Using Domain Knowledge to Improve End to End Performance in a Speech Translation System

Oren Glickman

May 12, 1995

Masters Project
Computational Linguistics
Carnegie Mellon University

Committee:
Alex Waibel
Bob Carpenter
Lori Levin

Abstract

My work consists of using domain knowledge to reduce ambiguity and improve end to end performance in a multi-lingual spoken language translation system. I take advantage of domain knowledge, such as current date and context, to help reduce the difficulties of spoken language and errors introduced during speech recognition.

The main feature of the work is that the system works on the recognizers lattice rather then the top-best hypothesis. Parsing a word lattice involves finding a path of connecting words within the lattice that is grammatical. In this way more information is obtained from the recognition state and hence forcing disambiguation to a later state where more knowledge is available.

Chapter 1

Introduction

1.1. Problem Statement

Machine Translation of spoken language encounters all of the difficulties of written language (such as ambiguity) with the addition of problems that are specific to spoken language such as speech disfluencies, errors introduced during speech recognition, and the lack of clearly marked sentence boundaries. Normal speech is imperfect it contains misspoken words, incomplete sentences, restarts, pauses and both human and non human noises. Coping with spontaneous spoken speech involves, if then, a large search space.

1.2. Outline of Work

My work consists of using domain knowledge to reduce ambiguity and improve end to end performance in a multi-lingual spoken language translation system. The work involves a specific restricted domain that of two people scheduling a meeting. I take advantage of domain knowledge, such as current date and context, to help reduce the difficulties of spoken language and errors introduced during speech recognition.

Due to the limitations of current recognition and parsing technology my system uses the recognizer lattice (see Chapter 3) rather than the top-best hypotheses. In this way information obtained by the recognizer is not lost and more than just the best hypotheses is considered.disambiguation is forced if then to a later state. Parsing a word lattice involves finding a path of connecting words within the lattice that is grammatical. It is my belief that even when *perfect* recognition spoken speech is ambiguous in a way that still more than just one hypotheses should be involved.

The techniques described in this paper were essential to help prune the large number of ambiguities involved.

The techniques involve statistical models as well as knowledge-based models, resulting in a robust end to end speech translation system.

Chapter 2

Overview and Related Work

2.1. Introduction

A traditional speech translation system has a few stages - a recognition stage a parsing stage and a discourse processing stage.

In the next sections I will go over the following modules and the current techniques they use.I will emphasize on how discourse context and domain specific knowledge could be incorporated at each stage to achieve robustness, and on how the different modules could interact.

2.2. Language Modeling

Language modeling is the attempt to characterize, capture and exploit regularities in natural language.

A language model assigns a probability value to every string of words $w_1, w_2, \ldots, w_n$ taken from the prescribed vocabulary. In speech recognition, this value is interpreted as the a priori probability that the speaker will say that string.These probabilities guide the search of the recognizer among various (partial) text hypotheses and are a contributing factor in determining the final transcription.

In statistical language modeling, large amounts of text are used to automatically determine the model's parameters, in a process known as *training*.

In *automatic speech recognition* language modeling plays an important rule, because more knowledge most be brought to bear on the recognition process.

2.2.1. Statistical Language Modeling

In speech recognition, an acoustic signal A is given, and the goal is to find the linguistic hypothesis L that is most likely to have given rise to it. Namely we seek the L such that :

$$\arg\max_L \Pr(L \mid A) = \arg\max_L \Pr(A \mid L) \times \Pr(L) \tag{2.1}$$

Where $\Pr(L)$ is given by the language model.

2.2.2. Conventional N-grams

Using elementary rules of probability theory, the language model's probability $\Pr(L)$ can be formally decomposed as

$$\Pr(L) = \prod_{i=1}^{n} \Pr(w_i \mid w_1, \ldots, w_{i-1}) \tag{2.2}$$

Where $Pr(w_i \mid w_1, \ldots, w_{i-1})$ is the probability that w_i will be spoken given that words $w_1, \ldots, w_{i-1}$ were said previously. For even moderate values of i , $\Pr(w_i \mid w_1, \ldots, w_{i-1})$ would be impossible to estimate.different conditioning histories, if then, must be distinguished as belonging to some manageable number of different equivalence classes.

A N-gram language model makes the assumption that only the previous N-1 words have any effect on the probabilities for the next word. Hence a 3-gram (trigram) takes the following form:

$$\Pr(w_1, \ldots, w_n) \cong \Pr(w_1) \Pr(w_2 \mid w_1) \prod_{i=1}^{n} \Pr(w_i \mid w_{i-2}, w_{i-1}) \tag{2.3}$$

The bigger the N is the greater the differencing power of the corresponding N-gram is . But, the greater N is more training data is needed to provide a reliable model.

The N-gram models are easy to implement and are easy to interface to the application. They seem to capture well short term dependencies and surprisingly difficult to improve on [4]. The two obvious limitations of N-grams are:

- They are completely "blind" to any phenomenon outside of their limited scope.
- No linguistic role is taken into consideration.

2.2.3. Class Based N-grams

In order to reduce the parameter space spanned by N-gram models, words could be clustered into classes. If $g(w)$ denotes the class a word w is assigned, then $\Pr(w_i \mid w_{i-2}, w_{i-1})$ in 2.3 could be estimated as :

$$\Pr(w_i \mid w_{i-2}, w_{i-1}) \cong \Pr(w_i \mid w_{i-1}, g(w_{i-2})) \tag{2.4}$$

Clustering could be done by :

- By linguistic role.
  (Part Of Speech, for example).
- By Domain Knowledge.
  (Example:days of week, restaurant names, ...).
- Automatically derived by statistical means.

2.2.4. Syntactic Constraints

Syntactic knowledge could be incorporated by using a grammar, which ensures that no syntactically or semantically inaccurate words are matched in the speech signal.

ATE labs, developed a speech recognition algorithm using Hidden Markov Models (HMMs) and predictive LR parsing [17]. Predictive LR parsing is an extension of generalized LR parsing, and makes it possible to predict phonemes in speech according to a context-free grammar.Predicted phonemes are then verified by using corresponding HMMs.

In the MINDS system [12, 13], a dynamically constructed lexicon and grammar are used to control the search for words in the acoustic phonetic lattice of alternate segmentations and features. They use the grammar to compute the semantic word categories which could appear in a certain

segment of speech and expanding these concepts into words which are contained in a dynamically generated lexicon. The parser accesses all the word models for those words and selectively calls the word matcher with word models and locations in the speech signal. In this manner, the parser forms a rank-ordered set of phrase hypotheses to span the utterance.

2.2.5. Incorporating Past Information

Long distance

Rosenfeld [11] worked on capturing the information present in the longer distance history. He introduced the notion of a *trigger pair* - a pair of words ,(A,B), were A is significantly correlated with B. When A occurs in the past history it triggers B causing it's probability estimate to change. In his work a set of trigger pairs was derived from a large corpus, Each trigger pair partitioned the history into two classes, based on whether the trigger ,A, occured or did nor occur in it.
He noted that *self triggers* (that means triggers of the form (A,A)), are particularly powerful and robust.
When interfaced with a speech system his work reduced error rate by 10%-14%.

Context

Context could be any of semantic, pragmatic or discourse knowledge.

In the MINDS system [12, 13], predictions derived from the problem- solving dialog situation to limit the search space of the recognition.
At each point in the dialog, they asses the possible states a user could logically progress toward in the next utterance.Associated with each state is a static set of concepts which may or may not be applicable during a specific problem solving session with the system.Thus, the concepts associated with each state are evaluated in light of the current context and the constraints derived from previous dialog information to dynamically create a set of concepts that may be expressed in the next utterance. The result is a dynamically constructed semantic network grammar, which reflects all the constraints derived from the knowledge sources, (see Section 2.2.4).

The predictions are derived from the set of concepts which the user could logically mention to further either progress toward their current goal or their understanding of prior answers.

2.3. Combining Information Sources

When considering a new word, w, there might be different knowledge sources $(h_1, h_2, \ldots, h_n)$ that you want to take into consideration. There are a few ways to combine them all , I will discuss the two basic ones following.

Linear Interpolation

Given n sources of knowledge - $h_1, h_2, \ldots, h_n$, and the appropriate conditional probabilities - $\Pr(w \mid h_1), \Pr(w \mid h_2), \ldots, \Pr(w \mid h_n)$ they can be combined linearly with:

$$\Pr(w \mid h_1, h_2, \ldots, h_n) \cong \sum_{i=1}^{n} \lambda_i \Pr(w \mid h_i) \tag{2.5}$$

Where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.
Any schema for choosing the weights should work, though an Estimation-Maximization type algorithm could be used to determine the optimal weights for a specific test data.

Linear interpolation is easy implement and extremely general - any model can be used as a component. And it cannot hurt, the interpolated model is guaranteed to be no worse than any of its components. The substantial draw-back of this schema is that the different information sources are consulted "blindly", without regard to their strengths and weaknesses in particular contexts.

Back-off

In the back-off method, the different information sources are ranked in order of detail or specificity.At run time, the most detailed model is consulted first.
The back-off method does not actually reconcile multiple models, it chooses among them.
A trigram-backoff model takes the following form :

$$\Pr(w_n \mid w_{n-1}, w_{n-2}) = \begin{cases} k_3 \Pr(w_n \mid w_{n-1}, w_{n-2}) & \text{if } C(w_{n-2}, w_{n-1}, w_n) > 0 \\ k_2 \Pr(w_n \mid w_{n-1}) & \text{if } C(w_{n-2}, w_{n-1}, w_n) = 0 \\ & \text{and } C(w_{n-2}, w_{n-1}) > 0 \\ k_1 \Pr(w_n) & \text{otherwise} \end{cases} \tag{2.6}$$

Where k_1, k_2, k_3 are factors that depend on the counts C and assure that the probability P when summed over all words adds up to 1.

2.4. Mutual Information

There are many potentially useful information sources in the history of a word. A way to asset their potential before incorporating them into the language model in desired.
Such a measure is the *mutual information*, which is defined as the reduction in entropy once the information source is known.

$$I(X;Y) \stackrel{\text{def}}{=} H(Y) - H(Y|X) = \sum_{x,y} \Pr(x,y) \log \frac{\Pr(x,y)}{\Pr(x)\Pr(y)} \tag{2.7}$$

The mutual information ,$I(h;w)$, is the information history, h, provides about the word, w.

2.5. Perplexity

Perplexity is a measure of language model quality.
Given a language model, the per word difficulty of the recognition of speech generated by a given

(large) text ,$w_1, \ldots, w_n$, is measured by it's logprob:

$$LP = -\frac{1}{n} \log \overline{\Pr}(w_1, \ldots, w_n) \tag{2.8}$$

Where $\overline{\Pr}$ is the estimate the language model provides the recognizer. The perplexity , PP, of a language model relative to a text ,$w_1, \ldots, w_n$, is defined as :

$$PP \stackrel{\text{def}}{=} 2^{LP} = \left[\overline{\Pr}(w_1, \ldots, w_n)\right]^{-\frac{1}{n}} \tag{2.9}$$

Thus, the task of recognizing the given text with a language model is as difficult as would the recognition of a language with PP equally likely words.Perplexity is therefore a measure of the average "branching" of the text when presented to the language model. To compare two language models, you will prefer the one that gives a lower perplexity when tested on the same representative text.

2.6. Parse disambiguation

Almost any natural language sentence is ambiguous in it's structure or meaning. When a sentence has more than one parse one would like to know which parse was intended. The following are some ways to achieve this.

2.6.1. Probabilistic Parsers

The motivation of probabilistic parsing is the assumption that a more probable parse is more likely to be the correct parse of an input sentence.
Probabilistic parsers give an ordering of the parses by assigning each one a probability.
In a probabilistic grammar, each rule has a probability assigned to it. (One simply counts the number of times each rule is used in a corpus containing parsed sentences). The probabilities for all the rules that expand the same non-terminal must some to one. An algorithm is used to find the most likely parse tree that could have generated a given sentence.
The technique involves making certain independence assumptions about rule use. In particular, you must assume that the probability of a constituent being derived by a rule R is independent of how the constituent is used as a subconstituent.For example, this assumption would imply that the probabilities of NP rules are the same whether the NP is a subject, the object of a verb, or the object of a preposition.
A probabilistic parser, if then, assigns higher probability to parses that use common constructions than to those with less common ones.

2.6.2. History-based Grammars

At IBM [3] a model incorporating lexical, syntactical, semantical and structural information was composed to help parse disambiguation. Their motive was that humans overcome these ambiguities by examining the context of the sentence.
Decision trees were used to allow any information anywhere in the partial derivation tree to determine the probability of different expansions of a non-terminal. They report a 25% increase in parse accuracy.

2.6.3. Semantic and Pragmatic Knowledge

In the MINDS-II project [14, 15], Pragmatic and semantic knowledge was used to correct or reject parses of spoken language utterances.
The system corrects the following types of problems:

- Information that is missing from the output of the recognizer.
- Constraint violations.
- Inaccurate parses.
- Unanswerable queries and commands.

When the system receives a query involving information not included in the current, restricted database, out of the chosen domain or when the user is asking the system to perform a function it is not designed to do - it outputs specific error codes and specific corrective information to the user.

2.6.4. Context

Context other than the current sentence and it's tree structure exist. This includes the last sentences and their structure. From a probabilistic point of view the parse tree T for which (2.10) holds is desired.

$$T = \arg \max_{T \in \Im(S_n)} \Pr(S_1, T_1, \ldots, S_{n-1}, T_{n-1}, S_n) \tag{2.10}$$

Where $S_1, \ldots, S_{n-1}$ are the different sentences and $T_1, \ldots, T_{n-1}$ are their parse trees respectively.And $\Im(S)$ is the set of all parses produced by the grammar for the sentence S.

However how to fully exploit this information is still an open field of research.

2.7. Discourse Processing

As noted in Sections 2.6.4 and 2.2.5, a good discourse processor could help improve the overall performance of a speech recognition system.

2.7.1. Conventional

Many researchers have contended that a coherent discourse consists of segments that are related one another through some type of structuring relationship. To quote a few - Man and Thompson, Hobbs, Polani, Allen, and Grosz and Sidner. Most of the above do not consider *plan inference*, model goals or extended discourse. Furthermore are limited to handle only specific types of discourse. One of the main limitations of the current systems is that many of them are theoretical in nature and have no implementation.
The work of Lambert [7] presents a plan-based model for understanding cooperative negotiation dialogues.It infers both the communicative actions that people pursue when speaking and the beliefs underlying them, and identifies the relationship of utterances to one another.

2.7.2. Statistical Approaches

ATR [8], proposed a purely statistical model of dialogue based on an information-theoretic interpretation of a discourse. Their model predicts the illocutionary force type of the next utterance.It consists of a second order Markov model of utterances classified by their illocutionary force type, such as *request, inform, ...*
They report a 60% percent accuracy in prediction.

Both approaches currently show poor results.
A combined system which incorporated statistical techniques is strongly suggested.

2.8. Verbmobil

Verbmobil is speech to speech translation system, translating spoken dialogs between two persons who want to find a date for a business meeting. The system uses *top-down* predictions of the speech-act to narrow down the set of words which are likely to occur in the following utterance.Top down predictions are also used to limit the set of applicable grammar rules to a specific subgrammar. To use contextual knowledge a discourse history is maintained. A combined statistical and knowledge-based model is used to predict the next speech-act where the statistical unit is used as a backup.

2.9. Lattice Parsing Algorithms

A word lattice is a set of hypothesized words with different starting and ending positions in the input signal. Parsing a word lattice involves much more search than conventional typed natural language parsing, and a very efficient algorithm is desired. Tomita ([9]) offers an efficient LR lattice parsing algorithm, where the LR parsing tables are utilized. The algorithm is based on a Generalized LR style substring parser, that can parse an input string in arbitrary order. An efficient computation strategy is achieved by using an A* heuristic to determine the order in which words of the lattice are parsed.

Chow and Roukos ([19]) describe a bottom-up CYK style parsing algorithm that does not suffer from the uni-directionality restriction. The algorithm uses Dynamic Programing and Chart Parsing techniques in order to parse the word lattice and find the highest scoring grammatical path.

I don't know of any running system that uses the above implementation.

Chapter 3

Framework

3.1. Janus

Janus is a speech-to-speech translation system for dialogs in the scheduling domain (two people scheduling a meeting with each other). The main modules of Janus are speech recognition, parsing, discourse processing, and generation. A system diagram is shown in Figure 3.1.

Processing starts with speech input in the source language. Recognition of the speech signal is done with acoustic modeling methods, constrained by the language model. The output of speech recognition is a list of the N-best sentence candidates or a word lattice. The current system uses only the best hypothesis from the N-best list which is then sent to the translation components of the system. The system currently does not use the word lattice.

3.2. the GLR* parser

Translation begins with analysis by the GLR* parser ([2]) . The GLR* parser skips parts of the utterance that it cannot incorporate into a well-formed structure. Thus it is well-suited to domains in which extra-grammaticality is common. Multi-sentence conversational turns are broken into separate sentences automatically during parsing with the help of a statistical method that determines the probability of sentence breaks at each point in the utterance. The output of parsing is an interlingua, or ILT, which is intended to be a language-independent representation of meaning.

The parser has an attached statistical module in which shift and reduce actions of the LR parsing tables are directly augmented with probabilities. Training of the probabilities is performed on a set of disambiguated parses. The probabilities of the parse actions induce statistical scores on alternative parse trees, which are then used for parse disambiguation. The GLR* parser used 1050 sentences from 1440 sentences in the 30 dialog training set for training.

Currently only the best scoring ILT is sent on for farther processing.

3.2.1. ILTs

The core of the translation system is the interlingua, which is intended to be a languageindependent representation of meaning. Multi-sentence conversational turns are assumed to be broken down into separate sentences or or sentence fragments represented by a list of ILTs. The interlingua for the sentence :

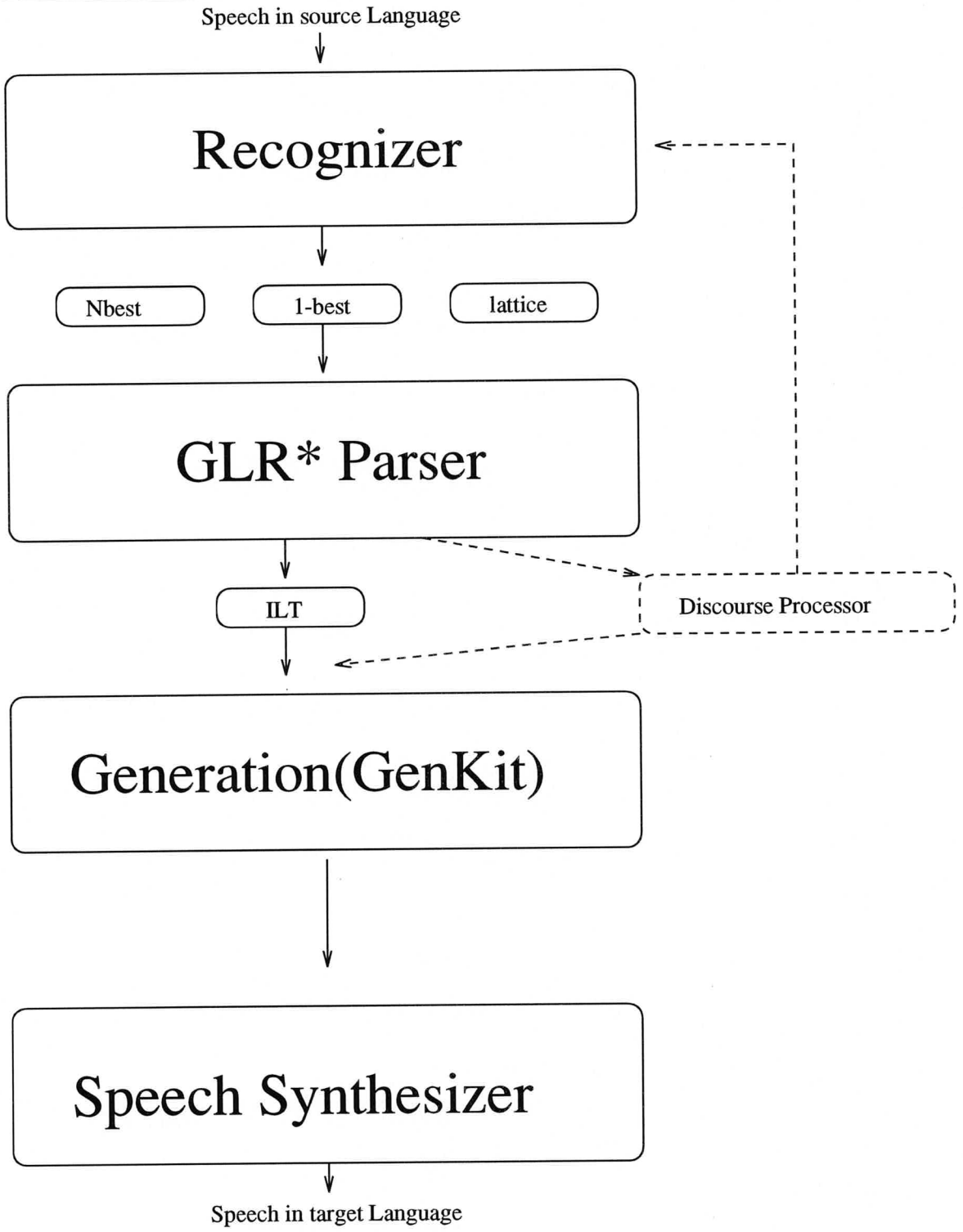


Figure 3.1 Janus System

"I have something ten to noon".

is:

```
[ SPEECH-ACT *state-constraintSENTENCE-TYPE *state
  FRAME *booked
  WHO [FRAME* I]
  WHAT [FRAME *something]
  WHEN [ FRAME *interval
         START [ FRAME *simple-time
                 HOUR 10 ]
         END   [ FRAME *simple-time
                 HOUR 12
                 AM-PM pm ] ]
```

3.2.2. The GLR* scoring mechanism

The parser assigns each ambiguity a score by combining the following features:

1. The number and position of skipped words.
2. The fragmentation of the parse analysis.
3. The statistical score of the disambiguated parse tree.

The penalty scheme for skipped words is designed to prefer parses that correspond to fewer skipped words.

3.3. Discourse Processing

[1] After parsing, the ILT is modified by the discourse processor. The discourse processor, disambiguates the speech act of each sentence, normalizes temporal expressions, and incorporates the sentence into a discourse plan tree. The discourse component also updates a calendar in the dynamic discourse memory to keep track of what the speakers have said about their schedules. Once the ILT is fully specified, it can be sent to the generator to be rendered in the target language.

3.4. Databases

The Spanish scheduling database used consists of 150 *push-to-talk* dialogs and and 150 *cross talk dialogs* totaling in 300 dialogs or 5,472 utterances. Analysis training and development was done on a training set of 30 dialogs.*Target ILTs* were hand written for the training set and for a test set of 15 dialogs.

[1] Still not incorporated in the system

Chapter 4

Word lattice Processing

4.1. Introduction

A word lattice is a set of hypothesized words with different starting and ending positions in the input signal. Parsing a word lattice involves much more search than parsing the best hypotheses of the recognizer. parsing a word lattice is equivalent to parsing a large n-best list in an efficient manner. The GLR* parser was modified to parse such lattices. The problem of a very large search space and running time brought me to develop the following tools to pre process the lattice.

4.2. Lattice Cleaning

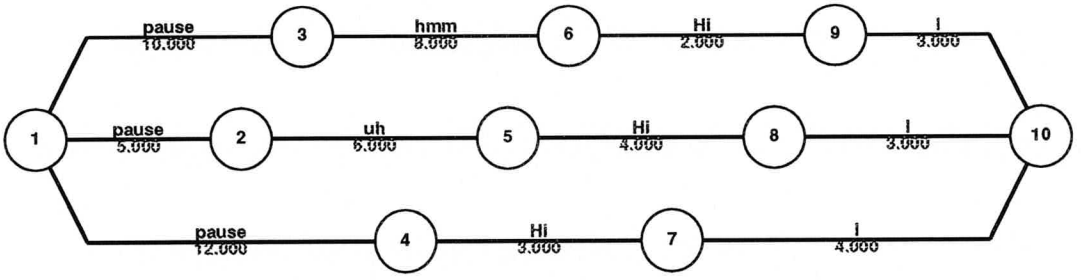
The lattice given by the recognizer is contain many different noise words and contains redundant paths. The *lattice cleaner* maps all non human noises and pauses to a generic pause. Consequent pauses are then adjoined to one long pause. Redundant paths are then removed keeping the highest acoustically scoring one. two paths are redundant if they begin and end in the same nodes, and contain the same sequence of words.

In Figure 4.1 the three paths through the lattice are redundant, and after cleaning only the middle path, that is the highest scoring one, will remain as shown. An important note is that no *linguistic* information is lost in the process of cleaning, any hypothesis existing in the original lattice will appear in it's cleaned version.

4.3. Lattice Breaking

The lattice is broken into smaller lattices, Corresponding(hopefully) to the sentence breaking of the utterance. This is done by breaking the lattice at time points where no human input is recognized in the speech signal. A threshold of time length is set to determine optimal breaking. Where a small threshold might break the utterance at wrong places, and a too big threshold leaves the utterances to long for the current parser to handle. The lattice breaking reduces the complexity of the parsing significantly and forces the parser to break the sentences at certain points and hence avoiding ambiguities as in the utterance: " ...I'm busy Monday and Tuesday I am free ...". Breaking also forces the parser to recover from mistakes in the case that one or more of the sentences in the utterance are out of domain, misrecognized or just not covered by the current state of analysis.

In Figure 4.2 the lattice could be broken into two independent lattices as shown.



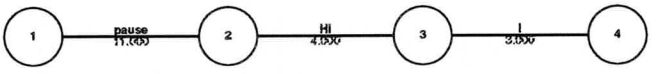
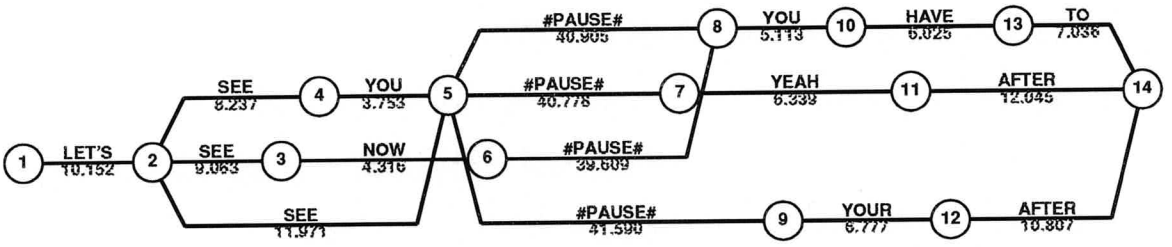
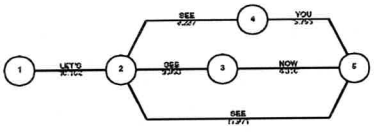


Figure 4.1 Lattice Cleaning - Example





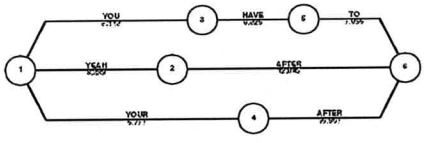


Figure 4.2 Lattice Breaking - Example

4.4. Lattice Re-scoring and Shrinking

The Lattice contains at this point only the linguistically important information and therefore it's vertices could be re-scored now by the same or a new language model. A tool to shrink the lattice was written to leave a lattice in a size we are currently able to handle efficiently. The shrinking process removes the arcs in a decreasing order by the score of the best hypothesis they can contribute to. The re-scoring forces that the correct hypothesis will not be lost shrinking the shrinking stage.

4.4.1. Calendar Knowledge

In the shrinking process the knowledge of the current day and date was used. I found that some words in the vocabulary have different probabilities depending on the current month. For example, if the current month is June, words like *June* and *July* will have higher probabilities and words like *Thanksgiving*, *class*, and *February* will have lower probabilities. A different language model with the above changes was made. Another use of calendar knowledge involved sentences containing phrases like *Sunday the fifth* and *Wednesday December 5*. Using transcribed dialogs, we calculated the percentage of cases in which the day and date actually go together. (When speakers make mistakes, or when the wrong month is used in calculating the date, the day and date might not match.) The probability of the path in the lattice containing a day and a date is altered accordingly from its original probability in the language model. This knowledge is easy to test when you have the lattice representation.

Chapter 5

Triggers

5.1. Introduction

In this chapter I describe an experiment involving modifying the speech recognizer's language model to be sensitive to certain features of context and by this reducing search space. The goal of language modeling, in speech recognition, is to identify and exploit sources of information in the language stream, so as to bring the perceived entropy down as close as possible to its true value. Most current systems use different types of N-gram models to constrain speech. Such models use the last N-1 words as their sole information source. Therefore they are totally blind to any phenomenon that is out of their scope. A way to exploit long-distance information is desired. Work to incorporate long-distance information was done ([6, 11]) where *Trigger Words* were examined and used. Trigger words are words that bear a predictive power on future words. In [6, 11], trigger words were considered in the last M words of text, for a constant M. My work involves dialogs, which are different from text in that there are two speakers. Because of this, the history of a specific word should be accounted for differently depending on the speaker. In my case, therefore, I only look at the last speaker's words.

Examples of trigger words are shown in Figure 5.1. For example, *lunch, restaurant* and *office* could be trigger words that give a higher probability to the words *lunch* and *office* respectively.

Speaker 1: Okay do you wanna go to a restaurant or something, we could have lunch or else we could have it at the office.

Speaker 2: No, I don't wanna go out for lunch because I'll have a brunch meeting just before that, but the office is good.

Figure 5.1 Sample Dialog

5.2. Trigger Pairs

A *trigger pair* is an ordered pair of words (A, B), were A is *significantly* correlated with B. When A occurs in the past history it triggers B, causing its probability estimate to change. To find such

words all word pairs from the lexicon were tested. For a pair $(\text{word}_1, \text{word}_2)$ the average mutual information of the events

$P = \{\text{word}_1 \text{was uttered by previous speaker}\}$ and
$Q = \{\text{word}_2 \text{was spoken by current speaker}\}$

was calculated by Formula (5.1). The probabilities $\Pr(P, Q)$ were smoothed because they might be zero. This enabled me to extend the work in [6] by accounting for negative trigger pairs where the trigger word causes the other word to have a lower probability.

$$I(P;Q) = \Pr(P,Q) \log \frac{\Pr(P,Q)}{\Pr(P)\Pr(Q)} \tag{5.1}$$

Running over the corpus of dialogs, all the pairs with average mutual information greater in absolute value than 0.0001 (where the log is base 10), were listed. Examples from the resulting list are:

- Positive triggers: (bye, goodbye), (lunch, restaurant), (fine, sounds), (first, second), (brunch, O'clock), (where, at) ...
- Negative triggers: (fifth, nineteenth), (goodbye, hello), (O'clock, day), (dinner, morning), (can't, bye), (how, Hi) ...

5.2.1. Method

To show how the trigger pairs can be incorporated into an existing language model, I assume a regular bigram is being used. In a regular bigram you calculate $P(w_i|w_{i-1})$, but we are interested in

$$P(w_i|w_{i-1}, L) \tag{5.2}$$

where L is the previous utterance by the other speaker.

I cluster all such sentences by the set of all trigger words that appeared in them. For example, in the case of the utterance of speaker 1 in the sample dialog, L = {lunch, restaurant, office}. However, due to sparse data Equation (5.2) can not be evaluated directly. The way to overcome this is to notice that the events of the *last utterance*, L, and *previous word*, w_{i-1}, could be assumed to be independent. If this is the case we have:

$$\begin{aligned} \Pr(w_i \mid w_{i-1}, L) &= \frac{\Pr(w_i \mid w_{i-1})\Pr(w_i \mid L)}{\Pr(w_i)} \\ &= \frac{\Pr(w_i \mid L)}{\Pr(w_i)} \times \Pr(w_i \mid w_{i-1}) \end{aligned} \tag{5.3}$$

Equation (5.2) is therefore equivalent to multiplying the regular bigram value by the coefficient $\lambda_{w,L} = \frac{\Pr(w_i|L)}{\Pr(w_i)}$. Once again these coefficients were calculated only for the set of trigger pairs and not for all word pairs due to sparse data. Because there could be more than one trigger in the history, L, a difficultly in computing $\lambda_{w,L}$ occurs. To avoid this for every word only it's best trigger is considered. For each word a constant coefficient, μ was used for all non-trigger pairs.

The coefficient, μ_w, was determined by Equation (5.4) in order to make sure that I end up with a correct probability measure.

$$\begin{aligned} & \Pr(w_i \mid w_{i-1}) = \textstyle\sum_L \Pr(w_i \mid w_{i-1}, L) \Pr(L) = \\ & \quad \textstyle\sum_L \lambda_{L,w} \times \Pr(w_i \mid w_{i-1}) \Pr(L) \\ \Rightarrow \quad & \quad 1 = \textstyle\sum_L \lambda_{L,w} \Pr(L) = \\ & \quad \lambda_{t_w,w} \Pr(t \in L) + \mu_w \Pr(t \notin L) \\ \Rightarrow \quad & \quad \mu_w = \frac{1-\lambda_{t_w,w} \Pr(t_w \in L)}{1-\Pr(t_w \in L)} \end{aligned} \tag{5.4}$$

5.2.2. Self Triggers

Among the positive triggers, the class of *self-triggers* — triggers of the form (A, A) — turns out to be very robust. In fact, for more than a third of the words, the highest trigger turned out to be the word itself.The self-trigger list contains the following categories:

1. months - January, February ...
2. days - Sunday, Monday ...
3. dates - first, second ...
4. times - one, two ...
5. time expressions - tomorrow, noon, PM ...
6. others - Mr., lunch, busy, office, bye ...

An example of the coefficients of the self-triggers can be seen in Table 5.1. The table shows, for example, that when the word *office* appears in an utterance it will get ten times the regular bigram probability if it appearers in the previous utterance as well.

Sunday	15.3	Monday	3.5	Tuesday	3.6
October	9.6	August	11.9	June	10.6
one	3.1	two	1.6	there	3.2
first	5.0	second	5.3	ninth	6.6
tomorrow	22.4	today	15	noon	6.1
bye	7.1	Mr.	26.4	lab	48.0
class	4.5	lunch	3.4	office	9.8
busy	3.4	available	5.3	free	1.4

Table 5.1 Self Triggers - Coefficients

5.3. Results

The technique described above is very easy to incorporate into a running system because it does not change a language model but rather uses it. An existing language model is adapted to different contexts by multiplying it by different coefficients.

Some additional improvement resulted from changing the definition of the history in the ways listed below. Using these improvements, training was done on a collection of 750 scheduling dialogs (155,000 words). Perplexity was then calculated on a test set of 14 dialogs (approximately 3,700 words), taking triggers into account the resulting value was 5% lower than the regular bigram perplexity.

- A beginning history, $L = \{\text{beginning}\}$, which is different from the empty history $\{\}$, was added as the history of the first sentence in a dialog. This causes certain words, like *hi* or *hello*, to have a higher probability in the first sentence of a dialog.

- Once the second member of a trigger pair is encountered, the trigger is removed from the history list. This causes the trigger pair to have a bigger coefficient.

- The same trigger cannot appear in two consecutive history lists. For example, *lunch* will be in the history list obtained from Speaker 1 in the dialog of Figure 5.1, but it will not be in the history obtained from Speaker 2.

Chapter 6

Parse Disambiguation

6.1. Introduction

When working on lattices the system needs to choose between parses due to differ ant recognition hypotheses, and not just between ambiguities coming from the same hypothesis. A good and robust scoring mechanism is needed, if then, to distinguish between such ambiguities.

6.2. Word skipping penalties

The GLR* Parser skips parts of the utterance that it cannot incorporate into a well-formed structure. Words are assigned weights as a skipping penalty. In English a word's penalty is computed by comparing it's frequency in our domain, to it's frequency in other domains, and hence a higher skipping penalty for domain specific words. For Spanish, a weighting heuristic that doesn't depend on an existence such of a general corpus was needed.

A function to compare two ILTs was derived that gives you a score rather than equal/not equal result. To do so The possible slots in an ILT where weighted by importance. An example of a few weights can be seen in Table 6.1. In Figure 6.1 you can see how these weights are used to return the resulting score. Alignment is then used to compare sequences of ILTs corresponding to full turn output.

To calculate word skipping penalties the parser was applied over all sentences of the training set and the words where counted for if they appeared among the skipped words of the parse giving the **highest** score when compared to the target ILT. Each word is then assigned a skipping probability which is :

$$\Pr(\text{w skipped}) = \frac{\text{number of times w skipped}}{\text{number of times w appeared}}$$

Note that it is essential to use also non perfect matches for they usually do not require any skipping.

SENTENCE-TYPE	10
FRAME	30
WHO	25
WHAT	10
WHEN	25
HOUR	30

Table 6.1 ILT slot weights

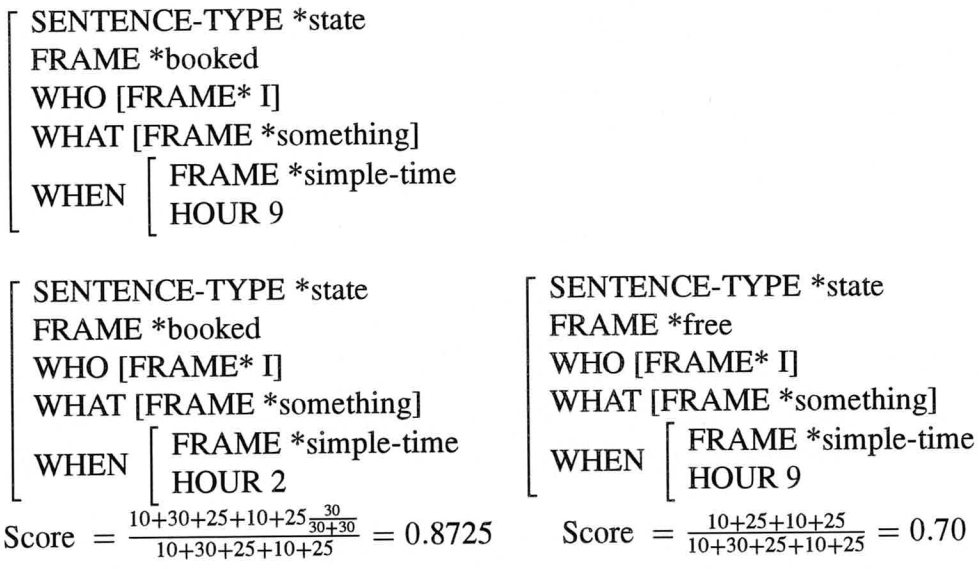


Figure 6.1 Comparing ILTs

6.3. Full Turn Disambiguation

As I described above, the parser uses statistical scores to choose the most likely parse based on sentence structure without taking the context of surrounding sentences into account. In this section I describe a statistical approach that uses context to help parse disambiguation. This work involved assigning a probability to a full turn parse given information from previous ILTs.

Given a full turn hypothesis — ILT_1, ILT_2, ... ILT_n
it could be assigned a probability by Equation 6.1.

$$\Pr(\text{ILT}_1,\ \text{ILT}_2,\ \ldots\ \text{ILT}_n) = \Pr(\text{ILT}_1) \times \Pr(\text{ILT}_2 \mid \text{ILT}_1)\ \ldots \Pr(\text{ILT}_n \mid \text{ILT}_{n-1}) \qquad (6.1)$$

Where any n-gram approach could be used and Probabilities could be derived from the training set of target ILTs.

My preliminary work involved checking the `sentence-type` and top-level `frame` of the ILTs and using bigram probabilities.

Note that in this current stage of the system no unique speech-act is assigned to the ILTs and hence cannot be used to assign probabilities. The amount of training data was not sufficient to calculate more complex N-grams such as

$$\Pr(\text{ILT}_n \mid \text{ILT}_{n-1}\ ,\ \text{ILT}_{n-2}) =$$
$$\Pr(\text{sentence-type}_n\ ,\ \text{frame}_n \mid \text{sentence-type}_{n-1}\ ,\ \text{frame}_{n-1}\ ,\ \text{sentence-type}_{n-2}\ ,\ \text{frame}_{n-2})$$

Currently information from the past (previous speakers) turn, is not taken into account.

6.4. Language Modeling of Generation Output

The output from generation could be assigned a probability as well. The generation output follows certain forms and is restricted in style. Therefore a regular n-gram model could be applied to assign a probability to each ambiguity of the system. The assumption here is that strange generation output comes from "strange" ILTs, which are allowed by the grammar roles, but are not correct. The training set for the bigram was the generation output of all English and Spanish training set target ILTs. In order not to give preference to shorter sentences, the average word probability was calculated and compared for every output sentence.

Chapter 7

Results

7.1. The integrated model

The heuristics described were integrated in a system that is outlined in Figure 7.1. Note that in the new model no information is lost, the parser gets a processed lattice and it passes it's best hypotheses on for disambiguation. After post generation processing the system has a list of target language hypotheses and the following score for each one of them:

1. acoustic score.
2. parser's score.
3. Full turn probability.
4. Generation Language modeling probability.

All the above are combined with hand set weights and a final score is assigned to each hypothesis.

7.2. Evaluation

To test the **end-to-end** performance, the integrated system was run over a set of 3 dialogs (1 *push-to-talk* and 2 *cross-talk*) with a total of 62 utterances. All data was unseen by recognition or analysis. The output was compared to the current systems output and graded by a Spanish and English speaker. The grader was asked to grade each utterance as perfect,acceptable or bad.

7.3. Results

	My system	My system no domain knowledge	Current System	Phoenix
perfect matches	13	12	11	6
acceptable	19	17	15	15
performance	52%	47	42%	34%

Where the ouput of my system using lattices and combining acoustic score and parse score was added for comparesment.

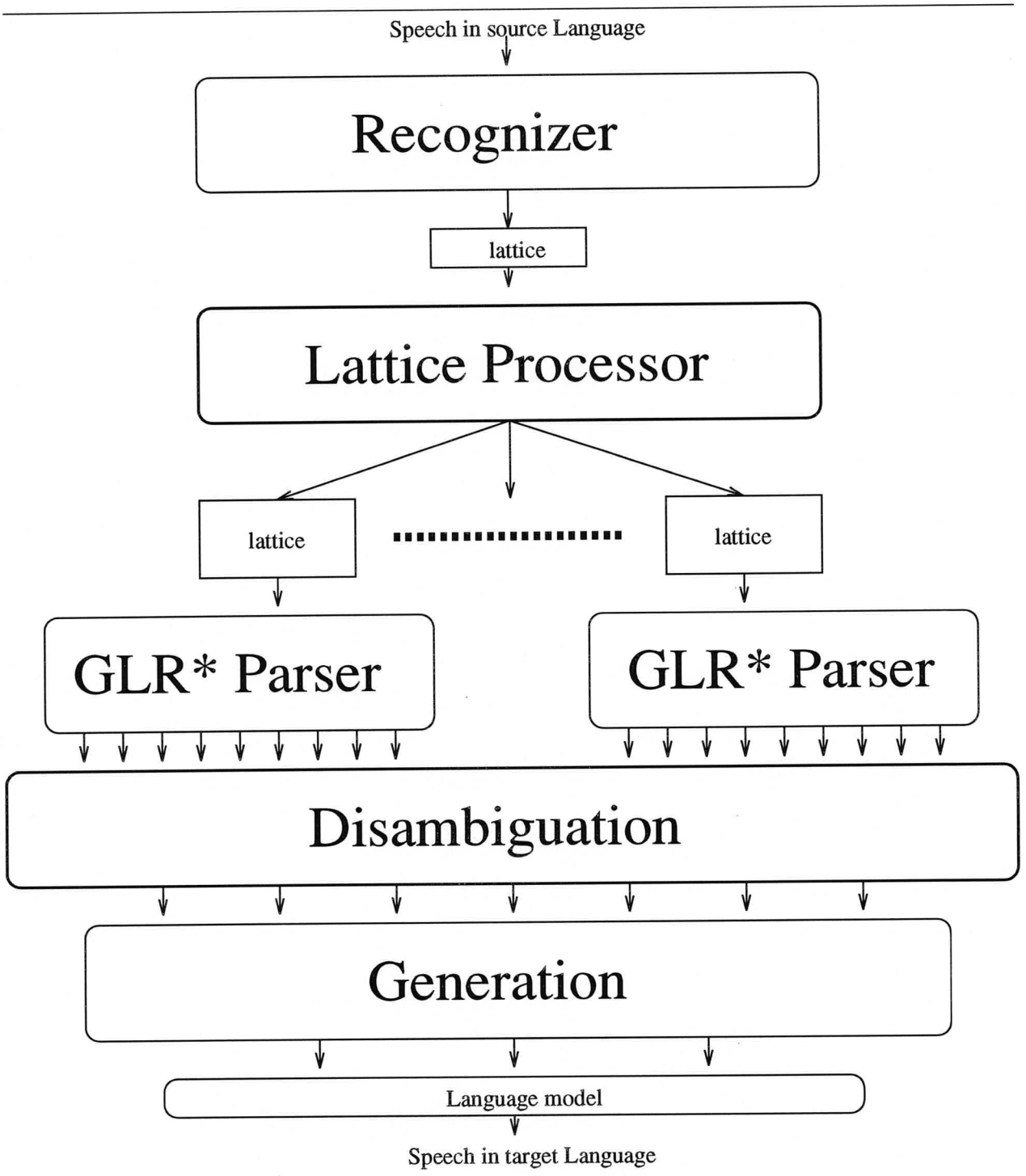


Figure 7.1 System Diagram

7.4. Conclusion

A nice improvment of 24% increase in system performance was atchieved. One needs to take into consideration the current limits of recognition,analysis and generation to fully understand the reported results. My estimate is that only in around 60% of the shrunk lattices an acceptable path exists.

Another Problem arises from the fact that the acoustic score and parse score are not probabilities but just penalty scores and that linear weights to combine the different scores might not be effective.

Chapter 8

Future Work

My system takes advantage of the recognizer's graph representation of hypotheses, to parse efficiently a set of hypotheses rather than the top-best one. Currently the *post-processing* stage works on the list of hypotheses given by the parser. In my future work I am interested in modifying the parser so to get a graph representation of the different hypotheses. An example of such a `Lattice of ILTs` can be seen in Figure 8.1. Such a representation will be very time efficient and easy to process farther on.

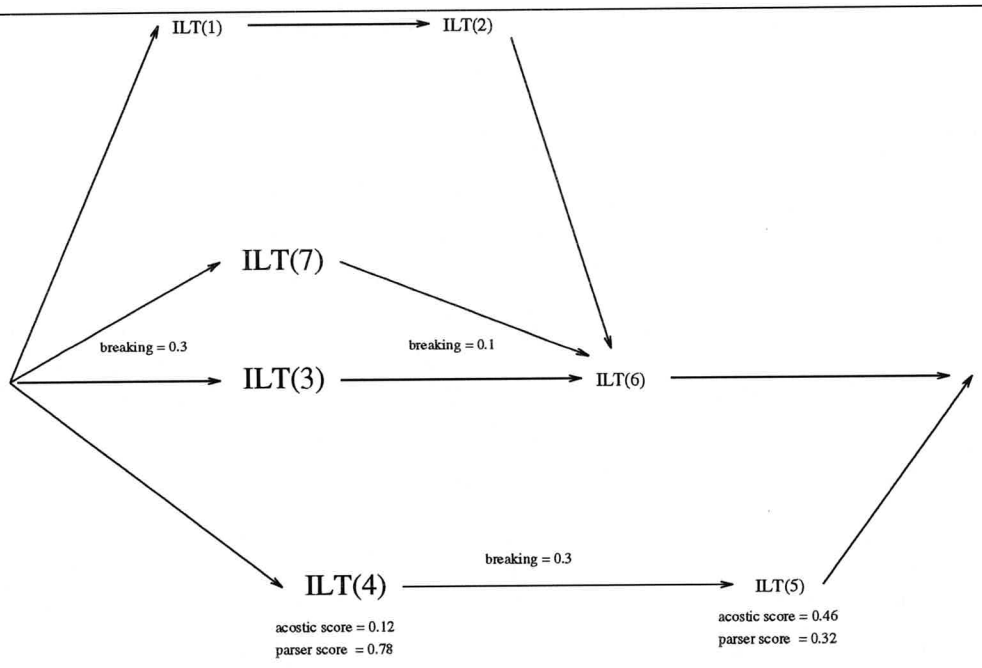


Figure 8.1 Lattice of ILTs

Appendix A

Output From the system

A.1. Example 1

Transcribed:

pos c {uh} cua'ndo cua'ndo esta's libre ?

Recognizers top best:

+NONHUM+ PODER +H#+ SIL +LS+ QUE+ HACER ESTE ESTA+S LIBRE PARA SIL PARA COMER

```
(%3% (0 1) 0.973)
(%4% (0 2) 1.265)
(%4% (0 3) 1.714)
(PUES (1 4) 11.295)
(PUES (1 5) 14.345)
(PUES (1 6) 14.345)
(POS (2 5) 13.728)
(POS (2 6) 13.728)
(POS (2 7) 13.728)
(TODOS (3 7) 13.614)
(%17% (4 8) 6.279)
(%11% (5 8) 3.188)
(%11% (6 8) 3.292)
(%11% (7 8) 2.846)
(TODO (8 9) 14.871)
(CUA1NDO (9 10) 13.52)
(CUA1NDO (9 11) 13.52)
(COSAS (10 13) 15.852)
(COSAS (10 14) 15.852)
(CO1MO (11 12) 8.549)
(ESTA1S (12 14) 8.702)
(DE (13 15) 3.806)
(DE (13 16) 4.542)
(DE (13 17) 5.694)
(LIBRE (14 18) 14.076)
(ABRIL (15 18) 10.869)
(HORAS (16 18) 9.735)
(UNA (17 18) 8.624)
(%1% (18 19) 0.831)
```

"WELL... WHAT IS GOOD ?"

A.2. Example 2

Transcribed:

pos a comer. a do'nde quieres ir a comer ?

Recognizer:

POS A COMER DOS DE QUE ES QUE COMER +NONHUM+

```
(%13% (0 1) 3.802)
(POS (1 2) 6.224)
(A (2 3) 3.278)
(COMER (3 4) 10.282)
(COMER (3 5) 10.971)
(COMER (3 6) 11.825)
(DO1NDE (4 12) 9.015)
(DOS (5 8) 4.945)
(O (6 7) 4.102)
(DE (7 10) 3.342)
(DE (7 11) 3.342)
(DE (8 9) 2.991)
(DE (8 10) 2.991)
(DE (8 11) 2.991)
(QUE1 (9 13) 5.213)
(TRES (10 16) 10.828)
(QUE (11 13) 5.298)
(QUE (11 14) 6.265)
(QUE (11 15) 6.265)
(QUIERES (12 16) 9.701)
(QUIERES (12 18) 9.701)
(ES (13 16) 5.017)
(ES (13 17) 5.017)
(SEA (14 20) 6.782)
(SER (15 20) 6.899)
(SER (15 21) 7.277)
(DE (16 21) 3.56)
(QUE (17 21) 3.769)
(IR (18 19) 2.623)
(A (19 21) 1.843)
(A (20 22) 1.491)
(COMER (21 23) 15.343)
(COMER (22 23) 14.894)
(%24% (23 24) 7.95)
```

"WELL..." " EAT WHERE YOU WOULD LIKE TO EAT"

A.3. Example 3

Transcription:

> #beep# #rustle# si'. me parece bien, a la a las dos en punto. porque a las cuatro tengo que #begin-mike-noise# que salir del del trabajo. #end-mike-noise#

Recognizer:

> +LS+ SI+ ME PARECE BIEN A LAS +HUMAN+ A LAS DOCE EN PUNTO PORQUE A LAS CUATRO TENGO QUE +H#+ +HUMAN+ QUE ESTA+ BIEN EL TRABAJO +NONHUM+ +CLICK+ SIL +CLICK+ SIL +LS+ SIL +CLICK+ +NONHUM+ +CLICK+

```
(%95% (0 1) 23.973)
(SI1 (1 2) 8.979)
(ME (2 3) 3.943)
(PARECE (3 4) 14.219)
(BIEN (4 5) 9.247)
(BIEN (4 6) 11.292)
(BIEN (4 7) 11.292)
(BIEN (4 8) 11.697)
(NO (5 10) 5.28)
(A (6 11) 3.365)
(A (6 12) 3.642)
(A (6 13) 3.642)
(A (6 14) 3.642)
(A (7 9) 2.1565)
(A (7 16) 4.6595)
(%73% (8 20) 24.819)
(EL (9 15) 2.1565)
(EL (10 17) 6.378)
(EL (11 17) 6.099)
(%63% (12 20) 21.609)
(LAS (13 19) 8.119)
(LA (14 18) 7.064)
(%61% (15 20) 20.442)
(EL (16 17) 4.6595)
(%45% (17 20) 15.504)
(%42% (18 20) 14.547)
(%39% (19 20) 13.606)
 Parse of input utterance :
   (SI1 ME PARECE BIEN $)
(%45% (0 1) 15.504)
(A (1 2) 6.296)
(LAS (2 3) 7.101)
(LAS (2 4) 7.101)
(DOCE (3 6) 8.866)
(DOS (4 5) 8.137)
(EN (5 7) 5.65)
(EN (6 7) 4.82)
(PUNTO (7 8) 21.47)
(PORQUE (8 9) 13.549)
```

```
(PORQUE (8 10) 14.387)
(A (9 11) 1.557)
(LAS (10 12) 7.692)
(LAS (11 12) 7.261)
(CUATRO (12 13) 17.401)
(TENGO (13 14) 10.869)
(QUE (14 15) 8.857)
(%50% (15 16) 18.178)
 Parse of input utterance :
   (A LAS DOS EN PUNTO PORQUE A LAS CUATRO $)
(%50% (0 1) 18.178)
(%51% (0 2) 18.673)
(QUE1 (1 3) 4.657)
(QUE1 (1 4) 4.657)
(QUE1 (1 7) 5.704)
(QUE (2 5) 4.536)
(QUE (2 6) 4.536)
(QUE (2 8) 5.555)
(QUE (2 9) 5.555)
(ES (3 12) 9.042)
(ESTA1 (4 10) 8.434)
(ES (5 12) 8.731)
(ESTA1 (6 10) 8.122)
(SE (7 11) 7.623)
(SE (8 11) 7.202)
(SER (9 13) 10.11)
(BIEN (10 16) 10.454)
(BIEN (10 18) 11.653)
(BIEN (10 19) 23.07)
(VIVEN (11 20) 22.451)
(MI (12 14) 7.244)
(MI (12 16) 9.264)
(MI (12 17) 9.713)
(EN (13 19) 20.393)
(%4% (14 15) 1.993)
(EN (15 19) 13.237)
(EN (16 19) 12.805)
(%33% (17 19) 12.541)
(%31% (18 19) 11.678)
(EL (19 21) 5.834)
(DE (20 21) 5.996)
(TRABAJO (21 22) 21.217)
(%116% (22 23) 38.164)
 Parse of input utterance :
   (QUE ESTA1 BIEN $)
```

"YES" "THAT IS GOOD FOR ME" "EXACTLY TWELVE O+CLOCK" "BECAUSE FOUR O+CLOCK" "THAT IS GOOD"

Appendix B

Lattice Proccessing Example

Example of the lattice proccessing on the lattice given for the sentence :

> {h} twenty seventh, {uh} I have a seminar from nine thirty to four thirty.{um} I take it the earl, any earlier in that week isn't good for you. {h} let me see,{um} #microphone# you're out through June second ? did you say ? #key-click#.

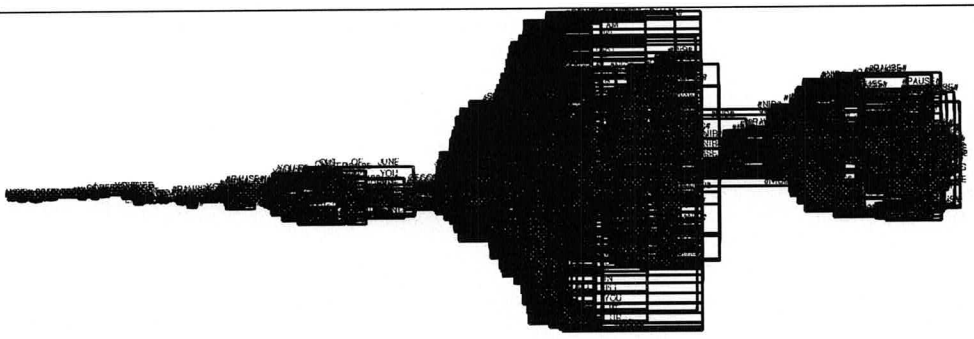


Figure B.1 Example lattice

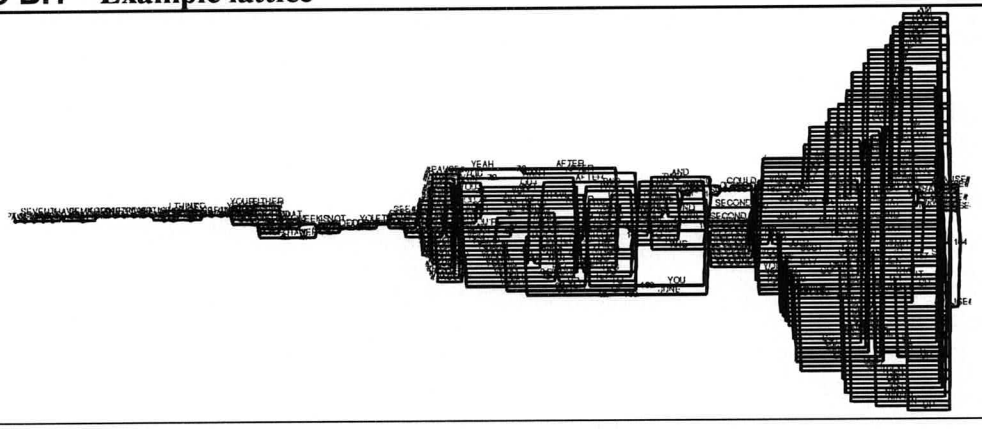


Figure B.2 Clean lattice

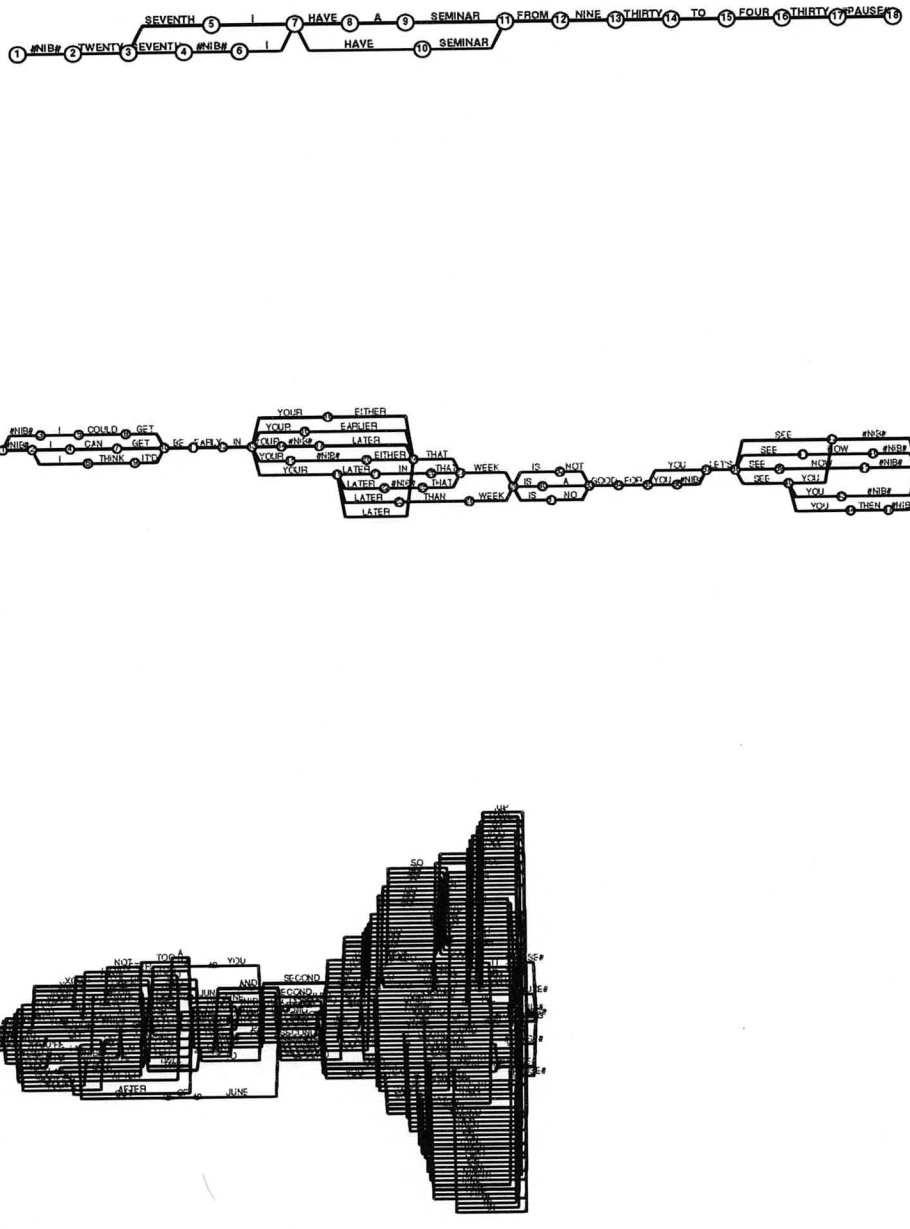


Figure B.3 Broken lattice

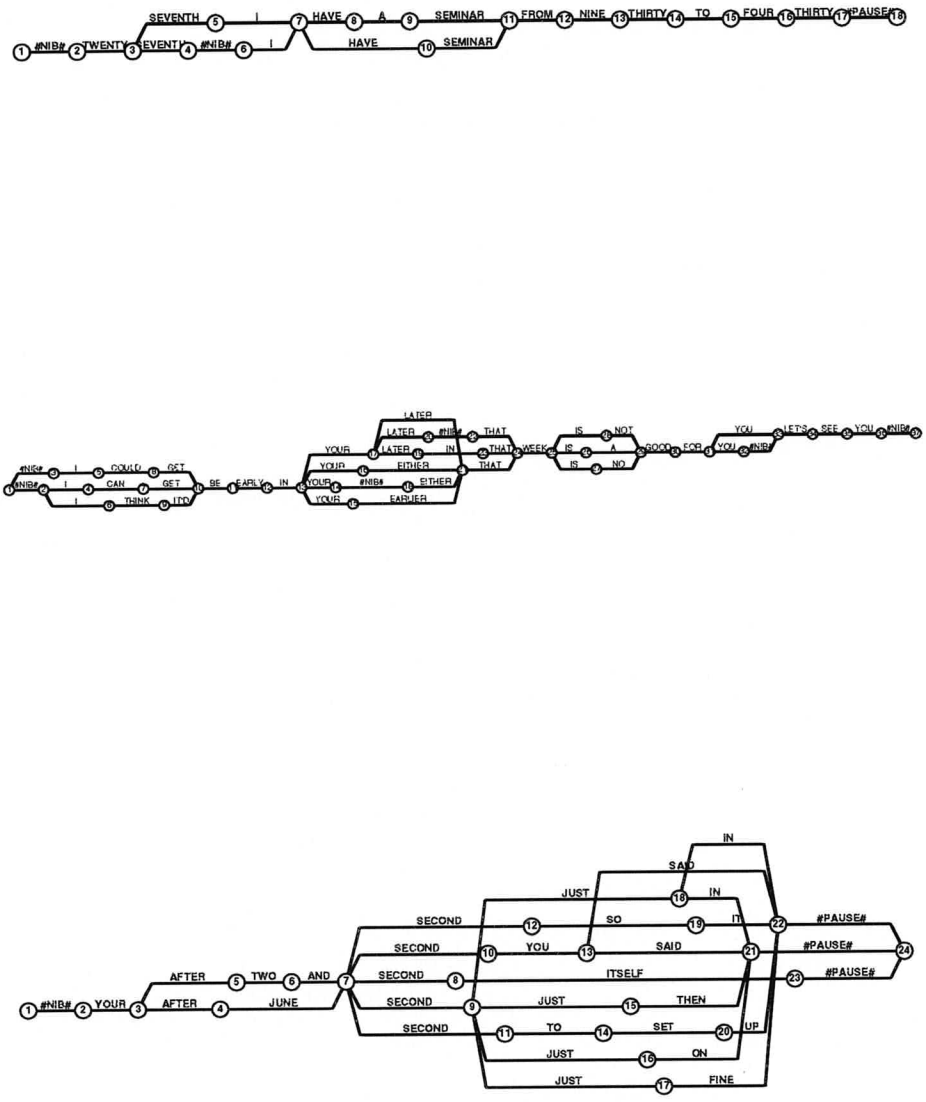


Figure B.4 Shrunk lattice

Bibliography

[1] Lavie Alon. An integrated heuristic for partial parse evaluation. In *ACL*, 1994.

[2] Lavie Alon and Tomita M. GLR* - an efficient noise-skiping parsing algorithm for context-free grammars. In *Third International Workshop on Parsing Technologies*, 1993.

[3] Black Ezra. Towards history-based grammars : Using richer models for probabilistic parsing. In *ACL 31*, 1993.

[4] Jelinek Fred. Up from trigrams. In *Eurospeech-91*, 1991.

[5] Alexandersson J., Maier E., and Reithinger N. A robust and efficient three-layered dialogue component for a speech-to-speech translation system, 1994.

[6] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: a maximum entropy approach. In *ICASSP*, April 1993.

[7] Lambert Lynn. *Recognizing Complex Discourse Acts:A Triplate Plan-Based Model of Dialogue*. PhD thesis, University of Delaware, 1993.

[8] Nagata Masaaki. An experimental statistical dialogue model to predict the speech act of the next utterance. Technical report, ATR, 1993.

[9] Tomita Masaru. An efficient word lattice parsing algorithm for continuous speech recognition. In *ICASSP 86, Tokyo*, 1986.

[10] Brown Peter. Maximum entropy methods and their applications to maximum likelihood parameter estimation of conditional exponential models. Technical report, IBM, 1990.

[11] Ronald Rosenfeld. *Adaptive Statistical Language Modeling: a Maximum Entropy Approach.* PhD thesis, Carnegie Mellon University, April 1993.

[12] Young Sheryl. Towards habitable systems: Use of world knowledge to dynamically constrain speech recognition. Technical report, CMU, 1988.

[13] Young Sheryl. Using dialog level knowledge sources to improve speech recognition. In *International conference on AI*, 1988.

[14] Young Sheryl. Using pragmatic and semantic knowledge to correct parsing of spoken language utterances. In *Eurospeech-91*, 1991.

[15] Young Sheryl. Using semantics to correct parser output for atis utterances. In *DARPA speech recognition workshop*, 1991.

[16] Morimoto T. Linguistic knowledge for spoken dialog processing. In *ICSLP-90*, 1990.

[17] Takezawa T. Atr hmm-lr continuous speech recognition system. In *ICASSP-90*, 1990.

[18] Monica Woszczyna, Alon Lavie, Tomas Polzin, Ivica Rogina, Bernard Suhm, and Alex waibel. Janus 93: Towards spontaneous speech translation. In *ICASSP*, 1994.

[19] Chow Y. and Roukos S. Speech understanding using a unification grammar. In *ICASSP 89*, 1989.