

[2] J. Peckham, "Speech understanding and dialogue over the telephone: An overview of the ESPRIT SUNDIAL project," in *Proceedings of the DARPA Speech and Natural Language Workshop*, Pacific Grove, California, Feb 1991, pp. 14–27. 1.1.2

[3] MITRE, *DARPA Communicator homepage*, http://fofoca.mitre.org/. 1.1.2

[4] AT&T, *Communicator System*, http://communicator.research.att.com/research/communicator.html. 1.1.2

[5] BBN Technologies, *Talk'N'Travel*, http://www.speech.bbn.com/communicator/home.htm. 1.1.2

[6] Carnegie Mellon University, *CMU Communicator*, http://fife.speech.cs.cmu.edu/Communicator/. 1.1.2

[7] Massachusetts Institute of Technology, *Mercury System*, http://www.sls.lcs.mit.edu/DARPAactivities.html. 1.1.2

[8] SRI International, *SRI Communicator*, http://www.ai.sri.com/ communic/. 1.1.2

[9] P. Price, "Evaluation of spoken language systems: the ATIS domain," in *Proc. of the DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, June 1990, pp. 91–95, Morgan Kaufman Publishers, Inc. 1.1.2

[10] J. Peckham, "A new generation of spoken dialog systems: Results and lessons from the SUNDIAL project," in *Proc. of Eurospeech*, Berlin, Germany, Sep. 1993, pp. 33–40. 1.1.2

[11] Lucent Technologies, *Natural Language Call Routing*, http://www.bell-labs.com/org/1133/Research/SpokenDialogSystems/routing.html. 1.1.2

[12] AT&T, *How May I Help You (HMIHY)*, http://www.research.att.com/ãlgor/hmihy/. 1.1.2

[13] Lucent Technologies, *Voice User Interface (VUI) for Accessing Personalized Information and Messaging Services*, http://www.bell-labs.com/org/1133/Research/SpokenDialogSystems/vui.html. 1.1.2

[14] General Magic Inc., *second-generation virtual assistant (Serengeti)*, http://www.genmagic.com. 1.1.2

[15] Nuance, *Nuance Automatic Banking System*, Available at +44 1223 516959. 1.1.2

[16] Unisys, *Automated Mortgage Broker*, http://www.unisys.com. 1.1.2

[17] E*Trade Financial, *Voice-automated trading of stocks*, http://www.etrade.com. 1.1.2

[18] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2, pp. 131–163, 1997. 1.4, 2.3.6, 4, 4.1.2, 4.4.4, 5.1, 6.1

[19] HTK, *Hidden Markov Model Toolkit (HTK) 3.2*, Cambridge University Engineering Department, 2004, http://htk.eng.cam.ac.uk/. 1.4, 5.1, 6.1

[20] D.A. Dahl, M. Bates, M. Brown, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and L. Shriberg, "Expanding the scope of the ATIS task: the ATIS-3 corpus," in *ARPA Human Language Technology Workshop*, Princeton, NJ, Mar. 1994. 1.5, 3.5.2

[21] CUData, *DARPA Communicator Travel Data*, University of Colorado at Boulder, 2004, http://communicator.colorado.edu/phoenix. 1.5, 3.5.2, 5.4.2.3

[22] H. Bunt, "Context and dialogue control," *Think*, vol. 3, pp. 19–31, 1994. 2.1.1

[23] R. Power, "The organization of purposeful dialogues," *Linguistics*, vol. 17, pp. 105–152, 1979. 2.1.1

[24] J. Carletta, A. Isard, J.C. Kowtko, G. Doherty-Sneddon, and A.H. Anderson, "The reliability of a dialogue structure coding scheme," *Computational Linguistics*, vol. 23, no. 1, pp. 13–32, 1997. 2.1.1

[25] J. Allen and M. Core, *Draft of DAMSL: dialogue act markup in several layers*, 1997, Unpublished manuscript. 2.1.1, 2.3.1.2

[26] W. Ward and S. Issar, "Recent improvements in the CMU spoken language understanding system," in *Proc. of the ARPA Human Language Technology Workshop.* 1996, pp. 213–216, Morgan Kaufman Publishers, Inc. 2.2

[27] S. Seneff, "Robust parsing for spoken language systems," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, San Francisco, 1992. 2.2

[28] J. Dowding, R. Moore, F. Andry, and D. Moran, "Interleaving syntax and semantics in an efficient bottom-up parser," in *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Maxico, June 1994, pp. 110–116. 2.2

[29] R. Pieraccini and E. Levin, "Stochastic representation of semantic structure for speech understanding," in *Proc. of Eurospeech*, Genova, Italy, Sep. 1991, pp. 383–386. 2.2.1

[30] R. Pieraccini, E. Levin, and C. Lee, "Stochastic representation of conceptual structure in the ATIS task," in *Proc. of the DARPA Speech and Natural Language Workshop*, Pacific Grove, California, Feb. 1991, pp. 121–124, Morgan Kaufman Publishers, Inc. 2.2.1

[31] R. Pieraccini, E. Tzoukermann, Z. Gorelov, E. Levin, C. Lee, and J. Gauvain, "Progress report on the CHRONUS system: ATIS benchmark results," in *Proc. of the DARPA Speech and Natural Language Workshop*, Harriman, NY, Feb. 1992, pp. 67–71, Morgan Kaufman Publishers, Inc. 2.2.1

[32] E. Levin and R. Pieraccini, "CHRONUS, the next generation," in *Proc. of the DARPA Speech and Natural Language Workshop*, Austin, TX, Jan. 1995, pp. 269–271, Morgan Kaufman Publishers, Inc. 2.2.1, 2.2.2.1, 3.5.1

[33] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modelling," *Computer Speech and Language*, vol. 8, no. 1, pp. 1–38, 1994. 2.2.1, 3.5.2

[34] S. Miller, M. Bates, R. Bobrow, R. Ingria, J. Makhoul, and R. Schwartz, "Recent progress in hidden understanding models," in *Proc. of the DARPA Speech and Natural Language Workshop*, Austin, TX, Jan. 1995, pp. 276–280, Morgan Kaufman Publishers, Inc. 2.2.2, 2.2.2.1, 2.2.2.1, 2.2.2.1

[35] R. Schwartz, S. Miller, D. Stallard, and J. Makhoul, "Language understanding using hidden understanding models," in *Proc. of Intl. Conf. on Spoken Language Processing*, Philadelphia, PA, Oct 1996. 2.2.2, 2.2.2.1, 2.2.2.1, 2.2.2.1

[36] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden markov model: Analysis and applications," *Machine Learning*, vol. 32, pp. 41–62, 1998. 2.2.2, 2.2.2.1, 2.2.2.1

[37] K.P. Murphy and M. Paskin, "Linear time inference in hierarchical hmms," in *Proceedings of Neural Information Processing Systems*, Vancouver, Canada, Dec. 2001. 2.2.2, 2.2.2.1, 2.2.2.1

[38] E. Charniak, "Immediate-head parsing for language models," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001. 2.2.2, 2.2.2.2

[39] C. Chelba and F. Jelinek, "Structured language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, Oct. 2000. 2.2.2, 2.2.2.3

[40] H. Erdogan, R. Sarikaya, Y. Gao, and M. Picheny, "Semantic structured language models," in *Proc. of Intl. Conf. on Spoken Language Processing*, Denver,Colorado, Sep. 2002. 2.2.2, 2.2.2.3

[41] S. Miller, R. Schwartz, R. Bobrow, and R. Ingria, "Hidden understanding models of natural language," in *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, June 1994, pp. 25–32, Morgan Kaufman Publishers, Inc. 2.2.2.1

[42] S. Miller, R. Schwartz, R. Bobrow, and R. Ingria, "statistical language processing using hidden understanding models," in *Proc. of the ARPA Human Language Technology Workshop*, Plainsboro, NJ, Mar. 1994, pp. 278–282, Morgan Kaufman Publishers, Inc. 2.2.2.1

[43] S.M. Katz, "Estimation of probabilisties from sparse data for the language model component of speech recognizer," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-35, pp. 400–401, 1987. 2.2.2.1

[44] P. Placeway, R. Schwartz, P. Fung, and L. Nguyen, "The estimation of powerful language models from small and large corpora," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 1993, pp. 33–36. 2.2.2.1

[45] J. Earley, "An efficient context-free parsing algorithm," *Communications of the ACM*, vol. 6, pp. 451–455, 1970. 2.2.2.1

[46] K. Lari and S.J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," *Computer Speech and Language*, vol. 4, no. 1, pp. 35–56, 1990. 2.2.2.1

[47] F. Jelinek, L. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos, "Decision tree parsing using a hidden derivation model," in *Proc. of the 1994 Human Language Technology Workshop*, 1994, pp. 272–277. 2.2.2.2, 2.2.2.3

[48] D. Magerman, "Statistical desicion-tree models for parsing," in *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995, pp. 276–283. 2.2.2.2

[49] E. Charniak, "Statistical parsing with a context-free grammar and word statistics," in *Proc. of the Fourteenth National Conference on Artificial Intelligence*. 1997, AAAI Press/MIT Press, Menlo Park. 2.2.2.2

[50] C. Chelba and M. Mahajan, "Information extraction using the structured language model," in *Proc. of Conference on Empirical Methods in Natural Language Processing*, 2001. 2.2.2.3

[51] C. Chelba and P. Xu, "Richer syntactic dependencies for structured language modeling," in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, Madonna di Campiglio, Trento-Italy, Dec. 2001. 2.2.2.3

[52] C. Chelba, "Portability of syntactic structure for language modeling," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Salt Lake City, Utah, 2001. 2.2.2.3

[53] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39 of B, pp. 1–38, 1977. 2.2.2.3

[54] T. Ward, "How long until a high school student can build a language understanding system," in *Proc. of Intl. Conf. on Spoken Language Processing*, Beijing, China, Oct. 2000. 2.2.2.3

[55] D.M. Magerman, *Natural language parsing as statistical pattern recognition*, Ph.D. thesis, Stanford University, Palo Alto, CA, Feb. 1994. 2.2.2.3

[56] C. Chelba and F. Jelinek, "Recognition performance of a structured language model," in *Proc. of Eurospeech*, Budapest, Hungary, Sep. 1999. 2.2.2.3

[57] J. Henderson, "Inducing history representations for broad coverage statistical parsing," in *Proceedings of the joint meeting of the North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May 2003. 2.2.3

[58] H. Meng, W. Lam, and C. Wai, "To believe is to understand," in *Proceedings of the 6th European Conference on Speech Communication and Technology*, 1999. 2.3, 2.3.1, 2.3.1.1, 6.2

[59] S. Keizer, R. Akker, and A. Nijholt, "Dialogue act recognition with bayesian networks for Dutch dialogues," in *3rd SIGdial Workshop on Discourse and Dialogue*, Philadelphia, Pennsylvania, July 2002. 2.3, 2.3.1, 2.3.1.2, 6.2

[60] J.R. Bellegarda and K.E.A. Silverman, "Natural language spoken interface control using data-driven semantic inference," *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 3, pp. 267– 277, May 2003. 2.3, 2.3.2, 2.3.2.1

[61] K. Lagus and J. Kuusisto, "Topic identification in natural language dialogues using neural networks," in *3rd SIGdial Workshop on Discourse and Dialogue*, Philadelphia, Pennsylvania, July 2002. 2.3, 2.3.2, 2.3.2.2

[62] A.L. Gorin, "On automated language acquisition," *Journal of the Acoustical Society of America*, vol. 97, no. 6, pp. 3441–3461, 1995. 2.3, 2.3.3, 2.3.3.1

[63] A.L. Gorin, "Processing of semantic information in fluently spoken language," in *Proc. of Intl. Conf. on Spoken Language Processing*, Philadelphia, Oct. 1996. 2.3, 2.3.3, 2.3.3.1

[64] A.L. Gorin, G. Riccardi, and J.H. Wright, "How may i help you," *Computer Speech and Language*, vol. 23, no. 1-2, pp. 113–127, 1997. 2.3, 2.3.3, 2.3.3.1

[65] H.K.J. Kuo, C.H. Lee, I. Zitouni, E. Fosler-Lussier, and E. Ammicht, "Discriminative training for call classification and routing," in *Proc. of Intl. Conf. on Spoken Language Processing*, Denver, Colorado, Sep. 2002. 2.3, 2.3.3, 2.3.3.2

[66] K. Samuel, S. Carberry, and K. Vijay-Shanker, "Dialogue act tagging with transformation-based learning," in *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, Montreal, Quebec, Aug. 1998, pp. 1150–1156. 2.3, 2.3.4

[67] M. Araki, K. Ueda, T. Nishimoto, and Y. Niimi, "A semantic tagging tool for spoken dialogue corpus," in *Proc. of Intl. Conf. on Spoken Language Processing*, 2000. 2.3, 2.3.4

[68] K. Jokinen, H. Tanaka, and A. Yokoo, "Context management with topics for spoken dialogue systems," in *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLIN-ACL)*, Montreal, Canada, Aug. 1998, pp. 631–637. 2.3, 2.3.5

[69] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial Intelligence*, vol. 42, no. 2, pp. 393–405, 1990. 2.3.1

[70] S.G. Pulman, "Conversational games, belief revision and Bayesian networks," in *7th Computational Linguistics in the Netherlands (CLIN) meeting*, Nov. 1996, ed. J. Landsbergen et al. 2.3.1

[71] D. Heckerman and E. Horvitz, "Inferring information goals from free-text queries: a Bayesian approach," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998, pp. 230–237, Morgan Kaufmann Publishers. 2.3.1

[72] H. Meng, C. Wai, and R. Pierracinni, "The use of belief networks for mixed-initiative dialog modeling," in *Proc. of Intl. Conf. on Spoken Language Processing*, Beijing, China, Oct. 2000. 2.3.1.1

[73] H. Meng, S. Lee, and C. Wai, "CU FOREX: A bilingual spoken dialog system for foreign exchange inquiries," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2000. 2.3.1.1

[74] H. Meng, W. Lam, and K.F. Low, "Learning belief networks for language understanding," in *Proceedings of the 1999 International Workshop on Automatic Speech Recognition and Understanding*, 1999. 2.3.1.1, 7, 1, 6.2

[75] H. Meng and K.C. Siu, "Semi-automatic acquisition of domain-specific semantic structures," *IEEE Trans. on Knowledge and Data Engineering*, 2001. 2.3.1.1

[76] J. Rissanen, "Modelling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978. 2b

[77] G.F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, pp. 309–347, 1992. 2.3.1.2

[78] D. Heckerman, "A tutorial on learning with Bayesian networks," *Learning in Graphical Models*, 1999, In M. Jordan, editor. 2.3.1.2

[79] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman, "Indexing by latent semantic analysis," *Journal of American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1991. 2.3.2.1

[80] J.R. Bellegarda and K.E.A. Silverman, "Data-driven semantic inference for unconstrained desktop command and control," in *Proc. of Eurospeech*, Aalborg, Denmark, Sep. 2001. 2.3.2.1

[81] J.R. Bellegarda, "Exploiting latent semantic information in statistical language modeling," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1279–1296, Aug. 2000. 2.3.2.1

[82] W. Kintsch, V.L. Patel, and K.A. Ericsson, "The role of long-term working memory in text comprehension," *Psychologia*, vol. 42, pp. 186–198, 1999. 2.3.2.1

[83] G. Salton, A. Wong, and C.S. Yang, "A vector-space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975. 2.3.2.2

[84] K. Jokinen, T. Hurtig, K. Hynna, M. Kaipainen, and A. Kerminen, "Self-organizing dialogue management," in *Proceedings of the Second Workshop on Language Processing and Neural Networks*, Tokyo, Japan, Nov. 2001. 2.3.2.2

[85] J. Chu-Carroll and B. Carpenter, "Vector-based natural language call routing," *Computer Linguistics*, vol. 25, no. 3, pp. 361–388, 1999. 2.3.3, 2.3.3.2

[86] K. Arai, J.H. Wright, G. Riccardi, and A.L. Gorin, "Grammar fragment acquisition using syntactic and semantic clustering," *Speech Communication*, vol. 27, no. 1, pp. 43–62, 1999. 2

[87] S. Cox, "Discriminative techniques in call routing," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Hong Kong, Apr. 2003. 2.3.3.2, 2.3.3.2

[88] S. Katagiri, B.H. Juang, and C.H. Lee, "Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method," *Proceedings of IEEE*, vol. 86, pp. 2345–2373, Nov. 1998. 2.3.3.2

[89] E. Brill, "Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging," *Computational Linguistics*, vol. 21, no. 4, pp. 543–566, 1995. 2.3.4

[90] P.R. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit," in *Proc. of Eurospeech*, 1997. 2.3.5

[91] G.F. Cooper, *Probabilistic Inference using Belief Networks is NP-hard*, 1987. 2.3.6, 4

[92] J.L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994. 2.4, 2.4.1, 6.1

[93] M.J. Gales and P.C. Woodland, "Mean and variance adaptation within the MLLR framework," *Computer Speech and Language*, vol. 10, pp. 249–264, Oct. 1996. 2.4, 5.1

[94] M. Bacchiani and B. Roark, "Unsupervised language model adaptation," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Hong Kong, Apr. 2003. 2.4, 2.4.1, 6.1

[95] B. Roark and M. Bacchiani, "Supervised and unsupervised PCFG adaptation to novel domains," in *Proceedings of the joint meeting of the North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May 2003. 2.4, 2.4.1, 2.4.1, 6.1

[96] X. Luo, S. Roukos, and T. Ward, "Unsupervised adaptation of statistical parsers based on Markov transform," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Keystone, Colorado, Dec. 1999. 2.4, 2.4.2.1, 2.4.2.2, 2.4.3

[97] X. Luo, "Parser adaptation via householder transform," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Istanbul, Turkey, June 2000. 2.4, 2.4.2.2, 2.4.2.2, 2.4.3

[98] O. Siohan, C. Chesta, and C.-H. Lee, "Joint maximum a posteriori adaptation of transformation and hmm parameters," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 4, pp. 417–428, 2001. 2.4.2

[99] C. Phillips, "Right association in parsing and grammar," in *Papers on Language Processing and Acquisition*, C. Schtze, K. Broihier, and J. Ganger, Eds., pp. 37–93. 1995, MITWPL 26. 3.1

[100] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2)*, Cambridge University Engineering Department, 2004, http://htk.eng.cam.ac.uk/. 3.3.3

[101] CUPheonix, *CU Pheonix Parser*, University of Colorado at Boulder, 2003, http://communicator.colorado.edu/phoenix. 3.5.2

[102] V. Goel and W. Byrne, "Task dependent loss functions in speech recognition: Application to named entity extraction," in *ESCA ETRW Workshop on Accessing Information from Spoken Audio*, Cambridge, UK, 1999, pp. 49–53. 3.5.2, 5.2

[103] I.H. Witten and T.C. Bell, "The zero frequency problem: estimating the probabilities of novel events in adaptive text compression," *IEEE Trans. on Information Theory*, pp. 1085–1093, 1991. 3.5.2

[104] C.K. Chow and C.N. Liu, "Approximating discrete probability distributions with dependence tree," *IEEE Trans. on Information Theory*, vol. 14, pp. 462–467, 1968. 4.1.1

[105] J.P. Sacha, *New synthesis of Bayesian network classifiers and interpretation of cardiac SPECT images*, Ph.D. thesis, University of Toledo, 1999. 4.1.1, 4.1.1

[106] D. Spiegelhalter, P. Dawid, S. Lauritzen, and R. Cowell, "Bayesian analysis in expert systems. statistical science," *Statistical Science*, vol. 8, pp. 219–282, 1993. 4.1.1

[107] D. Heckerman, D. Geiger, and D.M. Chickering, "Learning Bayesian networks: the combination of knowledge and statistical data," *Machine Learning*, vol. 20, pp. 197–243, 1995. 4.1.1

[108] E. Castillo, J.M. Gutierrez, and A.S. Hadi, *Expert Systems and Probabiistic Network Models*, Springer-Verlag New York, Inc., 1996. 4.1.1

[109] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992. 4.1.1

[110] P. Haddawy, "An overview of some recent developments in bayesian problem solving techniques," *AI Magazine*, vol. 20, no. 2, pp. 11–19, 1999. 4.1.1

[111] D. Heckerman, "A tutorial on learning Bayesian networks," *Technical Report MSR-TR-95-06*, 1995. 4.1.2

[112] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *Computational Statistics Quarterly*, vol. 5, no. 4, pp. 269–282, 1990. 4.2, 6.1, A

[113] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *International Journal of Approximate Reasoning*, vol. 15, no. 3, pp. 225–263, 1996. 4.2, A

[114] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994. 4.4.5

[115] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003. 4.4.5

[116] D.L.T. Rohde, *A Connectionist model of sentence comprehension and production*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2002. 4.4.5

[117] D.J.C. MacKay, "Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, pp. 469–505, 1995. 4.4.5, 7

[118] A. Zell, N. Mache, R. Huebner, M. Schmalzl, T. Sommer, and T. Korb, *SNNS: Stuttgart Neural Network Simulator*, Institute for Parallel and Distributed High Performance Systems. University of Stuttgart, Germany, 1998, http://www-ra.informatik.uni-tuebingen.de/SNNS/. 4.4.5

[119] Yulan He and Steve Young, "Hidden vector state model for hierarchical semantic parsing," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Hong Kong, Apr. 2003. 5.1

[120] N. Kumar, *Investigation of Silicon Auditory Models and Generalization of Linear Discriminant analysis for Improved Speech Recognition*, Ph.D. thesis, Johns Hopkins University, Baltimore MD, 1997. 5.1

[121] R. Kneser and H. Ney, "Improved clustering techniques for class-based statistical language modelling," in *Proceedings of the European Conference on Speech Communication and Technology*, 1993, pp. 973–976. 5.3

[122] T. Briscoe, "Robust parsing," in *Survey of the State of the Art of Human Language Technology*, R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Eds., chapter 3.7. Cambridge University Press, Cambridge, England, 1996. 5.4

[123] A.P. Varga, H.J.M. Steeneken, M. Tomlinson, and D. Jones, "The NOISEX-92 study on the effect of additive noise on automatic speech recognition," Tech. Rep., DRA Speech Research Unit, 1992. 5.4.1

[124] D. Klakow, "Log-linear interpolation of language models," in *Proc. of Intl. Conf. on Spoken Language Processing*, Sydney, Australia, Nov. 1998. 5.4.2.2, 6.1

[125] S. Martin, A. Kellner, and T. Portele, "Interpolation of stochastic grammar and word bigram models in natural language understanding," in *Proc. of Intl. Conf. on Spoken Language Processing*, Beijing, China, Oct. 2000. 5.4.2.2

[126] M. Rayner, D. Carter, V. Digalakis, and P. Price, "Combining knowdge sources to recoder N-best speech hypothesis lists," in *Proceedings of the ARPA Human Language Technology Meeting*, 1994. 5.5

[127] R. Moore, D. Appelt, J. Dowding, J.M. Gawron, and D. Moran, "Combining linguistic and statistical knowledge sources in natural-language processing for ATIS," in *ARPA Spoken Language Technology Workshop*, 1995. 5.5

[128] K. Hacioglu and W. Ward, "Combining language models: Oracle approach," in *Human Language Technology Conference*, San Diego, CA, Mar. 2001. 5.5

[129] A. Chotimongkol and A.I. Rudnicky, "N-best speech hypotheses reordering using linear regression," in *Proc. of Eurospeech*, Aalborg, Denmark, 2001, pp. 1829–1832. 5.5

[130] S.J. Young, "The statistical approach to the design of spoken dialogue systems," Tech. rep. cued/f-infeng/tr.433, Cambridge University Engineering Department, 2002. 6.1

[131] S.M. Aji and R.J. McEliece, "The generalized distributive law," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000. 4

[132] MySQL, *MySQL Database Server*, http://www.mysql.com. C

Appendix A

Probability Propagation in Trees of Clusters (PPTC)

The Probability Propagation in Trees of Clusters (PPTC) algorithm [112; 113] consists of four major steps, graphical transformation, initialization, global propagation, and marginalization. This appendix explains each of these four steps in detail.

A.1 Graphical Transformation

Assume a DAG of a Bayesian network can be represented by $G\langle \mathbf{V}, \mathbf{E} \rangle$ where $\mathbf{V}$ is a set of vertices and $\mathbf{E}$ is a set of edges in the DAG. The *graphical transformation* step transforms a DAG into a junction tree whose vertices are labelled by cliques and whose edges are labelled by separator sets formed by intersection of the two cliques at either side. The definition of *clique* and *separator set* will become clear in the procedure described below.

1. Transform the DAG of a Bayesian network into an undirected graph by dropping the directions of the arcs and moralizing the graph by adding links between parents of each node. The result is a *moral graph* (as in Figure A.1-(b)).

2. Triangulate the moral graph using the elimination algorithm.

   (a) Order the nodes by *maximum cardinality search* and visit in reverse order. An ordering of the vertices in a graph is obtained by first assigning 1 to an arbitrary vertex, then selecting the next vertex as the vertex adjacent to the largest number of previously numbered vertices, increasing the counter by 1 and assigning new counter to the selected vertex.

   (b) For each visited node V, add links so that all neighbours of V are linked to each other.

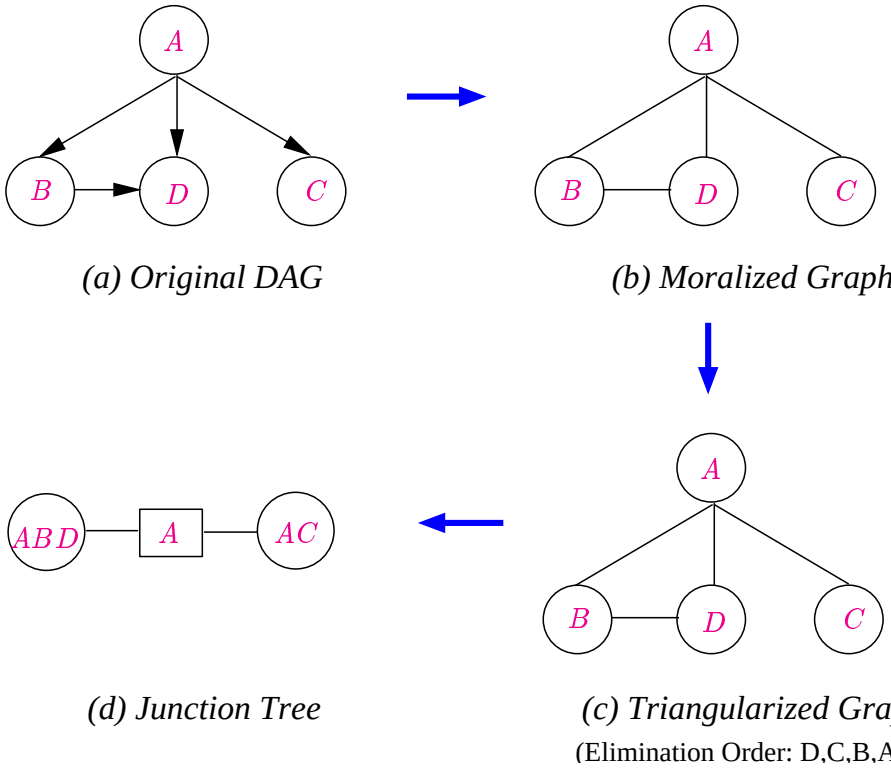


Figure A.1: Example of Graphical Transformation from DAG to Junction Tree.

(c) Remove V together with all the links and continue until all the nodes have been eliminated.

(d) The original graph together with all the links that were added during the elimination process is triangulated (see Figure A.1-(c)).

3. Identify subsets of nodes, called *cliques*, from the triangulated graph. A node V together with its neighbours form a *cluster*. Cliques can be simply extracted during the triangulation process by saving each cluster that is not a subset of any previously saved cluster. For example, nodes A, B, D in Figure A.1-(c) form a clique.

4. Connect the cliques to form a junction tree. This can be done as follows. First add an edge between clique i and clique j if $clique_i \cap clique_j \neq \oslash$. The weight of this edge is set to be $|clique_i \cap clique_j \neq \oslash|$, i.e. the number of variables they have in common. Such common elements form the separator set between two cliques (A is a separator between clique ABD and AC in Figure A.1-(d)). As any maximal weight spanning tree (MST) is a junction tree (see [131] for a proof), an MST can be constructed using Kruskal's algorithm in $O(E \log E)$ time, or using Prim's algorithm in $O(V^2)$ time.

A.2 Initialization

The following procedure assigns initial junction tree potentials, using the conditional probabilities from the Bayesian network:

1. For each clique and separator X, set its potential $\phi_X(x) = 1$.
2. For each variable V, do the following:
   - Assign to V a clique X which contains the parents of V (denoted by Π_V)
   - Multiply ϕ_X by $P(V|\Pi_V)$, i.e. $\phi_X \leftarrow \phi_X P(V|\Pi_V)$
   - Note that the initialization satisfies the joint probability distribution of the original network, since

$$\frac{\prod_{i=1}^{N} \phi_{X_i}}{\prod_{j=1}^{N-1} \phi_{S_j}} = \frac{\prod_{k=1}^{Q} P(V_k|\Pi_{V_k})}{1} = P(V_1, \cdots, V_M) \tag{A.1}$$

     where N is the number of cliques, Q is the number of variables, M is the number of vertices, and ϕ_{X_i} and ϕ_{S_j} are the clique and separator set potentials respectively.

A.3 Global Propagation (Message Passing)

Global propagation or message passing causes each clique to pass a message to its neighbouring cliques. Such message passes are ordered so that each message pass preserves the consistency introduced by previous message passes. After global propagation, the junction tree is locally consistent in the sense that for each clique X and its neighbouring separator set S, it holds that

$$\sum_{X \setminus S} \phi_X = \phi_S \tag{A.2}$$

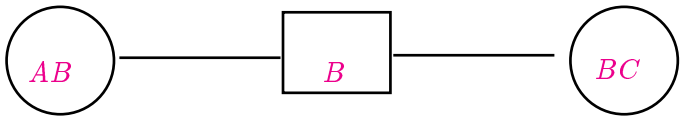


Figure A.2: Example of a junction tree.

Taking Figure A.2 as an example where circles represent cliques, rectangles denotes separator sets, a single message passing can be described as follows:

- First update separator potential $\phi_B^* = \sum_A \phi_{AB}$

- Then update the clique potential $\phi^*_{BC} = \frac{\phi^*_B}{\phi_B}\phi_{BC}$ Note that these updates leave the joint probability invariant since

$$P(A,B,C) = \frac{\phi_{AB}\phi^*_{BC}}{\phi^*_B} = \frac{\phi_{AB}\phi^*_B\phi_{BC}}{\phi_B\phi^*_B} = \frac{\phi_{AB}\phi_{BC}}{\phi_B} \tag{A.3}$$

- Finally, update $\phi^{**}_B = \sum_C \phi_{BC}$ and $\phi^*_{AB} = \frac{\phi^{**}_B}{\phi^*_B}\phi_{AB}$

For a junction tree with n cliques, the global propagation algorithm begins by choosing an arbitrary clique X, and then performing $2(n-1)$ message passes, divided into two stages. During the CollectEvidence stage, each clique passes a message to its neighbours in X's direction, starting from the cliques farthest from X. During the DistributeEvidence stage, each clique passes a message to its neighbours away from X's direction, starting from X itself.

A.4 Marginalization

Once a consistent junction tree is obtained, the probability $P(V)$ of any variable V can be computed as follows:

1. For a variable V, identify a clique X that contains V.
2. Compute $P(V)$ by marginalizing ϕ_X by

$$P(V) = \sum_{X\setminus\{V\}} \phi_X \tag{A.4}$$

Appendix B

Definition of a TAN Multinet Model

Figure B.1 gives a sample topology definition of a TAN multinet model which may consist of a set of TAN networks. The TAN multinet definition is bracketed by the symbol <BeginBBN> and <EndBBN>. The first few lines of the definition specify global features of the TAN multinet model, such as the total number of goals, the number of distinct semantic concept tags (or attributes), and the prior probability of each goal. In the implementations, one TAN network represents one goal. Therefore, there are altogether 16 TAN networks for the model definition illustrated in Figure B.1 and each of them is bracketed by the symbol <BeginNet> and <EndNet>.

For each TAN network, the <NetName> specifies the name of the network, which is typically the dialogue act that the network represents [1]. The keyword <NumAtt> indicates the number of attribute or concept nodes in the network. Recall from Chapter 4 that a Bayesian network consists of a set of random variables and the conditional probability distribution (CPD) for each variable. Such variables are introduced following the keyword <Variable>. Variable names and prior probability of each variable are listed by the keyword <VarName> and <VarPrior> respectively. The CPD is introducted by the keyword <CPD>. Conditional relationships are represented by the keywords <For> and <Given>. For example,

<For> FLIGHT

<Given> airfare

<Given> COST

indicates that the CPD is for the probability $P(FLIGHT|airfare, COST)$. The keyword

[1] Note here character strings in small case represent dialogue acts while character strings in upper case represent semantic concepts. Therefore, in Figure B.1, "airfare" repsents a dialogue act, and "AIRFARE" represents a semantic concept.

<Sizes> lists out the number of values that each variable can take. In the implementations, each variable can only be in either *present* state or *absent* state. Thus, the size for each variable is always 2. Probabilities of variables in different states are then listed following the keyword <Table>.

```
<BeginBBN>
    <NumGoals> 16
    <NumTags> 110
    <GoalPrior>
       0.025 0.020 0.088 0.031
       0.004 0.003 0.028 0.028
       0.746 0.001 0.003 0.056
       0.002 0.002 0.010 0.005
    <BeginNet>
       <NetName> airfare
       <NumAtt> 5
       <Variable>
          <VarName>
             airfare AIRFARE COST
             COST_RELATIVE FLIGHT ROUND_TRIP
          <VarPrior>
             0.088 0.198 0.043
             0.086 0.059 0.097
       <CPD>
          <For> airfare
             <Sizes> 2
             <Table>
                0.088 0.912
       <CPD>
          <For> AIRFARE
          <Given> airfare
             <Sizes> 2 2
             <Table>
                0.198 0.004 0.802 0.996
       <CPD>
          <For> FLIGHT
          <Given> airfare
          <Given> COST
             <Sizes> 2 2 2
             <Table>
                0.230 0.051 0.211 0.173
                0.770 0.949 0.789 0.827
       ...
    <EndNet>
    ...
<EndBBN>
```

Figure B.1. Definition for a TAN multinet model

Appendix C

Case Frame Sample File

The SQL query generator relies on the case frame definition file to generate the appropriate queries in order to fetch answers from the database. Each case frame contains a frame name which corresponds to one of the predefined dialogue acts, and a number of slots which correspond to a set of semantic concepts that are relevant to the specific dialogue act. For each user's input utterance, the semantic parser produces a sequence of semantic concept/value pairs and the dialogue act decoder infers the most probable dialogue act based on the parse results. Such a dialogue act is then used to identify the case frame and the concept/value pairs extracted from the semantic parse results are used to fill in the respective slots to form the partial SQL queries. Finally, the query generator "glues" all the partial queries to form the final SQL query.

In the research work reported here, the MySQL database server [132] is used and the queries generated thus need to follow the formats defined by MySQL. A typical example of an SQL query produced by the query generator is:

```
Utt: I need a flight from toronto to newark one way leaving wednesday
     evening or thursday morning
SQL: SELECT DISTINCT v0.flight_id FROM flight v0, city_airport v1,
     city_airport v2, days v3, date_day v4, flight_fare v5, fare v6
     WHERE (v0.from_airport = v1.airport_code AND v1.city_name =
     'TORONTO') AND (v0.to_airport = v2.airport_code AND v2.city_name
     = 'NEWARK') AND ((v0.flight_days = v3.days_code AND v3.day_name =
     v4.day_name AND v4.day_name = 'WEDNESDAY' AND v4.day_number = 6
     AND v4.month_number = 4 AND v4.year = 1994) AND (v0.departure_time
     >= 1800 AND v0.departure_time <= 2200) OR (v0.flight_days =
     v3.days_code AND v3.day_name = v4.day_name AND v4.day_name =
     'THURSDAY' AND v4.day_number = 7 AND v4.month_number = 4 AND v4.year
```

```
     = 1994) AND (v0.departure_time >= 0 AND v0.departure_time <= 1200))
     AND (v0.flight_id = v5.flight_id AND v5.fare_id = v6.fare_id AND
     v6.round_trip_required = 'NO');
```

The following illustrates an example of the case frame defintion for the frame FLGIHT.

```
frame FLIGHT
{
  Sql: SELECT DISTINCT flight.flight_id
  Table: flight
----
    COST_RELATIVE_ROUND
    Sql_p: SELECT @cost := COST_RELATIVE_ROUND(fare.round_trip_cost) FROM
           $1 WHERE fare.fare_id = flight_fare.fare_id AND
           flight_fare.flight_id = flight.flight_id
    Sql: flight.flight_id = flight_fare.flight_id AND flight_fare.fare_id
         = fare.fare_id AND fare.round_trip_cost = @cost
    Table: flight_fare, fare
----
    COST_RELATIVE_SINGLE
    Sql_p: SELECT @cost := COST_RELATIVE_SINGLE(fare.one_direction_cost)
           FROM $1 WHERE fare.fare_id = flight_fare.fare_id AND
           flight_fare.flight_id = flight.flight_id
    Sql: flight.flight_id = flight_fare.flight_id AND flight_fare.fare_id
         = fare.fare_id AND fare.one_direction_cost = @cost
    Table: flight_fare, fare
----
    ROUTE
    Sql: v0.flight_id = flight_leg.flight_id AND flight_leg.flight_id =
         flight.flight_id
    Table: flight_leg, flight
----
    FROMLOC
    Sql: flight.from_airport = city_airport.airport_code
    Table: city_airport
    ....CITY_NAME
        Sql: city_airport.city_name = 'CITY_NAME'
```

```
    ....STATE_NAME
        Sql: city_airport.state_name = 'STATE_NAME'
    ....STATE_CODE
        Sql: city_airport.state_code = 'STATE_CODE'
    ....AIRPORT_NAME
        Sql: city_airport.airport_name = 'AIRPORT_NAME'
    ....AIRPORT_CODE
        Sql: city_airport.airport_code = 'AIRPORT_CODE'
    ....COUNTRY_NAME
        Sql: city_airport.country_name = 'COUNTRY_NAME'
----
    TOLOC
    Sql: flight.to_airport = city_airport.airport_code
    Table: city_airport
    ....CITY_NAME
        Sql: city_airport.city_name = 'CITY_NAME'
    ....STATE_NAME
        Sql: city_airport.state_name = 'STATE_NAME'
    ....STATE_CODE
        Sql: city_airport.state_code = 'STATE_CODE'
    ....AIRPORT_NAME
        Sql: city_airport.airport_name = 'AIRPORT_NAME'
    ....AIRPORT_CODE
        Sql: city_airport.airport_code = 'AIRPORT_CODE'
    ....COUNTRY_NAME
        Sql: city_airport.country_name = 'COUNTRY_NAME'
----
    STOPLOC
    Sql: flight.flight_id = flight_stop.flight_id AND
         flight_stop.stop_airport = city_airport.airport_code
    Table: flight_stop, city_airport
    ....CITY_NAME
        Sql: city_airport.city_name = 'CITY_NAME'
    ....STATE_NAME
        Sql: city_airport.state_name = 'STATE_NAME'
    ....STATE_CODE
```

```
        Sql:  city_airport.state_code = 'STATE_CODE'
    ....AIRPORT_NAME
        Sql: city_airport.airport_name = 'AIRPORT_NAME'
    ....AIRPORT_CODE
        Sql: city_airport.airport_code = 'AIRPORT_CODE'
    ....COUNTRY_NAME
        Sql: city_airport.country_name = 'COUNTRY_NAME'
----
    DEPART_DATE
    Sql: flight.flight_days = days.days_code AND days.day_name =
         date_day.day_name
    Table: days, date_day
    ....DAY_NAME
        Sql: date_day.day_name = 'DAY_NAME'
    ....DAY_NUMBER
        Sql: date_day.day_number = DAY_NUMBER
    ....MONTH_NUMBER
        Sql: date_day.month_number = MONTH_NUMBER
    ....YEAR
        Sql: date_day.year = YEAR
----
    DEPART_TIME
    ....START_TIME
        Sql: flight.departure_time >= START_TIME
    ....END_TIME
        Sql: flight.departure_time <= END_TIME
    ....TIME
        Sql: flight.departure_time TIME
----
    DEPART_TIME.TIME_MOD
    Sql_p: SELECT @time := DEPART_TIME.TIME_MOD(flight.departure_time)
           FROM $1 WHERE
    Sql: flight.departure_time = @time
----
    ARRIVE_DATE
    Sql:  flight.flight_days = days.days_code AND days.day_name =
```

```
          date_day.day_name AND $2 AND flight.departure_time >
          flight.arrival_time AND (flight.time_elapsed >=  60 OR
          flight.arrival_time <  41)
    Table: days, date_day
    ....DAY_NAME
        Sql: date_day.day_name = 'DAY_NAME'
    ....DAY_NUMBER
        Sql: date_day.day_number = DAY_NUMBER
    ....MONTH_NUMBER
        Sql: date_day.month_number = MONTH_NUMBER
    ....YEAR
        Sql: date_day.year = YEAR
----
    ARRIVE_1_DATE
    Sql: flight.flight_days = days.days_code AND days.day_name =
         date_day.day_name AND $2 AND NOT(flight.departure_time >
         flight.arrival_time AND (flight.time_elapsed >=  60 OR
         flight.arrival_time <  41))
    Table: days, date_day
    ....DAY_NAME
        Sql: date_day.day_name = 'DAY_NAME'
    ....DAY_NUMBER
        Sql: date_day.day_number = DAY_NUMBER
    ....MONTH_NUMBER
        Sql: date_day.month_number = MONTH_NUMBER
    ....YEAR
        Sql: date_day.year = YEAR
----
    ARRIVE_TIME
    ....START_TIME
        Sql: flight.arrival_time >= START_TIME
    ....END_TIME
        Sql: flight.arrival_time <= END_TIME
    ....TIME
        Sql: flight.arrival_time TIME
----
```

```
    ARRIVE_TIME.TIME_MOD
    Sql_p: SELECT @time := ARRIVE_TIME.TIME_MOD(flight.arrival_time)
           FROM $1 WHERE
    Sql: flight.arrival_time = @time
----
    FARE_CAT
    Sql: flight.flight_id = flight_fare.flight_id AND
         flight_fare.fare_id = fare.fare_id
    Table: flight_fare, fare
    ....ROUND_TRIP
        Sql: ROUND_TRIP
    ....CLASS_TYPE
        Sql: fare.fare_basis_code = fare_basis.fare_basis_code AND
             fare_basis.class_type = 'CLASS_TYPE'
        Table: fare_basis
    ....ECONOMY_TYPE
        Sql: fare.fare_basis_code = fare_basis.fare_basis_code AND
             fare_basis.economy = ECONOMY_TYPE
        Table: fare_basis
    ....FARE_BASIS_CODE
        Sql: fare.fare_basis_code = 'FARE_BASIS_CODE'
    ....FARE_AMOUNT_SINGLE
        Sql: fare.one_direction_cost FARE_AMOUNT_SINGLE
    ....FARE_AMOUNT_ROUND
        Sql: fare.round_trip_cost FARE_AMOUNT_ROUND
    ....DISCOUNT
        Sql: fare.fare_basis_code = fare_basis.fare_basis_code AND
             fare_basis.discounted = 'DISCOUNT'
        Table: fare_basis
    ....DEPART_DATE
        Sql: fare.fare_basis_code = fare_basis.fare_basis_code AND
             fare_basis.basis_days = days.days_code
        Table: fare_basis, days
----
    AIRLINE_CODE
    Sql: flight.airline_code AIRLINE_CODE
```

```
----
    AIRLINE_NAME
    Sql: flight.airline_code = airline.airline_code AND
         airline.airline_name LIKE 'AIRLINE_NAME%'
    Table: airline
----
    AIRPORT_NAME
    Sql: (flight.from_airport = airport.airport_code AND
         airport.airport_name LIKE 'AIRPORT_NAME%') OR
         (flight.to_airport = airport.airport_code AND
         airport.airport_name LIKE 'AIRPORT_NAME%')
    Table: airport
----
    CITY_NAME
    Sql: flight.from_airport = city_airport.airport_code AND
         city_airport.city_name = 'CITY_NAME' OR
         flight.to_airport = city_airport.airport_code AND
         city_airport.city_name = 'CITY_NAME'
    Table: city_airport
----
    STOP_CONNECT
    ....FLIGHT_STOP
        Sql: flight.stops = FLIGHT_STOP
    ....CONNECT
        Sql: flight.connections CONNECT
----
    FLIGHT_NUMBER
    Sql: v0.flight_id = flight_leg.flight_id AND
         flight_leg.leg_flight = flight.flight_id AND
         flight.flight_number = FLIGHT_NUMBER
    Table: flight_leg, flight
----
    FLIGHT_DAYS
    Sql: flight.flight_days = 'FLIGHT_DAYS'
----
    AIRCRAFT_CAT
```

```
    Sql: flight.aircraft_code_sequence =
         equipment_sequence.aircraft_code_sequence AND
         equipment_sequence.aircraft_code = aircraft.aircraft_code
    Table: equipment_sequence, aircraft
    ....AIRCRAFT_CODE
        Sql: aircraft.aircraft_code = 'AIRCRAFT_CODE' OR
             aircraft.basic_type = 'AIRCRAFT_CODE'
    ....MANUFACTURER
        Sql: aircraft.manufacturer = 'MANUFACTURER'
    ....PROPULSION
        Sql: aircraft.propulsion = 'PROPULSION'
----
    MEAL
    Sql: flight.meal_code = food_service.meal_code
    Table: food_service
    ....MEAL_DESCRIPTION
        Sql: food_service.meal_description = 'MEAL_DESCRIPTION'
    ....COMPARTMENT
        Sql: food_service.compartment = 'COMPARTMENT'
----
    FLIGHT_MOD
    Sql: $*
----
}
```