Discourse Processing in Enthusiast 3-Year Report

Yan Qu

May 23, 1997

Contents

1 Introduction **3**

2 The Discourse Model **4**
2.1 System Overview . . . 4
2.2 Discourse Coding Scheme . . . 4
2.3 Discourse-Related Components . . . 4
2.3.1 Grammar component . . . 4
2.3.2 Statistical ngrams . . . 5
2.3.3 Finite state machine . . . 5
2.3.4 Plan-based discourse processor . . . 5
2.4 Combination of different components . . . 5

3 Plan inference based discourse processing **6**
3.1 Speech Act Recognition . . . 7
3.2 Temporal Ellipsis and Anaphora Resolution . . . 8
3.2.1 Ellipsis resolution . . . 8
3.2.2 Anaphora Resolution . . . 9
3.3 Calendar update . . . 9
3.4 Parse disambiguation . . . 10
3.4.1 Non Context-Based Predictions . . . 10
3.4.2 Context-Based Predictions . . . 10
3.4.3 Combination of Context-based and non-context-based predictions . . . 12
3.5 Advantages and Disadvantages of a Plan-Inference Based Discourse Processor . . . 14
3.5.1 Complexity of Plan-Based Discourse Processor . . . 14
3.5.2 Scalability . . . 14
3.5.3 Cumulative error problem . . . 15

4 Finite State Machine Discourse Processor **16**
4.1 Speech act recognition . . . 16
4.2 Temporal ellipsis and anaphora resolution . . . 16
4.3 Calendar update . . . 16
4.4 Parse disambiguation . . . 16
4.4.1 Context-based predictions . . . 16
4.4.2 Combination of context-based and non context-based scores . . . 17
4.5 Advantages and Disadvantages of a FSM-Based Discourse Processor . . . 18

5 Cumulative Error in Discourse Processing **19**
5.1 FSM and the cumulative error . . . 19
5.1.1 Jumping context approach . . . 19
5.1.2 Hypothesis tree approach . . . 19
5.2 Plan-inference and cumulative error . . . 20

6 Evaluation **21**

7 Summary and New Directions **22**

1 Introduction

This document summarizes the major work on discourse processing in the Enthusiast Spanish-English speech translation project. Traditionally machine translation systems have processed sentences in isolation. Recently, there has been an interest in making use of discourse information in machine translation. In [15, 20], researchers at ATR advocate an illocutionary act based machine translation, arguing that equivalent sentence forms do not necessarily carry the same illocutionary forces across languages. In addition to speech act recognition, discourse information can be used for ellipsis and anaphora resolution to provide more accurate translation. Discourse processing has been shown to be useful in other natural language processing applications as well. For example, [45] reports that a discourse state can be used to select an appropriate language model for speech recognition and parsing to reduce the perplexity and improve recognition and parsing accuracy.

Our work on discourse processing has been done in four major directions:

- Designed discourse coding scheme for the appointment scheduling domain;
- Experimented with different methods for discourse processing: a plan inference based model and a finite state machine based model;
- Explored the use of discourse context to improve system performance:
  - Speech act recognition
  - Temporal anaphora and ellipsis resolution
  - Calendar update
  - Parse disambiguation
- Explored approaches to evaluating context and reducing cumulative error.

The major contributions of our work include:

- Implemented the first plan inference based discourse processor for large-scale spontaneous dialogues [37];
- Developed a computational model for multi-headed negotiations [38];
- Incorporated discourse knowledge for ambiguity resolution [26, 33, 39];
- Experimented with various methods for combining non-context- and context-based knowledge sources [26, 39];
- Explored techniques for reducing cumulative error in discourse context [32, 34].

In the next section, we give a brief system overview, which provides the context of the discourse work. We also describe the discourse coding schemes in this section. The two major discourse processing models – the plan inference based model and the finite state machine based model – are discussed in Section 3 and 4. The applications of the discourse context for speech act recognition, temporal ellipsis and anaphora resolution, calendar update and parse disambiguation are described together with their respective discourse processing models. We also summarize the advantages and disadvantages of each model. In section 5, we describe the techniques we experimented to reduce cumulative error. We evaluate various parts of our work in Section 6. We summarize our work in Section 7. Some of the work described in this report has been reported before in several conference papers and technical reports [37, 36, 38, 26, 33, 39, 32, 34]. The reader will be referred to these papers for more details.

2 The Discourse Model

2.1 System Overview

Enthusiast is a large Spanish-to-English speech-to-speech translation system in the appointment scheduling domain [42, 26, 25]. Enthusiast is composed of four main modules: speech recognition, parsing, discourse processing, and generation. Each module is domain-independent and language-independent but makes use of domain specific and language specific knowledge sources for customization.

YO PODRÍA MARTES EN LA MAÑANA
(I could meet on Tuesday in the morning)

```
((SENTENCE-TYPE *STATE)
 (FRAME *MEET)
 (SPEECH-ACT *SUGGEST)
 (A-SPEECH-ACT (*MULTIPLE* *SUGGEST *ACCEPT
                           *STATE-CONSTRAINT))
 (WHO ((FRAME *I)))
 (WHEN
   ((WH -) (FRAME *SIMPLE-TIME)
           (DAY-OF-WEEK TUESDAY)
           (TIME-OF-DAY MORNING)))
 (ATTITUDE *POSSIBLE))
```

Figure 1: **An Interlingua Text (ILT)**

The hypothesis produced by the speech recognizer about what the speaker has said is passed to the parser. The GLR* parser [24] produces a set of one or more meaning representation structures which are then processed by the discourse processor. The output of the parser is a representation of the meaning of the speaker's sentence. The meaning representation, called an interlingua text (ILT), is a frame-based language-independent meaning representation. The main component of an ILT are the speech act (e.g., `suggest`, `accept`, `reject`), the sentence type (e.g., `state`, `query-if`, `fragment`), and the main semantic frame (e.g., `free`, `meet`). An example of an ILT is shown in Figure 1.

The output of the parser is then sent to the discourse processor. The discourse processing module disambiguates the speech act of each utterance, updates a dynamic memory of the speakers' schedules, and incorporates the utterance into discourse context.

2.2 Discourse Coding Scheme

In Enthusiast, we identify the speech acts of dialogue utterances. Speech acts represent the communicative functions of utterances. Development of our discourse processing module was based on a corpus of 20 spontaneous Spanish scheduling dialogues containing a total of 630 utterances. We identify a total of fourteen possible speech acts in the appointment scheduling domain (Figure 2). The details about these speech acts are described in [37].

2.3 Discourse-Related Components

Four processing components in the system concerns with discourse processing: a grammar component, a statistical component, a finite state machine, and a plan-based discourse processor. We describe briefly the four components below. The plan-based discourse processor and the finite state machine based discourse processor are to be discussed in more detail in Section 3 and 4.

2.3.1 Grammar component

The grammar component assigns a set of possible speech acts to an ILT based on the syntactic and semantic information in the interlingua representation. The resulting set of possible speech acts is inserted into the `a-speech-act` slot of the ILT (See Figure 1).

Speech Act	Example
Accept	Thursday I'm free the whole day.
Acknowledge	OK, I see.
Address	Wait, Alex.
Affirm	Yes.
Closing	See you then.
Confirm	You are busy Sunday, right?
Confirm-Appointment	So Wednesday at 3:00 then?
Deliberation	Hm, Friday in the morning.
Negate	No.
Opening	Hi, Cindy.
Reject	Tuesday I have a class.
Request-Clarification	What did you say about Wednesday?
Request-Response	What do you think?
Request-Suggestion	What looks good for you?
State-Constraint	This week looks pretty busy for me.
Suggest	Are you free on the morning of the eighth?

Figure 2: **Speech Acts Covered by Enthusiast**

2.3.2 Statistical ngrams

The statistical component predicts the following speech act using knowledge about speech act frequencies acquired from our training corpus. The statistical component is able to provide ranked predictions in a fast and efficient way. To cater to the sparse data problem, bigram speech act probabilities are smoothed based on backoff models [18].

2.3.3 Finite state machine

The finite state machine (FSM) describes representative sequences of speech acts in the scheduling domain. It is used to record the standard dialogue flow and to check whether the predicted speech act follows idealized dialogue act sequences. The details of the FSM will be discussed in Section 4.

2.3.4 Plan-based discourse processor

The plan-based discourse processor handles knowledge-intensive tasks exploiting various knowledge sources, including the grammar component predictions and linguistic information. The details of the plan-based discourse processor are discussed in Section 3.

2.4 Combination of different components

The finite state machine and the statistical component have been implemented as a fast and efficient alternative to the more time-consuming plan-based discourse processor. Currently the two discourse processors operate independently. A possible way of combining them to take the advantages of both approaches is to adopt a layered architecture similar to the one proposed in Verbmobil [35, 1]. In such an architecture, the finite state machine would constitute a lower layer providing an efficient way of recognizing speech acts, while the plan-based discourse processor, at a higher layer, would be used to handle more knowledge intensive processes, such as recognizing doubt or clarification sub-dialogues and robust ellipsis resolution.

3 Plan inference based discourse processing

Our plan-based discourse processor follows previous work of using a plan inference model for understanding discourse utterances [2, 27, 7, 22]. Plan-based approaches are based on Austin's [3] and Searl's [40] notion of language as a series of actions, and Grice's [13] and Searle's description of language as goal-oriented behavior. Specifically, we developed our discourse processor following Lambert's work [22, 23] on understanding negotiation dialogues. Lambert's work extends previous work on discourse processing in several ways: (1) separating three levels of processing: the domain level, the problem solving level and the discourse level; (2) recognizing actions expressed over multiple sentences; and (3) being fully implemented to process negotiation dialogues. We have left out aspects of her model which are too knowledge intensive to gain the coverage we need for understanding spontaneous speech.

Plan recognition systems maintain a library of "generic" subplans which describe actions that when performed under certain conditions (i.e. if certain beliefs and preconditions hold), bring about certain effects or goals. These subplans are called plan operators. Discourse plan operators represent knowledge about how to achieve discourse goals. Our plan operators are similar to those of [2, 27, 7, 22]. Each plan operator consists of a header, a type, a (possibly empty) set of preconditions, a (possibly empty) set of constraints on the variables, a (possibly empty) set of applicability conditions, a (possibly empty) set of actions that form a body, a set of effects, and a goal. Figure 3 shows an example discourse plan operator defined in Enthusiast.

```
((Header 0 (Reject (_speaker1 VAR PARTICIPANT)
                   (_speaker2 VAR PARTICIPANT)
                   (_action VAR PROPOSITION)
                   (_rejection VAR PROPOSITION)))
 (Type Decomposition)
 (Applicability.Conditions )
 (Constraints
    (NIL 0 NIL rejection-to-prop (_rejection VAR PROPOSITION)
                                 (_action VAR PROPOSITION)))
 (Preconditions )
 (Specializes )
 (Decomposition
    (NIL + NIL Reject-Form (_speaker1 VAR PARTICIPANT)
                           (_speaker2 VAR PARTICIPANT)
                           (_rejection VAR PROPOSITION)))
 (Specializations )
 (Primary.Effects
   (NIL 0 NIL believe (_speaker2 VAR PARTICIPANT)
                      (relates PROP PROPOSITION
                               (_speaker1 VAR PARTICIPANT)
                               (_action VAR PROPOSITION)
                               (_rejection VAR PROPOSITION))
                       ((3 3) belief BELIEF.INTERVAL)))
 (Goals
   (NIL 0 NIL believe (_speaker2 VAR PARTICIPANT)
                      (relates PROP PROPOSITION
                               (_speaker1 VAR PARTICIPANT)
                               (_action VAR PROPOSITION)
                               (_rejection VAR PROPOSITION))
                        ((3 3) belief BELIEF.INTERVAL)))
 (Level.type Discourse))
```

Figure 3: **An Example Plan Operator**

As the system recognizes discourse actions, it builds a structure of the discourse. This structure is referred to as the dialogue model. It is represented as a tree representing the system's beliefs about the user's plan. The dialogue model is constructed from the semantic representations of the utterances in discourse, using the plan operators and a plan inference mechanism to infer the actions and plans performed by the dialogue participants. The right frontier of the plan tree represents the salient actions in discourse context.

We describe how the plan-inference model is used for speech act recognition in Section 3.1, temporal ellipsis and anaphora resolution in Section 3.2, and calendar update in Section 3.3.

3.1 Speech Act Recognition

Speech act selection is strongly related to the task of determining how the current sentence relates to the discourse plan tree. Out of context, an utterance has the potential of performing a number of different communicative functions. For example, *I'm free Tuesday* could be either a `Suggest` or an `Accept`. *Tuesday I have a class* could be a `State-Constraint` or a `Reject`. It is important to take context into consideration in deciding which speech act was actually performed in a given instance.

S1: 1. When can you meet next week?
S2: 2. Tuesday afternoon looks good.
3. I could do it Wednesday morning too.
S1: 4. Tuesday I have a class from 12:00-1:30.
5. But the other day sounds good.

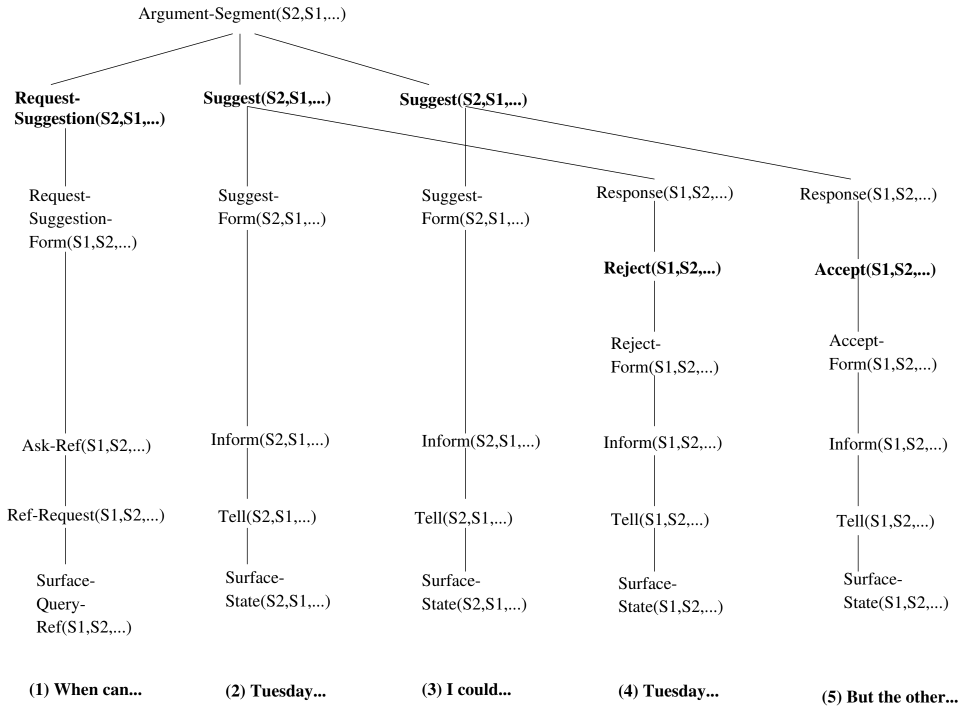


Figure 4: **Sample Discourse Structure**

When the discourse processor computes a chain of inference for the current input sentence, it tries to attach the inference line to the current plan tree. Where the inference line attaches determines which speech act is assigned to the input sentence. For example, in Figure 4, both sentences 4 and 5 attach as responses to the previous suggestions. Since sentence 4 chains up to an instantiation of the `Response` operator from an instantiation of the `Reject` operator, it is assigned the speech act `Reject`. Similarly, since sentence 5 chains up to an instantiation of the `Response` operator from an instantiation of the `Accept` operator, sentence 5 is assigned the speech act `Accept`. Typically more than one inference chain gets built for a single ILT, representing different possible speech acts which could be performed by the utterance. A focusing heuristic is used to rank the different inference chains and choose the one which attaches most coherently to the discourse plan tree [22].

A novel feature of our representation of discourse structure is that we allow multiple foci in parallel, which challenges the commonly held assumption that discourse has a recursive, tree-like structure. For example, in Figure 4, after sentence 3, not only is the second, rightmost suggestion in focus, along with its corresponding inference chain, but both suggestions are in focus, with the rightmost one being slightly more accessible than the previous one. So when the first response is processed, it can attach to the first suggestion. And when the second response is processed, it can be attached to the second suggestion. Both suggestions remain in focus as long as the node which immediately dominates the parallel suggestions is on the rightmost frontier of the plan tree. Our model of the discourse structure result in modified version of focusing heuristics compared with Lambert's version to take care of the multi-headed foci. Technical aspects of our focusing heuristic are described in more detail in [37, 38].

Once the discourse processor attaches the current utterance to the plan tree thereby selecting the correct speech act in context, it inserts the correct speech act in the `speech-act` slot in the interlingua structure. At the same time, the meaning representation for the current utterance is augmented from context. In the scheduling domain, our focus of meaning augumentation is on temporal ellipsis and anaphora.

3.2 Temporal Ellipsis and Anaphora Resolution

We developed a Temporal Expert Program to reason about relationships between time expressions and for filling explicit temporal information from context. The temporal expert program handles a wide range of temporal expressions including `<simple-time>, <time-list>, <interval>, <special-time>`, and temporal specifiers like *next*. The ILT specification for these types of expressions can be found by reference to the grammar specification.

The temporal expert program does not evaluate relationships between expressions in their interlingua form. It first maps this structure onto an internal representation called a *tstruct* (Figure 5). The tstruct structure was patterned after the interlingua specification for `<simple-time>`, so that these structures could easily be mapped onto the structure used for temporal reasoning. Representation for `<time-list>` can be constructed by creating separate tstructs for each temporal expression in the list and merging them into the fewest number of tstructs possible to cover the same range of time. `<interval>` can be constructed by first constructing tstructs for the endpoints and then computing a tstruct which bridges the gap between them. The one which bridges the gap is returned. `<special-time>`, temporal specifiers, temporal ellipsis and anaphora are handled by reference to context.

```
(tstruct :minute
         :hour
         :day
         :day-of-week
         :month
         :year)
```

Figure 5: **The *tstruct* data structure**

3.2.1 Ellipsis resolution

Often times temporal expressions are not fully specified. To understand the utterance and update the calendar, we need to resolve temporal ellipsis from context. A time with complete temporal information specifies the `year, month, day, day-of-week, hour`, and `minute` of that time as specified in the `tstruct` data structure. Any temporal expression which does not specify all of the above information needs to be augmented with the relevant information from context. For example, in the following dialog,

```
A: Is Tuesday good for you?
B: Yes.   How about 2:00 o'clock?
```

Both the temporal expressions *Tuesday* and *2 o'clock* need to be augmented with information of date for the program to keep track of which date the speakers are talking about. We introduce two temporal threads into the temporal expert program. One temporal thread is a static pointer `*today*`, which keeps track of the date when the speakers were supposed to conduct the conversation. This pointer is usually useful when scheduling is brought back to the initial reference time by linguistics clues such as specifiers like `this` (e.g. *this Friday, this coming week*). Another temporal thread is dynamic, updated by the utterances as the conversation goes on. This temporal thread includes several pointers to keep track the most recent temporal references. These pointers include `*ref-day*`, `*ref-week*`, `*ref-month*`, and `*ref-year*`. Note though `*ref-month*` and `*ref-year*` are important for filling out complete temporal information, they are usually quite constant unless scheduling spans over more than one month or one year. `*ref-day*` and `*ref-week*` are the most important for keep the temporal references as conversation goes on. An implicit assumption in our approach of resolving temporal ellipsis by keeping reference times is that in multi-agent dialogs, both agents observes the conversational cooperation principle, so that a reference time established by one agent is licit for both the agent and the other agent.

3.2.2 Anaphora Resolution

We classify the following types of anaphora based on the corpus analysis of our training dialogues:

- special time `then` refers to the latest temporal expression;
- frame `*SAME` refers to previous `frame` and `when` slots;
- special time or temporal specifier `THAT/IT` refer to previous `when` or `topic` slots;
- temporal specifier `those` (e.g. *on THOSE days, on the OTHER days*) refers to a list of previously mentioned times.

These types are used mostly as *inter-sentential* anaphora, whose reference is in sentences prior to the sentence with the anaphora; there are also some occurrences of *intra-sentential* anaphora, whose reference is within the sentence with the anaphora (e.g. *it will have to be around Tuesday or from THAT onward*). Our focus was on anaphora resolution of inter-sentential temporal anaphora.

We experimented with two approaches for resolving inter-sentential anaphora. The first approach was a *surface approach*, which only considers the ILT representations of sentences. The other is a *plan-based approach*, which employs contextual information in the discourse plan tree. The advantage of the surface approach is its efficiency in implementation and execution. The disadvantage of this approach is that the anaphora is resolved only to the most immediate context. The advantage of the plan-based approach is that in a correct plan tree, the constraining relation between utterances are recorded and that the global discourse structure is kept, thus it can take advantages of the hierarchical structure of discourse and resolve anaphora in a non-linear way. In this report we only report the results produced using the surface approach. The results of the surface approach provide hints on cases for which the surface approach works, and cases for which it fails to work. In cases where the approach does not work, ideally the plan-based approach would improve the performance by bringing into consideration more detailed contextual knowledge. However, because of cumulative error in context, we have not been able to see improvement in performance.

3.3 Calendar update

We keep a dynamic memory, a calendar, to keep a record of the set of temporal constraints for each speaker as expressed by the conversation. The data structure of the calendar is a linked list of week structures. Each week structure has a substructure for each day. Each day is a structure with 96 slots, 48 half hours slots for each conversational participant. Each of the slots is a number which indicates the status in term of how open this time slot is for scheduling a meeting. Every slot is originally initialized to 0 which has a neutral status. Possible status values are `4(accepted), 3(proposed), 2(good), 1(considering), 0(neutral), -1(not prefered), -2(busy), -3(bad)`, and `-4(rejected)`. The data structure for representing calendar is shown in Figure 7. Figure 7 gives the graphical representation of an example calendar entry for the following dialogue:

```
S1: How about the sixteenth from noon to 1 o'clock?
S2: From noon to one is good for me.
```

To insert a constraint into the calendar, a constraint structure is first constructed from the sentence. The temporal expression from the sentence is first mapped into a tstruct which is then inserted into the constraints times slot. The speech act is then mapped into a status which is then inserted into the status slot. In spontaneous dialogues, the speakers sometimes contradict themselves by changing their mind. Currently we assume that later constraints are always correct updates of former constraints for a particular time slot. Therefore, when a later constraint overlaps with a former constraint, the overlapping slots are updated with the status of the later constraint. Ideally the information of conflicting constraints can be used to detect system errors, and signal the need for repairs.

3.4 Parse disambiguation

Ambiguity is a major problem in a large scale machine translation system such as Enthusiast. This is because the parsing grammar must be large in order to cover the wide range of constructions in spontaneous speech. Additionally, the flexibility of the GLR* skipping parser [24] magnifies the problem.

We gauge ambiguities in terms of differences between members of the set of ILTs produced by the parser for the same source sentence. We noticed in our experiment the disambiguation task benefits from both non context- and context-based methods. We observed that some classes of ambiguities can be more perspicuously dealt with in one way or the other.

3.4.1 Non Context-Based Predictions

When the parser produces more than one ILT for a single sentence, it scores these ambiguities according to three different non context-based disambiguation methods. The first method, based on [8], assigns probabilities to actions in the GLR* parser's parse table. The probabilities of the parse actions induce statistical scores on alternative parse trees, which are then used for parse disambiguation. The resulting score is called the *statistical score*. The second method the parser uses to score the ILTs makes use of penalties manually assigned to different rules in the parsing grammar. The resulting score from this method is called the *grammar preference score*. The third score, called the *parser score*, is a heuristic combination of the previous two scores plus other information such as the number of words skipped. These three non context-based scores will be referred to later when we discuss combining non context-based predictions with context-based ones.

3.4.2 Context-Based Predictions

While a broad range of ambiguities can be handled well in a non context-based manner, some ambiguities (e.g. temporal ambiguities) must be treated in a context sensitive manner in order to be translated correctly.

Context-based ranking of ambiguities is performed by the plan-based discourse processor. In Section 3.1 we described how the discourse processor processes the *single* best parse returned by the parser. For a plan-based discourse processor to deal with multiple inputs, three major issues need to be addressed:

1. The discourse processor must be able to deal with more than one semantic representation as input at a time. Note that simply extending the discourse processor to accept multiple ILTs is not the whole solution to the disambiguation problem: finer distinctions must be made in terms of coherence with the context in order to produce predictions detailed enough to distinguish between alternative ILTs.

2. Before context-based predictions can be combined with quantitative non context-based predictions, they must be quantified. It was necessary to add a mechanism to produce more detailed quantifiable predictions than those produced by the original focusing heuristics described in Section 3.1.

3. Finally, context-based predictions must be combined successfully with non-context-based ones. The discourse processor must be able to weigh these various predictions in order to determine which ones to believe in specific circumstances.

Thus, we extended our original discourse processor as follows. It takes *multiple* ambiguous ILTs from the parser and computes three quantified discourse scores for each ambiguity. The discourse scores are derived by taking into account *attachment preferences* to the discourse tree, as reflected by two kinds of *focusing scores*, and the score returned by the *graded constraints*, a new type of constraint we introduced. Then for each ambiguity the discourse processor combines these three kinds of context-based scores with the non context-based scores produced by other modules of the system to make the final choice, and returns the chosen ILT. As in the first version of the discourse processor, the chosen ILT is attached to the plan tree and a speech act is assigned to it.

Focusing scores

The focusing scores are derived from focusing heuristics based on [22, 38]. The focusing heuristics identify the most coherent relationship between a new inference chain and the discourse plan tree. Attachment preferences by the focusing heuristics are translated into numerical preference scores based on attachment positions and the length of the inference chains. The assignment of focusing scores reflects the assumption that the most coherent move in a dialogue is to continue the most salient focused actions, namely, the ones on the rightmost frontier of the plan tree. We introduced two kinds of focusing scores: the focusing flag and the focusing score proper.

- The first focusing score is a boolean *focusing flag*. It returns 0 if the inference chain for the associated ILT attaches to the rightmost frontier of the plan tree, 1 if it either attaches to the tree but not to the right frontier or doesn't attach to the tree.

$$Focusingscore = \begin{cases} 1 & , \quad iffirstfocusingheuristic; \\ 0 & , \quad otherwise. \end{cases} \tag{1}$$

- The second focusing score, the *focusing score* proper, assigns a score between 0 and 1 indicating how far up the rightmost frontier the inference chain attaches. The maximal score is assigned in the case that the inference chain does not attach.

$$Focusingscore = \frac{attachmentdistancefromthefocuspoint}{lengthofthebranch}$$

Graded constraints

Once the discourse processor was extended to accept multiple ILTs as input, it became clear that for most ambiguous parses the original focusing heuristics did not provide enough information to distinguish among the alternatives. Our solution was to modify the discourse processor's constraint processing mechanism, making it possible to bring more domain knowledge to bear on the disambiguation task. In the original discourse processor, all of the constraints on plan operators, which we call *elimination constraints*, were used solely for the purpose of binding variables and eliminating certain inference possibilities. Their purpose was to eliminate provably wrong inferences, and in this way to give the focusing heuristics a higher likelihood of selecting the correct inference chain from the remaining set.

We introduced a different type of constraint, *graded constraints* (Figure 8), inspired by the concept of graded unification discussed in [19]. Unlike elimination constraints, they neither bind variables nor eliminate any inferences. Graded constraints always return true, so they cannot eliminate inferences. However, they assign numerical penalties or preferences to inference chains based on domain specific information. This information is then used to rank the set of possible inferences left after the elimination constraints are processed.

$$Sum - of - graded - constraints = \frac{sumofpenalties}{numberofconstraintsfired}$$

Figure 8 shows an example plan operator with graded constraint `prefered-time`. `Prefered-time` assigns a penalty score when the temporal information in the sentence and the status of the times is in conflict with constraints recorded in the calendar or the dialogue date.

By introducing graded constraints, we don't need to introduce more fine-tuned plan operators to distinguish some ambiguities (e.g. introducing `suggest-day` and `suggest-hour` for `suggest`), thus avoiding expanding the search space among the plan operators. Moreover, graded constraints only give preferences, they do not rule out inferencing and attachment possibilities. Thus introducing new constraints will not damage the broad coverage of the system.

The preferences in the preference constraints are basically encoded by linguistic knowledge and world knowledge of the scheduling domain. Some major constraints for disambiguation include day/hour ambiguity, when/where ambiguity, sentence-type ambiguity (statement/query-if/fixed-expression), frame ambiguity (how/greet, closing/interject, etc).

3.4.3 Combination of Context-based and non-context-based predictions

Once the information from the graded constraints and the focusing scores is available, the challenging problem of combining these context-based predictions with the non context-based ones arises. We experimented with two methods of automatically learning functions for combining our six scores into one composite score, namely a genetic programming approach and a neural net approach. The basic assumption of our disambiguation approach is that the context-based and non context-based scores provide different perspectives on the disambiguation task. They act together, each specializing in different types of cases, to constrain the final result. Thus, we want our learning approach to learn not only which factors are important, but also to what extent they are important, and under what circumstances. The genetic programming and neural net approaches are ideal in this respect.

The genetic programming approach

The genetic algorithm and genetic programming techniques have been applied to a few problems in computational linguistics [28, 4, 10, 41]. The advantage of the genetic programming approach is that it learns functions which are humanly understandable and editable. Also, it learns small functions which can be evaluated quickly. Since the learned function which combines the six scores must be called for each inference chain which the discourse processor is selecting from, speed is an important consideration. The genetic programming approach can generate functions in any desired computer language, so not only are the functions learned quick to evaluate, they are also convenient to evaluate. Finally, because they sample different parts of the search space in parallel, they avoid to some extent the problem of selecting locally optimal solutions which are not globally optimal.

Genetic programming is a method for "evolving" a program to accomplish a particular task. Genetic programming is an extension to the traditional genetic algorithm originally introduced in [14] and developed further in [29]. The genetic algorithm is a machine learning approach inspired by the Darwinian model of evolution [12]. Character strings which represent chromosomes, or more abstractly features of an organism's phenotype, encode data which is used by a program. The "fitness" of a character string is computed by measuring how well the program performs given that character string as data. A population of chromosomes is evaluated. A set of chromosomes which are relatively more fit than the rest of the population are chosen to "mate" with each other and produce a new set of chromosomes for the next generation. When chromosomes "mate", an operation called *crossover* is used. Some point along one of the chromosomes is chosen as a crossover point. Two new chromosomes are produced, one with the first part, that is the part before the crossover point, of the first chromosome matched with the second part of the second chromosome and one with the first part of the second chromosome matched with the second part of the first chromosome. The idea then is that what is good about one chromosome will hopefully be combined with what is good about the second chromosome. In this way, partial solutions to a problem can be built up over time. In addition to *crossover*, another operation which affects the next generation is *mutation* where a single character in a small subset of chromosomes is randomly changed. This is to partially compensate for what "genetic diversity" is lost as the population becomes more and more focused on those chromosomes which seem to be the most fit, until a chromosome is evolved which reaches some threshold of fitness or some maximum number of generations are exceeded.

For the genetic programming approach, we first preprocess the set of scores for each ambiguity. There are six scores for each clause. We average these scores getting six averaged scores plus one additional score which indicates how many clauses there were[1]. Because we do this preprocessing before we train, we must embed the trained function in a hand coded function which takes the sets of scores for each ambiguity and turns them into one set of seven scores for use inside of the discourse processor.

Before training can take place, a terminal set must be designated. The best combination which we found was to provide seven variables, one corresponding to each of the seven composite scores, and a random floating-point operator which is instantiated to a number between 12.0 and -3.0 when it is included in a program.

Next, a function set, preferably one with closure, must be selected. The function set which worked best overall was one containing $+$, $*$, $\%$, and $-$. $\%$ is protected division as described in [21].

Then a function for evaluating the fitness of each program must be written. The one which we use returns $1 - A$ where A is the percentage of test cases where the program which is being evaluated selected the correct ambiguity. Lower fitness values are preferred.

[1] This is useful since ambiguities with fewer clauses tend to be correct more often

The results presented below for the genetic programming approach were obtained by using a function created on the third generation of a run of the genetic programming algorithm set up with the specifications above with population size 3000 and using fitness proportionate reproduction. Fitness proportionate reproduction is a method in which individuals are selected for reproduction in proportion to their fitness value so that programs which perform better on the current generation will have a bigger influence on the development of the next generation.

The neural net approach

Connectionist approaches have been widely used for spoken language processing and other areas of computational linguistics because of their robustness to noise and easy integration of different information sources [44, 31, 17]. We chose to pursue a connectionist approach because of its robustness to noise and demonstrated successfulness in past applications. However, we have not seen any work so far that integrates complex discourse information with other information sources using a connectionist approach.

Various connectionist models have been proposed because of the wide interest in neural networks and connectionist architectures. No matter how much the proposals differ in detail, a typical network model consists of an input layer with a set of input units, an output layer with a set of output units, and typically some hidden layers between the input and output layers. Each unit is represented mathematically using real numbers. Each connection has an associated weight which determines the effect of an incoming input on the unit. The weights can be positive or negative.

The error back-propagation algorithm [11] is a powerful connectionist learning algorithm. The algorithm repeatedly adjusts the weights in a feed-forward network[2] of nonlinear perceptron-like units[3] to minimize the measure of the difference between the actual output vector of the network and a desired output vector given a particular input vector. The weight adjusting rule is derived by propagating partial derivatives of the error backwards through the network simultaneously refining the weights of all network units. More precisely, it attempts to minimize the squared error between the network outputs and the target values for the outputs provided by the training data. The degree to which weights are changed at each step is set by the learning rate. Starting from random weights, back-propagation networks can learn to user their hidden units to represent structures that are inherent in the input data.

Our network architecture is a simple three layer neural net trained by the back-propagation learning algorithm.

Input Representation: The neural net takes as input five sets of six scores, each corresponding to up to five possible ILTs for each processed sentence[4]. The six scores are the grammar preference value, the statistical score, the parse score, the focusing score, the attachment score, and the graded constraint scores. This amounts to a total number of 30 input units. If a processed sentence has fewer than five possible ILTs, we fill the empty slots by setting the units to zero.

Output Representation: The output layer consists of five output units, each representing a score corresponding to an input ILT. A final decision for disambiguating the input ILTs by the network is based on the ranking among the output units. The unit with the highest score is considered the output by the network. We view disambiguation by the network as a ranking problem rather than a classification problem. The final decision by the network is not decided on whether a particular output unit surpasses a threshold, but rather on whether the target ILT gets the highest ranking in the output units.

The evaluation reported in the later part of this paper is based on a network with the parameters set at the following values:

```
number of input units: 30
number of output units: 5
number of hidden units: 4
```

---

[2] A feed-forward network is one in which activation is propagated forward through the input layer, then through any hidden layers, and finally to the output layer.

[3] These units perform some function on the sum over the input activations in order to compute the output activation, in this case a non-linear function.

[4] The number five here is arbitrary. In practice, we have found that if the set of ILTs returned by the parser contains a correct one, it is usually within the top five.

```
learning rate: 0.01
```

Currently architectural parameters of the network such as the number of hidden units and the learning rate have to be manually adapted to the training corpus and domain. To avoid overfitting[5], a separate set of five dialogues is used for cross-validation[6]. It would be preferable in the future to implement an automatic architecture optimization algorithm such as the Automatic Structure Optimization [5] and Automatic Validation Analyzing System [6] to automatically tune network parameters and detect overfitting.

The confidence combination

Our experiments with the genetic programming and neural net approaches for combining the context-based and non-context-based predictions show that getting a good combination global function can be a hard problem. Thus we experiment with a new combination method, which takes into account the confidence of the discourse processor towards its own predictions. The method involves first to single out cases in which the discourse processor is confident with its decisions, and then make use of context scores only for these cases. For other cases, the discourse processor conducts disambiguation based on the parser scores, and neglects its own predictions. With this combination method, we hope to guarantee that the discourse processor will at least not do worse than the parser.

3.5 Advantages and Disadvantages of a Plan-Inference Based Discourse Processor

The plan-based discourse processor we developed for Enthusiast has both advantages and disadvantages. Among the advantages, the plan-based processor can incorporate into the plan operators the contextual constraints and use the inference mechanism to filter out conflicting constraints. Moreover, plan-based discourse processor represents the context in a hierarchical tree format, which is capable to capture meta-utterance actions, such as augmentation acts [43, 30]. However, we noticed several drawbacks of using a plan inference system of processing context and of incorporating the contextual information with other components of the whole system. Some of these drawbacks are specific to the plan-inference model, while others are drawbacks for context-based approaches in general. In this section, I discuss three drawbacks as observed in the system: the complexity of plan inferencing, lack of scalability and the cumulative error problem.

3.5.1 Complexity of Plan-Based Discourse Processor

In a plan inference system, checking the preconditions or constraints and producing relevant goals and effects requires unification. Because unification is an expensive process, the plan-inference based discourse processor takes more time to process context than a real-time speech translation system allows. In order to take advantage of contextual information, we need to develop an fast and efficient way of presenting context that can be incorporated into the working system. This motivates us to develop a finite state machine based discourse processor in Section 4.

3.5.2 Scalability

There are two major scalability problems when we want to extend the current discourse processor into a new domain. First, since the plan operators are arranged in a hierarchical relation, adding new plan operators will generally require re-definition of at least the parent plan operators. Second, since graded constraints are domain dependent, we are required to introduce new domain-dependent plan operators simply because new domain-dependent graded constraints are required. One solution to the scalability problem is to separate the graded constraint processing from the plan inference model. In this way, introducing new graded constraints would not require modifying the plan operators.

[5] Overfitting refers to the situation in which the weights are being tuned to fit the idiosyncrasies of the training examples that are not representative of the general distribution of examples. Overfitting decreases the generalization accuracy of the network over unseen examples.

[6] The cross-validation approach is perhaps the most successful method for overcoming the overfitting problem. The method uses a set of test data in addition to the training data. The back-propagation algorithm can then monitor the error with respect to the test set, while using the training set to adjust the network weights. Training is terminated when the error is the lowest on the test set, as this is the best indicator of network performance over unseen examples.

3.5.3 Cumulative error problem

So far in our discussion, we have assumed we always get the right context for future processing. However, since we need focusing heuristics for the most relevant context and heuristics can make wrong predictions, the speech act may be misrecognized, thus making the context inaccurate for future context based predictions. Unexpected input, disfluencies, out of domain utterances, and missing information add to the frequency of misrecognition in spontaneous speech systems. Misrecognition of context states resulting from these features of spontaneous speech adversely affect the quality of contextual information for processing later information. For example, unexpected input can drastically change the standard flow of speech act sequences in a dialogue. Missing contextual information can make later utterances appear to not fit into the context.

Cumulative error is introduced when an incorrect hypothesis is chosen and incorporated into the context, thus providing an inaccurate context from which subsequent context based predictions are made. Cumulative error can be a major problem in natural language systems using contextual information. For example, in our evaluation of the genetic programming approach and the neural net approach for parse disambiguation, we observed that in absence of cumulative error, context based predictions contributed to the task of parse disambiguation (Table 1). This results in an improvement of 13% with the genetic programming approach and of 2.5% with the neural net approach compared with the parser's non-context based statistical disambiguation technique. However, cumulative error affected the contribution of contextual information. In the face of cumulative error, the performance decreased by 7.5% for the neural net approach and by 29.5% for the genetic programming approach compared to their respective performances in the absence of cumulative error, thus dragging the performance statistics of the context based approaches below that of the parser's non-context based statistical disambiguation technique.

	Training	**Testing without CE**	Testing with CE
Random	32%	45%	45%
Grammar Preferences	75.6%	81.3%	81.3%
Statistical Parse Disambiguation	76.5%	76.3%	76.3%
DP Genetic Programing	91.6%	89.5%	60%
DP Neural Net	85.2%	78.8%	71.3%
Number of Sentences	115	76	76

Table 1: **Disambiguation of All Ambiguous Sentences**

Cumulative error is not a problem inherent for the plan inference model. It is a general problem for any model which utilizes contextual information for future processing. Church and Gale [9] state that "it is important to estimate the context carefully; we have found that poor measures of context are worse than none." However, we are not aware of this issue having been raised in the discourse processing literature. We discuss the some techniques we experimented with to reduce the cumulative error in discourse context in Section 5.

4 Finite State Machine Discourse Processor

The finite state machine (FSM) describes representative sequences of speech acts in the scheduling domain. The speech acts are the same set of speech acts as described in Section 2.2. It is used to record the standard dialogue flow and to check whether the predicted speech act follows idealized dialogue act sequences. The FSM consists of states and transition arcs. The states represent speech acts in the corpus. The transitions between states can have the symbols: `S` (for the same speaker), `C` (for change of speaker), or null (no symbol); the latter represents the cases in which the transition is legal, independent of whether the speaker changes or remains the same. A graphical representation of the major parts of the FSM appears in Figure 9. We extended the FSM so that at each state of the finite state machine we allow for phenomena that might appear anywhere in a dialogue, such as `acknowledge`, `address`, `confirm`, `request-clarification`, and `deliberation`. A segment of the lisp-readable representation of the network is given in Figure 10.

4.1 Speech act recognition

For the task of dialogue act recognition, the finite state model itself is not sufficient. First, the number of predictions from such a model is not sufficiently restrictive, and second, the model does not represent preference information for the various dialogue acts given context. To represent the preferences, we developed a statistical component to predict the following speech act using knowledge about speech act frequencies in our training corpus. The statistical component is able to provide ranked predictions in a fast and efficient way.

Currently, we use n-gram speech act probabilities to compute the most likely follow-up dialogue act. The method is adapted from speech recognition, where language models are commonly used to reduce the search space when determining a word that can match a part of the input signal [16]. It was used for the task of dialogue act predictions by [35]. To cater to the sparse data problem, bigram and trigram speech act probabilities are smoothed based on backoff models [18].

A way to reduce the set of predictions from the FSM is to combine this set with the set of possible speech acts (`a-speech-act`) proposed by the grammar component for the same input utterance. The intersection of the finite state machine predictions and the grammar component predictions should yield the speech acts which are consistent both with the input semantic representation and with the standard dialogue flow. Oftentimes after the intersection, an utterance can still perform more than one legal function. Bigram speech act probabilities are then used to select the most probable one from the intersection set. We call this way of speech act assignment the *strict context approach*.

4.2 Temporal ellipsis and anaphora resolution

The temporal ellipsis and anaphora are resolved in the same way as described in Section 3.2.

4.3 Calendar update

The calendar memory is dynamically updated in the same way as described in Section 3.3. The only difference is that the discourse states are now provided by the FSM-based discourse processor instead of the plan-inference-based model.

4.4 Parse disambiguation

The philosophy for parse disambiguation is the same as described in Section 3.4. The non context-based scores used are the same as before: the statistical scores, the grammar preference scores, and the parser scores. Since in a finite state model, there is no concept like tree attachment, we redefined our context scores for the FSM discourse processor.

4.4.1 Context-based predictions

The FSM discourse processor is extended for disambiguation to produce two independent discourse scores: the speech act probabilities and the ILT constraints. The bigram speech act probabilities capture the preferences of follow up speech acts for corresponding ILTs. The ILT constraints capture the context-based preferences or penalties for the ILTs.

Bigram speech act probabilities

The bigram speech act probabilities are recorded to capture the likelihood of a projected speech act with respect to the given context.

ILT constraints

ILT constraints differ from the grammar preference scores (counter values) in that the latter are designed to capture the preferences and penalties of using parsing grammar rules, while the former are designed to capture the context sensitive preferences and penalties. ILT constraints resemble the graded constraints used by the plan-inference model, but some constraints particularly relevant to the focusing mechanism used in plan inference are not implemented because focusing heuristics are not relevant in the FSM model.

Currently, we implemented the following ILT constraints:

- **loc-possible**: decides to what extend an expression is a locative.
- **possible-time**: grades the goodness of a temporal expression with respect to the dialogue date.
- **prefered-time**: grades the goodness of a temporal expression with respect to the calendar and temporal focus.
- **tense-prefered**: assigns penalty to past tense.
- **possessor-prefered**: assigns penalty to third person pronouns.
- **opening-prefered**: assigns preference to opening if opening is in focus; assigns penalty to be an opening if still the same speaker.
- **fixed-prefered**: assigns penalty to a fixed-expression with no speech act; assigns preference to a fixed-expression if no time is mentioned.
- **closing-prefered**: assigns preference to a fix-expression to be a closing if a closing is recognized in focus.
- **speaker-reference**: assigns speaker reference preferences with respect to sentence types.

4.4.2 Combination of context-based and non context-based scores

The two scores from the FSM discourse processor are combined with the three scores from the parser (statistical score, parser score, and counter values) for disambiguation. For combination of the scores, we experimented with the neural net approach and the confidence combination method.

The neural net approach

Again we use our simple three layer neural net trained by the back-propagation learning algorithm for combination of these scores. Now the neural net takes as input five sets of five scores, each corresponding to up to five possible ILTs for each processed sentence. The five scores are the grammar preference value, the statistical score, the parse score, the speech act probabilities, and the sum of the ILT constraints. This amounts to a total number of 25 input units. If a processed sentence has fewer than five possible ILTs, we fill the empty slots by setting the units to zero. We use the same mechanism for the output layer as described in Section 3.4.3.

Currently the network has its parameters set at the following values:

```
(setf *eta* 0.01) ; learning rate
(setf *alpha* 0.9) ; weight change momentum
(setf *input-size* 25) ; number of input nodes
(setf *hidden-size* 4) ; number of hidden nodes
(setf *output-size* 5) ; number of output nodes
```

The confidence combination

The confidence combination method is implemented in the same way as described in Section 3.4.3.

4.5 Advantages and Disadvantages of a FSM-Based Discourse Processor

Compared with the plan-inference-based model, a finite state model of discourse is often argued to be less knowledge extensive, and thus not sophisticated enough to handle such tasks as ellipsis resolution, which requires the recording the tree structure of discourse context. In our case, since our plan-inference model has to be robust enough to cover a wide range of constructions, we have left out the knowledge intensive aspects of plan inferencing. Consequently, we have not observed a distinct advantage of the plan-inference model over a simpler model, such as the FSM model, in understanding spontaneous speech.

The simplicity of the FSM model makes it easy to implement. It is also fast and efficient enough to be combined with other components of the system in real-time translations. Moreover, constraint processing is separate from the dialogue model. We believe that this separation makes it easier to extend the model to new domains, since the dialogue model does not have to be changed when new domain-dependent constraints are introduced.

Similar with the plan-inference model, the FSM discourse processor also suffers from the cumulative error problem. We describe some experiments to reduce the cumulative error in the next section.

5 Cumulative Error in Discourse Processing

In this section, we first describe our efforts of dealing with the cumulative error problem in the FSM model. We then outline some preliminary ideas of reducing cumulative error in the plan inference model.

5.1 FSM and the cumulative error

When the grammar, statistical and FSM predictions are combined for speech act recognition, it is possible that an inconsistency occurs between the non-context based grammar predictions and the context based FSM predictions. The inconsistency can result from any among unexpected inputs, missing information, out of domain utterances, or simply misrecognized speech act. We tested two approaches for dealing with the conflicting predictions: a *jumping context approach* and a *hypothesis tree approach*. We describe the two approaches below.

5.1.1 Jumping context approach

The rationale behind the jumping context approach is that while we recognize the predictive power of a statistical model, a finite state machine, or a plan-based discourse processor, we abandon the assumption that dialogue act sequences are always ideal in spontaneous speech. Instead of trying to incorporate the current input into the dialogue context, we accept that speech act sequences can at times be imperfect. Instead of following the expectations provided by the context, we assume there is an inaccurate context and there is a need to re-establish the state in the discourse context. In such cases, we give more trust to the grammar predictions, assessing the current position using syntactic and semantic information. When there is more than one speech act proposed by the grammar component, we use speech act unigrams to choose the most likely one in the corpus. The context state will then be updated accordingly using the grammar prediction. In the graph representation of the finite state machine, this corresponds to allowing empty arc jumps between any two states. Note that this jump from one state to another in the finite state machine is *forced* and *abnormal* in the sense that it is designed to cater to the abrupt change of the flow of dialogue act sequences in spontaneous speech. Thus it is different from transitions with null symbols, which record legal transitions between states. The algorithm for this approach is described in Figure 11. We demonstrate later that this approach gives a better performance than one which trusts context in the case of conflicting predictions.

As an example, consider the following dialogue excerpt in Table 2. After speaker S2 accepts S1's suggestion and tries to close the negotiation, S2 realizes that they have not decided on where to meet. The utterance *no* after the closing *chau* does not fit into the dialogue model, since the legal dialogue acts after a `closing` are `closing`, `confirm-appointment` or `request-suggestion` (see Figure 10). When the standard dialogue model is observed (marked by *Strict Context* in Table 2), the utterance *no* is recognized as `closing` since `closing` is the most probable speech act following the previous speech act `closing`. If upon seeing this conflict we instead trust the grammar prediction (marked by *Jumping Context* in Table 2), by recognizing *no* as a `reject`, we bring the dialogue context back to the stage of negotiation. Trusting the grammar, however, does not imply that we should abandon context altogether. In the test set in the experiments discussed in the next section, context represented by the FSM has shown to be effective in reducing the number of possible speech acts predicted by the grammar component.

5.1.2 Hypothesis tree approach

The rationale behind the hypothesis tree approach is that instead of producing a single speech act hypothesis at the time an utterance is passed to the discourse processor, we delay the decision until a later point. In doing so, we hope to reduce cumulative error due to misrecognition because of early commitment to a decision. Specifically, we keep a set of possible speech act hypotheses for each input utterance as contextual states for future predictions. Each context state may in turn be followed by more than one speech act hypothesis for the subsequent utterance, thus yielding a tree of possible sequences of speech act hypotheses. The hypothesis tree is expanded within a beam so that only a certain total number of branches are kept to avoid memory explosion. When the turn shifts between speakers, the hypothesis path with the highest probability (calculated by multiplying speech act bigram probabilities in that path) is chosen as the best hypothesis for the sequences of ILTs in that turn. Each ILT is then updated with its respective speech act in the chosen hypothesis. For each new turn, the last context state in the best hypothesis of the previous turn is used as the starting root for building new hypothesis tree. Figure 12 gives the algorithm for the hypothesis tree approach.

Dialogue Utterances	Strict Context	Jumping Context
S1: QUÉ TE PARECE EL LUNES NUEVE ENTONCES (HOW IS MONDAY THE NINTH FOR YOU THEN)	`suggest`	`suggest`
S2: PERFECTO (PERFECT)	`accept`	`accept`
CHAU (BYE)	`closing`	`closing`
NO (NO)	`closing`	`reject`
ESPÉRATE (WAIT)	`address`	`address`
NO (NO)	`closing`	`reject`
ESPÉRATE (WAIT)	`address`	`address`
ALGO PASÓ MAL (SOMETHING BAD HAPPENED)	no parse	no parse
DÓNDE NOS VAMOS A ENCONTRAR (WHERE ARE WE GOING TO MEET)	`request-suggestion`	`request-suggestion`
S1: NO (NO)	`reject`	`reject`
ESPÉRATE (WAIT)	`address`	`address`
SÍ (YES)	`acknowledge`	`acknowledge`
DÓNDE NOS ENCONTRAMOS (WHERE ARE WE MEETING)	`request-suggestion`	`request-suggestion`

Table 2: **An Example Dialogue**

As in the jumping context approach, the predictions of speech acts for each utterance are the combined result of the context based FSM predictions and non-context based grammar predictions. The intersection of both predictions gives the possible speech acts which are consistent with both the dialogue model and the default functions of the input utterance. When there is no intersection, we face the choice decision of trusting the context based FSM predictions or the non-context based grammar predictions. We demonstrate later that, for the hypothesis tree approach, again, trusting grammar predictions gives better result than strictly following context predictions at the time of conflicting predictions.

5.2 Plan-inference and cumulative error

Recall that in the jumping context approach for the FSM cumulative error reduction, there are two major steps: evaluation of the context and update of the context. We'll discuss how we can incorporate these two steps in the plan-based discourse processor.

First, to conduct the evaluation step, we evaluate the inference lines generated from semantic representations first before applying the focusing heuristics to incorporate them into the plan tree. This is done by (1) pulling out the speech acts from these inference lines, (2) calculating the intersection between the speech acts from the inference lines and the speech acts from grammar. The inference lines with the resulting speech acts should both observe context and semantic predictions.

In the case of prediction conflicts, when we trust the grammar predictions, the inference lines constructed by the grammar predictions will not be able to be incorporated into the discourse context. Currently, we are still investigating how to update the discourse model when such conflicts occur. This remains as a major research issue in our future work.

6 Evaluation

7 Summary and New Directions

Our work on the Enthusiast discourse processors leaves open several research topics. The first important issue is how to recover from cumulative error. Our experience with incorporating discourse context into the whole speech system showed that cumulative error in the discourse context can have an adverse effect on the system performance, especially in spontaneous dialogues where disfluencies and unexpected input are abundant. We have experimented with different techniques to evaluate the quality of the discourse context and reduce cumulative error. We hope in the future to experiment with techniques from AI planning for dealing with unexpected input for the cumulative error problem in discourse.

Another open question is the nature of the discourse model. While the plan-based discourse model provides detailed and global contextual model, this approach requires a lot of computing resources because of complexity. The finite-state machine and speech act bigram models are fast and efficient alternatives for recognizing speech acts, but these approaches cannot represent detailed domain and discourse constraints as represented in the plan-based model. Since our work has been focused on a single domain, it is unclear to us of the scalability of the different processing models to new domains (possibly multi-domains or open domains). We hope to research these issues in a multi-domain or open domain setting, and investigate how we can adapt the plan inference and finite state models quickly to new (sub)domains.

Another direction of our future work is to develop user support tools for constructing, testing, and learning discourse models. Currently the plan operators and finite state machine discourse grammars are designed by system designers based on corpus analysis. We would like to develop a user interface for rule writing and debugging which would enable linguists to write and test discourse rules. We would also like to experiment with machine learning approaches for automatically learning discourse grammar rules from corpus.

We would also like to use discourse information for out-of domain detection. Domain detection will identify utterance as in one of the domains that we are treating or as out-of-domain. Out-of-domain utterances do not contribute to the completion of a specific task, but they may contribute background information. Many out-of domain utterances will not be analyzable by our parsers because of unknown vocabulary. However, we may be able to identify a rhetorical relation such as *explanation* or *reason* based on surrounding function words such (e.g., *(in order) to*, *so (that)*, etc.) and the current discourse structure.

True speech understanding requires that individual utterance meanings be understood in the context of the larger dialogue structure. We plan to apply discourse states to select an appropriate language model for speech recognition and parsing to reduce the perplexity and improve recognition and parsing accuracy.

References

[1] Jan Alexandersson. Plan recognition in verbmobil. Technical Report Verbmobil-Report 81-95, DFKI GmbH, 1995.

[2] J. F. Allen. *A Plan-Based Approach to Speech Act Recognition*. PhD thesis, University of Toronto, School of Computer Science, 1979.

[3] J. L. Austin. *How to Do Things with Words*. Claredon Press, Oxford, UK, 1975.

[4] R. C. Berwick. From rules to principles in language acquisition: A view from the bridge. In D. Powers and L. Reeker, editors, *Proceedings of Machine Learning of Natural Language and Ontology*, 1991.

[5] U. Bodenhausen and S. Manke. Connectionist architectural learning for high performance character and speech recognition. In *ICASSP-93, International Conference on Acoustic, Speech and Signal Processing*, 1993.

[6] U. Bodenhausen and S. Manke. Tuning by doing: Flexibility through automatic structure optimization. In *EUROSPEECH*, 1993.

[7] Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. The MIT Press, Cambridge, Massachusetts, 1990.

[8] J. Carroll and T. Briscoe. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1), 1993.

[9] K. W. Church and W. A. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103, 1991.

[10] R. Clark. A computational model of parameter setting. In D. Powers and L. Reeker, editors, *Proceedings of Machine Learning of Natural Language and Ontology*, 1991.

[11] G. Hinton D. Rumelhart and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[12] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, 1859.

[13] H. P. Grice. *Logic and Conversation*, pages 113–128. Academic Press, New York, 1975. In P. Cole (ed.), Syntax and Semantics, vol. 9: Pragmatics.

[14] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[15] H. Iida, T. Yamaoka, and H. Arita. Predicting the next utterance linguistic expressions using contextual information. *IEICE Trans. Inf. & Suyst.*, E76-D(1), 1993.

[16] Fred Jellink. Self-organized language modelling for speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann, 1990.

[17] Lorraine F. R. Karen. Identification of topical entities in discourse: a connectionist approach to attentional mechanisms. *Connection Science*, 2 (1):103–122, 1990.

[18] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987.

[19] A. Kim. Graded unification: A framework for interactive processing. In *Proceedings of the Association for Comoutational Linguistics*, 1994.

[20] K. Koguga, M. Kume, and H. Iida. Illocutionary act based translation of dialogues. In *The third international conference on theorectical and methodological issues in machine translation of natural language*, 1990.

[21] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[22] Lynn Lambert. Recognizing complex discourse acts: A tripartite plan-based model of dialogue. Technical Report Technical Report No. 93-19, Department of Computer and Information Sciences, University of Delaware, 1993.

[23] Lynn Lambert and S. Carberry. A tripartite plan-based model of dialogue. In *Proceedings of the 29th ACL*, pages 47–54, 1991.

[24] A. Lavie. *A Grammar Based Robust Parser For Spontaneous Speech*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1995.

[25] Alon Lavie, D. Gates, M. Gavaldá, L. Mayfield, A. Waibel, and L. Levin. Multi-lingual translation of spontaneously spoken language in a limited domain. In *Proceedings of the COLING*, 1996.

[26] L. Levin, O. Glickman, Y. Qu, D. Gates, A. Lavie, C. P. Rosé, C Van Ess-Dykema, and A. Waibel. Using context in machine translation of spoken language. In *Theoretical and Methodological Issues in Machine Translation*, 1995.

[27] Diane J. Litman and James F. Allen. Discourse processing and commonsense plans. In *Intentions in Communications*, 1990.

[28] R. M. Losee. Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. *Information Processing And Management*, 1995.

[29] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1994.

[30] David G. Novik. Controlling interaction with meta-acts. Technical Report Technical Report CS/E 91-001, Dept. of Computer Science and Engineering, Oregon Graduate Institute, 1991.

[31] Thomas S. Polzin. Parsing spontaneous speech: A hybrid approach. In *Workshop on Combining Connectionist and Symbolic Processing, ECAI-94*, Amsterdam, The Netherlands, 1994.

[32] Y. Qu, B. Di Eugenio, A. Lavie, L. Levin, and C. P. Rosé. Minimizing cumulative error in discourse context. In *ECAI Workshop on Spoken Dialogue Processing*, 1996.

[33] Y. Qu, C. P. Rosé, and B. Di Eugenio. Using context for ambiguity resolution. In *COLING*, 1996.

[34] Yan Qu. Reducing cumulative error in discourse processing, 1996. Unpublished Manuscript.

[35] Norbert Reitinger and Elisabeth Maier. Utilizing statistical dialogue act processing in verbmobil. In *AAAI Spring Symposium*, 1995.

[36] C. P. Rosé. The structure of multiple-headed negotiations. Technical Report CMU-PHIL-65, Carnegie Mellon University, 1995a.

[37] C. P. Rosé. Plan-based discourse processor for negotiation dialogues, 1995b. unpublished manuscript.

[38] C. P. Rosé, B. Di Eugenio, L. S. Levin, and C. Van Ess-Dykema. Discourse processing of dialogues with multiple threads. In *Proceedings of the ACL*, 1995.

[39] C. P. Rosé and Yan Qu. Automatically learning to use discourse information for disambiguation, 1996. unpublished manuscript.

[40] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK, 1969.

[41] E. Siegel. Competitively evolving decision trees against fixed training cases for natural language processing. In K. E. Kinnear, editor, *Advances in Genetic Programming*. MIT Press, 1994.

[42] B. Suhm, L. Levin, N. Coccaro, J. Carbonell, K. Horiguchi, R. Isotani, A. Lavie, L. Mayfield, C. P. Rosé, C. Van-Ess Dykema, and A. Waibel. Speech-language integration in a multi-lingual speech translation system. In *Proceedings of the AAAI Workshop on Integration of Natural Language and Speech Processing*, 1994.

[43] David R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, Department of Computer Science, University of Rochester, Rochester, New York, 1994.

[44] A. Waibel, T.S. Polzin, U. Bodenhausen, F.D. Buø, N. Coccaro, H. Hild, and B. Suhm. Hybrid connectionist and classical approaches in janus. an advanced speech-to-speech translation system. In *ICONNIP-94*, 1994.

[45] R. Michael Young. Reasoning about the effects of communication on beliefs. In *?*, 1994.

```
;;; Structure for a calendar day
(defstruct day
  month                  ; name of the month
  day                    ; number of the day in the month
  day-of-week            ; name of the day of the week
  year                   ; year
  sp1.sched              ; speaker1's schedule for this day
  sp2.sched              ; speaker2's schedule for this day
  constraints)           ; constraints already included in

;;; Structure for a calendar week
(defstruct week
  number                 ; number of days in this week structure
                         ; (some weeks are spread out over
                         ; two week structures since they span two
                         ; different months.
  days)                  ; list of day structures in this week structure

;;; Structure for a calendar month
(defstruct month
  month                  ; name of the month
  weeks)                 ; list of week structures included

;;; Structure for a calendar year
(defstruct year
  year                   ; year
  months)                ; list of months

;;; Structure for a constraint
(defstruct constraint
  status                 ; status of time slots in times (i.e. suggested)
  times                  ; list of tstructs which cover the time slots
                         ; which the constraint applies to
  speaker)               ; speaker who's schedule the constraint
                         ;should apply to
```

Figure 6: Data structures for calendar program

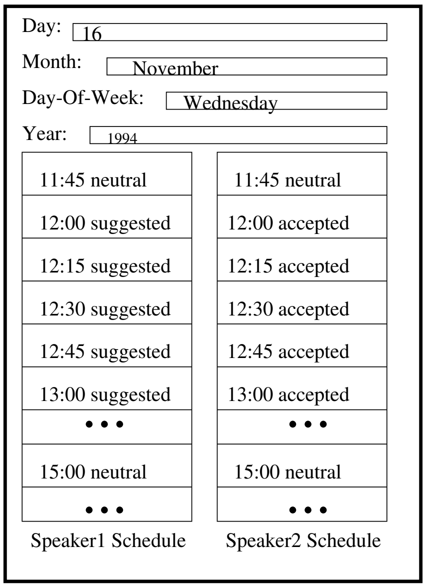
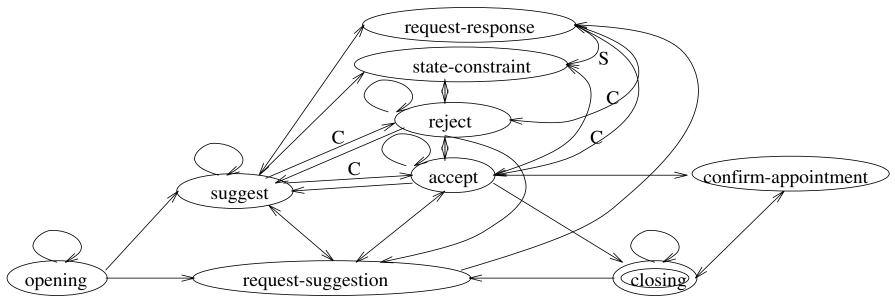


Figure 7: **The Main Component of the Finite State Machine**

```
((Header 0 (Reject (_speaker1 VAR PARTICIPANT)
                   (_speaker2 VAR PARTICIPANT)
                   (_action VAR PROPOSITION)
                   (_rejection VAR PROPOSITION)))
 (Type Decomposition)
 (Applicability.Conditions )
 (Constraints
    (NIL 0 NIL prefered-time (_rejection VAR PROPOSITION))
    (NIL 0 NIL rejection-to-prop (_rejection VAR PROPOSITION)
                                 (_action VAR PROPOSITION)))
 (Preconditions )
 (Specializes )
 (Decomposition
    (NIL + NIL Reject-Form (_speaker1 VAR PARTICIPANT)
                           (_speaker2 VAR PARTICIPANT)
                           (_rejection VAR PROPOSITION)))
 (Specializations )
 (Primary.Effects
   (NIL 0 NIL believe (_speaker2 VAR PARTICIPANT)
                      (relates PROP PROPOSITION
                               (_speaker1 VAR PARTICIPANT)
                               (_action VAR PROPOSITION)
                               (_rejection VAR PROPOSITION))
                       ((3 3) belief BELIEF.INTERVAL)))
 (Goals
   (NIL 0 NIL believe (_speaker2 VAR PARTICIPANT)
                      (relates PROP PROPOSITION
                               (_speaker1 VAR PARTICIPANT)
                               (_action VAR PROPOSITION)
                               (_rejection VAR PROPOSITION))
                        ((3 3) belief BELIEF.INTERVAL)))
 (Level.type Discourse))
```

Figure 8: **An Example Plan Operator with Graded Constraints**



The state **opening** is the initial state. The state **closing** is the final state.
All other states are non-final states.

Figure 9: **The Main Component of the Finite State Machine**

```
(setf *network*
'(
(Initial (opening))
(Final (closing))
(from accept to suggest)
(from accept to accept)
(from accept to reject)
(from accept to request-suggestion)
(from accept to request-response)
(from accept to state-constraint)
(from accept to closing)
(from closing to closing)
(from closing to request-suggestion)
(from opening to opening)
(from opening to request-suggestion)
(from opening to suggest)
(from request-suggestion to suggest)
(from request-suggestion to request-suggestion)
(from request-suggestion to reject)
(from request-response to accept)
(from request-response to reject by D)
))
```

Figure 10: Example FSM Grammar

```
context-state = 'start
FOR each input ilt
   context-predictions = predictions from the FSM
                         given context-state
   grammar-predictions = a-speech-act in input ilt
   Intersect context- and grammar-predictions
   IF intersection is not empty,
      use bigrams to rank the speech acts in intersection
      return the most probable follow up speech act
   ELSE ;;; use grammar predictions
      IF more than one speech act in a-speech-act
         use unigrams to rank the possible speech acts
         return the most probable speech act
      ELSE return a-speech-act
   update context-state using the returned speech act
```

Figure 11: Algorithm for Jumping Context Approach

```
hypothesis-tree = '(((start)))
ILTS = nil
FOR each input ilt
   IF still in the same turn
      push ilt into ILTs
      FOR each path in the hypothesis-tree
          context-state = last state in the path
          get speech act predictions for input ilt
          update hypothesis-tree
   ELSE ;;; turn shifts
      choose the path with the highest probability
      update ilts in ILTS with their respective speech act
          prediction in the chosen path
      ILTS = nil
      context-state = last state in the chosen path
      hypothesis-tree = (((context-state)))
      push ilt into ILTS
      get speech act predictions for input ilt
      update hypothesis-tree
   rank the paths in the hypothesis-tree and
   trim the tree within a beam.
```

Figure 12: Algorithm for the Hypothesis Tree Approach

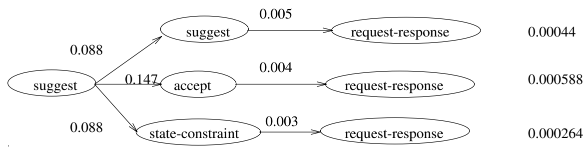


Figure 13: Example Hypothesis Tree