Probabilistic Text Understanding

Robert P. Goldman[*]
Department of Computer Science
Tulane University
301 Stanley Thomas Hall
New Orleans, LA 70115

Eugene Charniak[*]
Department of Computer Science
Brown University
Box 1910,
Providence, RI 02912
rpg@cs.tulane.edu, ec@cs.brown.edu

Abstract

We discuss a new framework for text understanding. Three major design decisions characterize this approach. First, we take the problem of text understanding to be a particular case of the general problem of abductive inference. Second, we use probability theory to handle the uncertainty which arises in this abductive inference process. Finally, all aspects of natural language processing are treated in the same framework, allowing us to integrate syntactic, semantic and pragmatic constraints. In order to apply probability theory to this problem, we have developed a probabilistic model of text understanding. To make it practical to use this model, we have devised a way of incrementally constructing and evaluating belief networks. We have written a program, Wimp3, to experiment with this framework. To evaluate this program, we have developed a simple 'single-blind' testing method.

Keywords: Natural language processing, Bayesian, belief networks, Bayesian networks.

This is a hand-assembled "postprint" of an old article: Goldman, R.P. and Eugene Charniak. "Probabilistic text understanding." *Statistics and Computing* 2 (1992): 105-114.

[*]This work has been supported in part by the National Science Foundation under grants IST 8416034 and IST 8515005 and Office of Naval Research under grant N00014-79-C-0529. The majority of the work described was done when both authors were at Brown University. The first author was supported by a summer fellowship from the Tulane University Senate Committe on Research while revising this article.

1 Introduction

For some time we have been interested in the problems posed by uncertainty in text understanding, in particular the problem of representing and reasoning about alternative interpretations of natural language texts. We are concerned with problems like pronoun reference, prepositional phrase attachment, and lexical ambiguity. We are particularly interested in plan-recognition as it is needed in text understanding: understanding the meanings of stories by understanding the way the actions of characters in the story serve purposes in their plans. Our work builds upon earlier work in script- and plan-based understanding of stories like that of Cullingford (1978), Wilensky (1983), Wong (1981), Charniak (1986) and Norvig (1987).

There are three features of our approach which distinguish it from other approaches to natural language processing. First, we treat the problem of natural language understanding as an abduction problem. Second, we use probability theory as our theoretical framework. Finally, we tightly integrate all aspects of natural language processing, syntax, semantics and pragmatics (largely plan-recognition). Our research is also set apart from earlier efforts by the use of a single-blind testing methodology.

We see text understanding as a particular case of the problem of abduction (reasoning from effects to causes), or diagnosis. This approach to natural language understanding is outlined in Charniak and McDermott (1985) and Hobbs *et al.* (1988). In particular, for the case of simple, declarative text, we simplify by viewing the language user as a transducer. The language user observes some thing (event or object) in the 'real world', and translates this thing into language. Our task is to reason from the language to the intentions of the language user and thence to the events described.

We have adopted probability theory as our framework because it is the most thoroughly-understood way of reasoning about uncertainty. Unlike other ways of treating uncertainty (e.g., default logic), probability provides a unified 'currency' in which we can weigh evidence from different sources. Furthermore, it is at least in some sense normative (see for example the proof given in Cox (1946)). Finally, techniques for graphically representing probability distributions have made the task of drawing up and reasoning with probabilistic models much less onerous than they were once thought to be.

The third distinguishing feature of our approach is the extent to which it integrates all levels of processing. Previous work in text understanding has almost universally assumed that the text will be pre-processed by an intelligent parser (e.g., Cullingford (1978), Hobbs *et al.* (1988), Norvig (1987), Wilensky (1983)). We, on the other hand, read the text word-by-word through a parser that simply returns all possible parses of the input. Because the structure of the sentence is described in the same framework as its semantics and pragmatics, constraints from different levels can be assembled into a global interpretation. For example, in deciding the proper attachment of the prepositional phrase "with some poison" in "Janet killed the boy with some poison." we can make use of our knowledge about murder, or about a particular poison-wielding boy that Janet dislikes.

We have developed a 'single-blind' technique for testing our story understanding program, Wimp3. One of us has developed a set of pairs of synonymous stories whose syntax and vocabulary were restricted so that they could be handled by Wimp3's parser. These pairs were randomly assigned to either test or training set. The other member (the first author) developed the program until it was able to handle the training set, and then used it to align the training set stories with their counterparts in the test set (which were kept hidden from him). We believe that this testing methodology allows us to eliminate some of the problems of 'wishful programming,' while still being within the capabilities of small research groups.

Our previous work in this area was codified in the program Wimp2, described in Charniak and Goldman (1988). Wimp2 used a multiple-context deductive database, based on the Assumption-based Truth Maintenance System (ATMS) presented in deKleer (1986). The contexts of Wimp2's database were tagged with probabilities.[1] Having a multiple-context database to cache inferences made it possible to consider simultaneously different interpretations, such as alternate meanings of words, or different parses, and their consequences. The probabilities made it possible to allocate Wimp2's computational resources to considering the most likely candidate explanations.

We were unhappy with Wimp2 for three reasons. First of all, it was difficult to reconcile the logical approach which motivated the use of the ATMS with the use of probability theory. This problem manifested itself particularly acutely in difficulties we had assigning a clear semantics to the probabilities we used. This difficulty arises because of the difference between the logical operation of material implication and the probabilistic operation of conditioning. Nilsson (1986) discusses this issue. On a more concrete level, the system was making every inference twice over: the system reasoned deductively using the ATMS, and in an ad-hoc probabilistic way when managing the probabilities of the ATMS' contexts. Finally, the deductive framework is unable to distinguish evidential and causal support for propositions. When a new justification is added to a proposition, one cannot tell whether it represents evidence for that proposition, in which case it would confirm hypotheses which explain that proposition; or if it represents an alternative explanation, in which case it should compete with other explanations. I.e., when writing rules of the form $A \rightarrow B$, there is no way to distinguish between the interpretations "A is evidence for possible explanation B" and "A can cause B to be true." For these reasons, we decided to try a wholly probabilistic approach to the problem.

Our current program, Wimp3, operates by turning problems of text interpretation into probability questions of a form like "What is the probability that this word has this sense, given the evidence?" (where the evidence is the text of the story so far). Wimp3 does this on a word-by-word basis. The probability questions are formulated in a language we have developed for this purpose. This formulation is given additional structure by a graphical representation, belief networks. In the next section of this paper, we will outline the language we have developed to express linguistic problems in terms of random variables. Following that, we will show how these sets of random variables can be structured as belief networks, and how this structure helps us assess the quantities we need for our probabilistic models. We will then show how Wimp3 actually constructs and solves these problems. We conclude with a discussion of the single-blind testing of Wimp3, and brief summary. Sample test data for the single-blind tests are given in an appendix.

2 The Formalism

In adopting a probabilistic approach, our first problem was to come up with a clear understanding of what probabilities we were to manipulate. We have created a simplified formal language suited to expressing facts about a text's structure and meaning. We have devised a semantics for this language which allows us to treat its formulae as random variables. We show that this semantics gives us the guidance we need to devise a consistent set of probabilities for use by a story-understanding program. A slightly earlier version of this language is described in Charniak and Goldman (1990) and Charniak and Goldman (1989); we will only have space for a brief discussion here.

We will start by describing the notation of our language and showing how it is used to formulate the text understanding problem. For convenience's sake, we will proceed in a way which follows the traditional syntax–semantics–pragmatics distinction. First we will show how the language can be used to describe the input text itself. Then we will go on to talk about how it can express semantic propositions. Finally, we will show the part of the language used for plan-recognition, and how this provides contextual information.

We use four kinds of propositions in formulating text understanding problems:

[1] At the same time as we developed this database architecture, Laskey and Lehner (1988), D'Ambrosio (1988a) and D'Ambrosio (1988b) reported probabilistic ATMS' that were very similar.

1. **(word-inst** x *root***)** The token x is an instance of the root form *root*. We use the term 'token' to emphasize the distinction between a *word* and a particular *use* of that word in a text.
2. **(syn-rel** *type* x y**)** The tokens x and y are in a syntactic relation of *type*.
3. **(inst** x *type***)** The thing x is of *type*. x is some thing in the real world, typically one named by a token in the text. We use the word 'thing' here because x may be an object, an event, or a state. We will sometimes use the word 'entity' for the same purpose.
4. **(==** x y**)** The objects x and y are the same. We use the symbol == rather than = because, for the sake of efficiency, we use an antisymmetric 'better-name' relation, rather than true equality. This distinction need not trouble the reader.

This notation is similar to the first order predicate calculus, but all terms are ground, i.e., no quantifiers are used. These propositions (we will also refer to them as 'statements') will be treated as random variables over the sample space $\{true, false\}$. In the following paragraphs we will give a more concrete description of the use of this formalism.

The parser and morphological analyzer component of Wimp3 generate the statements that describe the natural-language input. These statements are of two types:

- specifying the words used (*word-inst*), and
- specifying relations between words used (*syn-rel*).

For example, the sentence "Jack gave Mary the ball." (1) would be translated into:

(word-inst word1 Jack)
(word-inst word2 give)
(syn-rel subject word1 word2)
(word-inst word3 Mary)
(syn-rel indirect-object word3 word2)
(word-inst word4 ball)
(syn-rel object word4 word2)
(syn-rel det word4 the)

The morphological analyzer translates inflected forms like 'gave' into root forms. We simplify by ignoring issues of tense and time. In the case of syntactic ambiguity, the parser will generate and describe all possible parses of the input; semantic and pragmatic information will be used to choose between them. If the sentence were "Mary killed the boy with the poison." (2), the parser would give both (syn-rel with poison12 boy11) and (syn-rel with poison12 kill10). NOTE: From now on, for ease of understanding, we will use more descriptive names, like poison12 and boy11, rather than word1, word2, etc.

For semantic processing, we need to be able to express hypotheses about the denotations of the various words. Associated with each word instance is a constant representing the denotation of that word. We will write this as the same constant, but in boldface. We might more clearly write (denotation word*n*), but this is too bulky. For example, if ball27 was the constant representing the denotation of 'ball' in (1) above, two possible denotations might be expressed as:

(inst ball27 bouncing-ball)
(inst ball27 cotillion)

In order to express case relations, we have functions for the various cases. To return to example (2) ("Mary killed the boy with the poison."), the case relations corresponding to the different prepositional phrase attachments would be:

```
(== (accompaniment boy11) poison12)
(== (instrument kill10) poison12)
```

Moving into the gray area between semantics and pragmatics that Hobbs and Martin (1987) call 'local pragmatics,' equality statements are used to express coreference. For example, part of the task of understanding "Jack gave Mary the ball. She liked it." (3) is to reason from

```
(inst mary24 girl-)
(inst she28 girl-)
```

to (== she28 mary24).

Finally, we need to be able to express a plan-based understanding of our texts, such that when reading a story like "Jack went to the liquor-store. He paid for some bourbon." (4) our program can understand the entire story as a description of a plan to buy liquor.

Making such inferences requires generic information about plans and their parts. To do so, we have a frame-based database of events and objects. Events and objects are represented by instances of frames, which are organized into an *isa-hierarchy*. In such a system, frames representing different types are arranged in a taxonomy according to the *isa* relation. For example, mammal isa animal, elephant isa mammal, etc. A particular elephant, is an instance of the frame elephant. A frame which isa another frame inherits the features of its supertype. So elephants are warm-blooded, bear live young, have mammary glands, etc.

Components, or *slots*, of frames are represented by functions. For example, the trunk of an elephant called Clyde, might be represented by the function (trunk-of Clyde). The actions which make up a particular plan are slot-fillers of that plan instance. For example, in order to shop at a store, one must first go to the store. If we have a particular instance of shopping, say shop1, we would refer to the corresponding event of going to the store as (go-step shop1). Finally, we need to be able to represent constraints between the different slots of a plan. For example, we know that when one goes to a store as part of a plan to shop, one goes to the particular store at which one intends to buy something. The destination of the go-step of the shopping is the same as the store-of the shopping. In our notation: (== (destination (go-step shop1)) (store-of shop1)).

Here is an example of the kind of generic knowledge we have given Wimp3:

```
(inst ?shop shopping-) → (and (inst (go-stp ?shop) go-)
                              (== (agent (go-stp ?shop)) (agent ?shop))
                              (== (destination (go-stp ?shop))
                                  (store-of ?shop)))
```

Currently, Wimp3's database contains 148 frames, which are translated into 513 constraints in its internal database format. The internal format is resembles the example above, but 'syntactic sugar' is removed, and typed variables are used.

For plan-recognition in this framework, we need another kind of proposition: the proposition that a given frame-instance (say went2) fills a slot in a given kind of plan (say robbery). This would be written

```
(and (inst robbery27 robbery)
     (== (go-step robbery27) went2)) (5)
```

The atom *robbery27* is written in a different typeface to indicate that it represents a different kind of constant. *robbery27* is a function of went2. Analogously to the the denotation constants, these constants might more clearly be written (robbery-explanation-of went2). A detailed discussion of plan-recognition in Wimp3 is given in Charniak and Goldman (1991).

These propositions are virtually identical to those in our earlier logic for semantic interpretation (presented in Charniak and Goldman (1988)), but their semantics is different. Following probability theory, the expressions of our language get their semantics from a sample space. This sample space is the set of all possible stories from some restricted universe. The word constants denote random variables whose domain is particular words. Likewise, the denotations of words are random variables whose domain is the set of entities (including events). Propositions are random variables whose domain is the set {true,false}. A more extensive discussion of the semantics of our language can be found in Charniak and Goldman (1989).

3 Network representations

In the past, it has been difficult to apply probability theory to reasoning problems because such problems did not have sufficient structure. It appeared that in order to apply probability theory, it would be necessary to have a gigantic joint probability table, giving the probability of all possible combinations of the propositions of interest. However, recently, there has been renewed interest in graphical representations for probability distributions, such as Markov fields, belief networks and influence diagrams. Belief networks have provided us with a way of structuring the problem of text understanding as a problem of probabilistic inference, and have simplified the problem of acquiring the requisite numbers.

Belief networks are a way of representing probabilistic dependency information in directed acyclic graphs. Such graphical representations have been used in various fields for some time (Lauritzen and Spiegelhalter (1988) cite uses in path analysis as early as 1921). These graphs make possible qualitative reasoning about influence, and for distributions with desirable properties of conditional independence, make it possible to efficiently calculate posterior probabilities. Judea Pearl's book (Pearl (1988)) gives a thorough account of the properties of such networks.

The nodes in a belief network represent random variables, and the edges represent direct dependence. We follow Pearl's suggestion that edge direction be assigned causally: nodes at the tails of edges should be (direct) causes of the nodes at their heads. There are three advantages to belief networks as representations for probability distributions. First, properties of conditional independence can be read off a belief network. Second, the probability distribution corresponding to a belief network may be represented locally. For each node, it suffices to provide a conditional probability distribution for each combination of values of its parent nodes. Finally, while in general the problem of determining the posterior distribution of a partially-instantiated belief network is NP-hard (proof given in Cooper (1987)), considerable attention has been devoted to finding efficient approaches to evaluating such networks.

We give a partial belief network representation of the analysis of example the sentence "Jack went to the liquor-store." (6) in Figure 1. This fraction of the network is intended to capture the following relations: A liquor-shopping plan can cause there to be a go action. It will also cause there to be a liquor-store, the store-of the liquor-shopping. The liquor-store mentioned in the story, might be this liquor-store. The store-of the liquor-shopping will be the destination of the go-step of the liquor-shopping. These relations might be realized in the text sequence we have observed.

There are two things to be noted about this diagram. First of all, the 'explanatory' variables (e.g., liquor-shop4) are tied to the actions of characters, rather than to all entities mentioned. The liquor-store simply supports the liquor-shop hypothesis, and that only in the presence of evidence for it being the destination of Jack's going. This avoids having objects call into consideration every plan in which they could play a role. For example, we don't consider a shopping explanation when we read "Jack bombed the supermarket."

Note also that this is an immensely simplified version of the networks which are, in fact, used in our program. First of all, we have suppressed many of the nodes that would appear in the

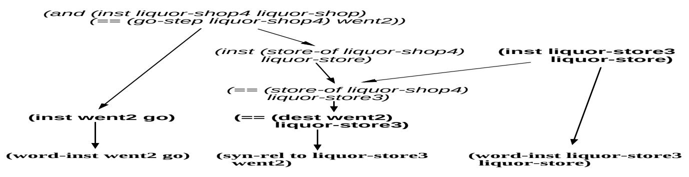

Figure 1: A belief network representation of the story "Jack went to the liquor-store...."

network for this interpretation (e.g., the nodes specifying Jack as the agent of the go). Second, the networks we construct are not restricted to contain only nodes which are part of *correct* interpretations of the text!

4 Quantifying the networks

The belief network representation makes it easier for us to specify the probability distributions we need. It permits us to express the distribution as a collection of conditional probabilities involving only a few nodes (no more than 3 in our current program). This representation is compact, and these low-order probabilities are easier to assess subjectively. While in principle we could collect statistics to find these distributions, this is not practical. In this section we will give a brief discussion of the kinds of conditional probabilities we need. For the sake of clarity, we will proceed from the leaves of the belief networks to their roots, in parallel with the syntax–semantics–pragmatics distinction, and our treatment in section 2.

The leaves of the network are the word-inst and syn-rel nodes. For this portion of the graph, the belief network formalism requires us to provide P(denotation) $[\mathrm{P}(d)]$ and P(word|denotation) $[\mathrm{P}(w|d)]$. In principle, these could be computed from a labeled corpus. In practice, this is not necessary. Because the denotation variable is a function of the word variable, its probability is zero in the absence of the word. Accordingly, we are only concerned with $\mathrm{P}(d|w)$. We can specify these by ranking the various different denotation hypotheses relative to each other. Precise values of $\mathrm{P}(d)$ and $\mathrm{P}(w|d)$ are not important: it is enough to know the values of the products $\mathrm{P}(d)\mathrm{P}(w|d)[= \mathrm{P}(w,d)]$ *relative to other possible denotations for the same word*. The exact values of the products are not necessary since they will be normalized. This also enables us to do without $\mathrm{P}(w)$.

We do not have a good theory of the conditional probabilities for the syn-rel nodes, which are of the form P(syn-rel|relation in the real world). E.g., P((syn-rel to store2 go1))|(== (destination go1) store2)). We use a rough subjective estimate of how often a given construction will be used to convey some relation. E.g., the instrument relation is more often expressed by 'with' phrases than is the accompaniment relation (which is often expressed by conjunction).

Deeper in the graph, we need probabilities for propositions which represent explanatory hypotheses. For example, the hypothesis that Jack has a shopping plan which explains his going to the liquor-store. This is the node labeled (and (inst *liquor-shop4* liquor-shop) (== (go-step *liquor-shop4*) went2)) in Figure 1.

The task of quantifying this part of the diagrams may seem well-nigh impossible, because it appears to require that we specify a prior probability for this node. In fact, however, this is not the case. This part of the network is subject to the same analysis as that of the words and denotations. Again, this is because the explanatory nodes have meaning only in the presence of their explanands. So we can specify this part of our network by ranking the different explanations for each action relative to each other.

In our program, we have tried to tie these probabilities to the frequencies of events in the real world. We want to do this so that, e.g., when our program reads a sentence like "Mary went to the airport." it will favor explanations like 'Mary is going to meet someone on the plane,' or 'Mary will fly somewhere,' over 'Mary is going to hijack the plane.' This is justified because we're assuming the speaker is a transducer. E.g., in 'Tom Clancy world,' the probabilities would be radically different.[2] In theory, one could collect statistics for these networks by labeling a large corpus of stories, and counting nodes in the instantiated networks.

Finally, we need probabilities of the form P((== thing1 thing2)|(inst thing1 type1), (inst thing2 type1)). These are needed for coreference and for linking together explanations for different objects (we must recognize that the shopping plan that explains the presence of a liquor-store in a story is the same shopping plan that explains the presence of a bottle of bourbon). These are difficult to assess. We do not have time to discuss the problem fully here, we do so in an earlier paper Charniak and Goldman (1990). In brief, these probabilities must be sensitive to the number of different objects of each type which are likely to appear in a given story. For example, in simple stories like those Wimp3 reads,

$$\begin{aligned}&P((== \text{person1 person2})|(\text{inst person1 person}), (\text{inst person2 person}))\\ &\quad < P((== \text{airport1 airport2})|(\text{inst airport1 airport}), (\text{inst airport2 airport}))\end{aligned}$$

because it is very likely that there will be more than one person discussed in a single story, but unlikely to be more than one airport. We hope that discourse theory (e.g., Grosz and Sidner (1986), Webber (1987)) will shed some light on these quantities, and that notions like focus can be used as conditioning events.

5 The Program

In the preceding, we have developed our probabilistic model of stories. Now we will show how we use this model to do story understanding. Because the network models we have outlined here are impractically large, we proceed by incrementally constructing and evaluating relevant portions of the full network. These partial networks are constructed by network-building rules which are similar to forward-chaining rules. We have experimented with evaluating them using a number of different belief network algorithms.

Wimp3 works as follows: an all-paths parser reads the English-language input, one word at a time, and yields statements about the input. These statements are given to the network-construction component. This component extends the belief network corresponding to the input. Then the network evaluation component computes the posterior distribution of the network. If any hypotheses seem particularly good or bad, they may be accepted or rejected categorically. After this, if necessary, the network may be further extended, and re-evaluated. At this point,

[2]We are grateful to Jack Breese for pointing out to us the existence of this alternate sample space.

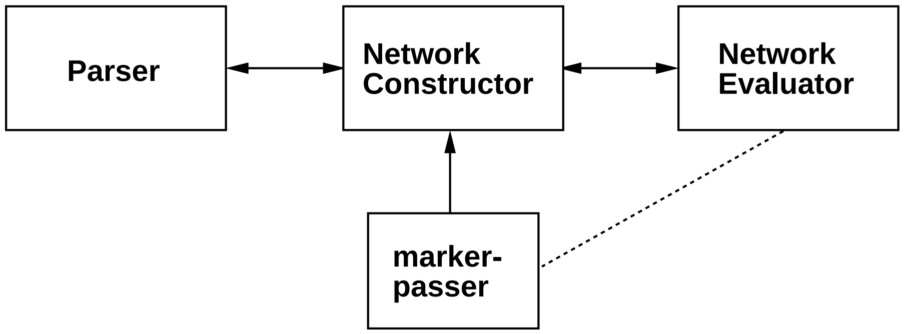

Figure 2: The architecture of Wimp3.

```
(→(word-inst ?i ?word) :label ?A
   (→←(word-sense ?word ?frame ?PROB)
        (inst ?i ?frame) :label ?C)
   :prob ((?C ⇒ ?A) ((t | t = ?PROB) (t | f = prior)) ))
```

Figure 3: One of the rules used to handle word-sense relationships.

control is returned to the parser, which reads the next word. A picture of the architecture of this system is given as Figure 2.

We have developed network building rules for constructing belief networks which correspond to story understanding problems. These rules operate on Wimp's deductive database. Statements in this database are also random variables in a belief network. The network construction rules are similar to conventional forward-chaining or production rules. They are augmented in two ways.

1. It is possible to add hyperedges between statements mentioned in the rules. These hyperedges are similar to the justifications in a data-dependency system (see Chapter 7 of Charniak and McDermott (1985) for an introduction to truth maintenance systems and data dependencies). They differ in that rules are not restricted to adding a single hyperedge from the antecedent nodes to the consequent. An arbitrary number of edges may be added, and the edges may point in any direction (as long as the resulting network is still acyclic).
2. Network-building rules contain information used to set the conditional probability matrices of the nodes in the network.

A sample network-building rule is given in Figure 3. This rule handles part of the problem of finding an appropriate referent for a word (the rules for plan-recognition are similar, but more complex). The meaning of the rule is:

> If a node is added to the network describing a new word-token (line 1 of the rule), and if there is a statement in the database specifying that one of the senses of this word is ?frame (line 2), add a node for an instance of the type ?frame (line 3). Draw an arc between this newly-added node (?C), and the word-inst node (?A) (line 4). Construct the conditional probability matrix for the word-inst node using information from the word-sense statement (?PROB) (line 4).

For example, if we were to learn of a use of the word 'bank,' (say bank1) and knew of two meanings: financial institution and river bank, this rule would tell us to add two nodes to our network: a node specifying that bank1 referred to a financial institution, and a node specifying that bank1 referred to a river bank. These two nodes would have arcs pointing to the node for (word-inst bank1 bank). The conditional probability matrix for the latter would be drawn up to reflect that the two possible senses of the word are related as exclusive causes. I.e., the probability that the word will be used given that one wishes to describe a financial institution or a river bank is some value determined by the word-sense rule; the probability that the word is used given one wishes to express both senses simultaneously is zero, and the probability that the word will be used given neither of these senses is meant is some default value.

We have similar rules for syntactic relations, reference resolution, etc. Initially we used this kind of 'forward-chaining' for plan-recognition as well. However, this led to an explosive growth in the belief networks. The reason for this is simple: consider how many possible reasons there are for 'going,' for example. Worse yet, many of these plans which could explain going can themselves play roles in more complex plans. To provide more direction to the way the networks are expanded, another researcher at Brown, Glenn Carroll, has developed a marker-passing search algorithm. For a discussion of the marker-passer, see Carroll and Charniak (1989). This marker-passer makes use of the probabilities in the network, hence the dashed line connecting these two components in Figure2.

See Goldman and Charniak (1991) for more details of the network building rules.

The posterior distribution for the belief network constitutes the interpretation of the input text. To see why this is the case, refer to the sample belief network given as Figure 1. What we want to assess is the probability that Jack is going to buy liquor, given that we have been told "Jack went to the liquor-store." We can read this information off the network, once we have evaluated it so that it reflects the posterior distribution – the probability of each node given the evidence in the story. It should be noted that our representation does not tie us to the idea of solution by computing posterior distributions: we could compute a maximum a posteriori instantiation of the networks to get a best global interpretation. Indeed, Shimony and Charniak (1990) have done some preliminary work in this direction. We do not do this because we want intermediate results, and because these algorithms do not provide better performance in practice.

We have been experimenting with a number of different algorithms for computing the posterior distribution of these networks. To do so, we have been using the IDEAL system (described in Srinivas and Breese (1989)), an environment in which one can define a belief net or influence diagram and apply to it a number of different evaluation algorithms. We find that a variant of the algorithm of Lauritzen and Spiegelhalter(1988) developed by Jensen (1989), gives the best results on networks like ours. Performance of this algorithm is improved by a clustering heuristic due to Kjærulff(1990). Our cutsets grow too large for conditioning (this algorithm is given in Pearl (1988)) and until now, we have not had a simulation-based approach that worked well on networks which contain extreme probabilities (Chin and Cooper (1987) discusses this problem). However, there has been continuous progress in belief network evaluation algorithms, and we are still experimenting with new algorithms.

6 Experiments with Wimp3

Wimp3 has been fully implemented. It is written in Common Lisp, and runs on Symbolics Lisp Machines and Sun SPARCstations. We have developed a small test corpus to test its abilities, focusing on problems which have been difficult for earlier approaches. It can parse and 'understand' 25 stories which describe everyday occurrences. It correctly parses and classifies 18 of 25 'test set' stories synonymous to these 25 stories, and 24 of 25 after small-scale augmentation of its knowledge base and lexicon.

The test and training set story pairs were developed by the second author, partitioned randomly, and kept hidden from the first author during his development and debugging of Wimp3.

The easiest story is

> Bill drank a milkshake with a straw.

To understand this story it is necessary to determine from context that the word 'straw' refers to a 'drink-straw,' rather than 'hay-straw,' and that the straw is the instrument, rather than the co-agent of the drinking. One of the most difficult stories is

> Bill took a bus to a restaurant. He drank a milkshake. He pointed a gun at the owner. He got some money from him.

which presents intertwined problems of plan-recognition and pronoun reference. In particular, it is necessary to recognize Bill's plan correctly in order to realize that 'he' in the last sentence refers to Bill, and 'him' to the owner.

Appendix A contains the 25 training set stories, and Appendix B contains a sample query, used to judge Wimp3's 'comprehension' of one of these stories. Processing time for the stories in this corpus ranges from c. 15 seconds to c. 10 minutes, on a Sun SPARCstation 1 with 135M swap, 16M memory, running Sun Common Lisp, Version 4.0, code compiled but not optimized.

We developed a simple testing method to avoid some problems of wishful interpretation of our experimental results. It has been customary to demonstrate natural language understanding programs by giving a selection of texts which the program is claimed to be able to interpret. Such claims are very difficult to evaluate, and leave open the possibility that the demonstrated programs are brittle and cannot handle even slight variations from the example texts.

Our solution is a single-blind test. During the development of the Wimp3 system, the second author developed a set of 25 pairs of synonymous stories whose syntax and vocabulary were within the capabilities of Wimp3's parser and lexicon. One of each pair was randomly assigned to the test set, and the other to the training set. The first author then worked on Wimp3 until it was capable of understanding the 25 stories of the training set.

To verify that Wimp3 was able to process stories other than the ones in the training set, we developed an alignment task. The stories in the test and training sets were to be synonymous with respect to a categorizing function. This categorizing function takes the internal representation derived in the course of Wimp3's processing of the story, and yields the following information: a count of the number of top-level plans in the story and, for each top-level plan, the most specific name for that plan, the agent of that plan, and the 'generalized patient' of that plan. For example, the simple story "Bill drank a milkshake with a straw." is categorized as: `(1 (STRAW-DRINK BILL- MILK-SHAKE))` — there is one top-level plan, a straw-drinking, performed by a boy named Bill, and a milk-shake was the patient of this drinking. The categories of each of the training set stories are given as Appendix C.

As mentioned above, Wimp3 was able to correctly align 18 of the 25 test set stories initially. With some minor alterations to its knowledge base and lexicon, it was able to correctly align 24 of 25. None of the program code needed to be changed to get this improvement, rather Wimp3's database needed to be augmented so that it could handle some unexpected case relations and some additional words.

While the 'single-blind' test briefly discussed here may not seem remarkable, it is, to the best of our knowledge, without precedent in story-understanding programs. No such test appears in, e.g., Norvig (1987), Cullingford (1978), Wilensky (1983), Wong (1981). For more on the difficulties of evaluating natural language systems, see Palmer and Finin (1990). The experiments are further discussed in the first author's thesis, Goldman (1990).

7 Summary

The contributions of this work are three-fold. First of all, we have constructed a probabilistic model of story comprehension. This makes it possible to address the problem of text understanding within axiomatic probability theory. Second, we have developed a way of constructing and evaluating probabilistic models on an as-needed basis. This makes it possible to apply probability theory to problems for which a precompiled model is either not available, or impractically large. Finally, we have introduced a single-blind testing methodology for text understanding programs. While weaker than standard testing methodologies, it is suitable for this still very experimental field, and for programs developed by small research groups.

A Stories in training set

1. "Jack went to the supermarket. He found some milk on the shelf. He paid for it."
2. "Bill went to the supermarket. He paid for some milk."
3. "Jack gave the busdriver a token. He got off at the supermarket."
4. "Jack got off the bus at the liquor-store. He pointed a gun at the owner."
5. "Jack went to the liquor-store. He found some bourbon on the shelf."
6. "Bill went to the liquor-store. He pointed a gun at the owner."
7. "Bill gave the busdriver a token."
8. "Fred robbed the liquor-store. Fred pointed a gun at the owner."
9. "Bill got a gun. He went to the supermarket."
10. "Fred went to the supermarket. He pointed a gun at the owner. He packed his bag. He went to the airport."
11. "Jack took the bus to the airport. He bought a ticket."
12. "Bill packed a suitcase. He went to the airport."
13. "Jack got on a bus. He got off at the park. Jack went to the supermarket."
14. "Jack gave the busdriver a token. He got off at the park. He went to the airport. He got on a plane."
15. "Fred sat down on the bus. He went to the supermarket."
16. "Jack went to a restaurant. He got a milkshake."
17. "Bill drank a milkshake with a straw."
18. "Fred got off the bus at a restaurant. He got a milkshake."
19. "Janet put a straw in a milkshake."
20. "Bill got on a bus. He got off at a restaurant. He drank a milkshake with a straw."
21. "Bill took a bus to a restaurant. He drank a milkshake. He pointed a gun at the owner. He got some money from him."
22. "Fred gave the busdriver a token. He got off the bus at the park. He went to a restaurant. He got some money from the owner."

23. "Jack took a taxi to the park."
24. "Bill took a taxi."
25. "Fred took a taxi to the bus-station. He got on a bus."

B Understanding stories

In order to decide when Wimp3 could be held to 'understand' the stories in the training set, we developed a set of database queries. Wimp3 was held to understand a story when it could successfully respond to the corresponding query. The query corresponding to the story

> Jack went to the supermarket. He found some milk on the shelf. He paid for it.

is

```
(and (inst ?jack jack-)
     (inst ?sming superming)
     (inst ?went go-)
     (inst ?find locate-)
     (inst ?pay pay-)
     (inst ?milk milk-)
     (== '(agent ?sming) ?jack)
     (== '(go-stp ?sming) ?went)
     (== '(pay-stp ?sming) ?pay)
     (== '(bought ?sming) ?milk)
     (== '(loc-stp ?sming) ?find))
```

An approximate English gloss is as follows: There appear in the story a person named Jack, a supermarket-shopping plan (`superming`), a going event, a locating event, a paying event, and some milk.

The actions mentioned in the story — the going, the locating and the paying, all play roles in the supermarket-shopping plan. Jack is the agent of the plan, and the milk is the object purchased in the plan.

Note that in order to make the last inference, Wimp3 must recognize that the pronoun 'it' refers to the milk, and not to the shelf, which is an equally good candidate at first glance. It makes this decision based on its knowledge about supermarket-shopping.

C Categorizing the stories

1. `(1 (SUPERMING JACK- MILK-))`
2. `(1 (SUPERMING BILL- MILK-))`
3. `(1 (SUPERMING JACK- FOOD-))`
4. `(1 (ROB- JACK- LIQUOR-STORE-))`
5. `(1 (LIQUOR-SHOP JACK- BOURBON-))`
6. `(1 (ROB- BILL- LIQUOR-STORE-))`
7. `(1 (BUS-TRIP BILL- BUS-))`

```
 8. (1 (ROB- FRED- LIQUOR-STORE-))

 9. (1 (ROB- BILL- SUPER-M))

10. (1 (ROB- FRED- SUPER-M))

11. (1 (AIR-TRIP JACK- AIRPLANE-))

12. (1 (AIR-TRIP BILL- AIRPLANE-))

13. (2 (SUPERMING JACK- FOOD-) (PARK-TRIP JACK- NIL))

14. (2 (AIR-TRIP JACK- AIRPLANE-) (PARK-TRIP JACK- NIL))

15. (1 (SUPERMING FRED- FOOD-))

16. (1 (EAT-OUT JACK- MILK-SHAKE))

17. (1 (STRAW-DRINK BILL- MILK-SHAKE))

18. (1 (EAT-OUT FRED- MILK-SHAKE))

19. (1 (STRAW-DRINK JANET- MILK-SHAKE))

20. (1 (EAT-OUT BILL- MILK-SHAKE))

21. (2 (EAT-OUT BILL- MILK-SHAKE) (ROB- BILL- RESTAURANT-))

22. (2 (ROB- FRED- RESTAURANT-) (PARK-TRIP FRED- NIL))

23. (1 (PARK-TRIP JACK- NIL))

24. (1 (TAXI-TRIP BILL- TAXI-))

25. (1 (BUS-TRIP FRED- BUS-))
```

References

Carroll and Charniak (1989) Glenn Carroll and Eugene Charniak. Finding Plans with a Marker-Passer. In *Proceedings of the Plan-Recognition Workshop*, 1989.

Charniak and Goldman (1988) Eugene Charniak and Robert P. Goldman. A Logic for semantic interpretation. In *Proceedings of the Annual Meeting of the ACL*, pages 87–94, 1988.

Charniak and Goldman (1989) Eugene Charniak and Robert P. Goldman. A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding. In *IJCAI-89*, pages 1074–1079, KAUFMANN-ADDRESS, 1989. KAUFMANN.

Charniak and Goldman (1990) Eugene Charniak and Robert P. Goldman. Plan Recognition in Stories and in Life. In M. Henrion, R.D. Schachter, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 343–351. ELSEVIER, ELSEVIER-ADDRESS, 1990.

Charniak and Goldman (1991) Eugene Charniak and Robert P. Goldman. Probabilistic Abduction for Plan-recognition. In Thomas L. Dean and Kathleen McKeown, editors, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 160–165, Menlo Park, CA, 1991. AAAI Press.

Charniak and McDermott (1985) Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, Reading, MA, 1985.

Charniak (1986) Eugene Charniak. A Neat theory of marker passing. In *AAAI-86*, pages 584–589, 1986.

Chin and Cooper (1987) Homer L. Chin and Gregory F. Cooper. Stochastic simulation of Bayesian belief networks. In Laveen F. Kanal, John F. Lemmer, and T. S. Levitt, editors, *Proceedings of the Workshop on Uncertainty and Probability in Artificial Intelligence*, pages 106–113, 1987.

Cooper (1987) Gregory F. Cooper. Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford University, 1987.

Cox (1946) R.T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14:1–13, 1946.

Cullingford (1978) Richard E. Cullingford. *Script Application: Computer Understanding of Newspaper Stories.* PhD thesis, Yale University Department of Computer Science, 1978.

D'Ambrosio (1988a) Bruce D'Ambrosio. A Hybrid approach to reasoning under uncertainty. *International Journal of Approximate Reasoning*, 2:29–45, 1988.

D'Ambrosio (1988b) Bruce D'Ambrosio. Process, structure and modularity in reasoning with uncertainty. In *The Fourth Workshop on Uncertainty in Artificial Intelligence*, pages 64–72, 1988.

deKleer (1986) Johan deKleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

Goldman and Charniak (1991) Robert P. Goldman and Eugene Charniak. Dynamic Construction of Belief Networks. In P.P. Bonissone, M. Henrion, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 6, pages 171–184. Elsevier Science Publishing Co.,Inc., 1991.

Goldman (1990) Robert P. Goldman. A Probabilistic Approach to Language Understanding. Technical Report CS-90-34, Computer Science Department, Brown University, 1990.

Grosz and Sidner (1986) Barbara J. Grosz and Candace Sidner. Attention, Intention and the Structure of Discourse. *CL*, 12, 1986.

Hobbs and Martin (1987) Jerry R. Hobbs and Paul Martin. Local Pragmatics. In *IJCAI-87*, pages 520–523, 1987.

Hobbs *et al.* (1988) Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as Abduction. In *ACL-88*, pages 95–103, 1988.

Jensen (1989) Finn V. Jensen. Bayesian Updating in Recursive Graphical Models by Local Computations. Technical Report R 89-15, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, Aalborg, DENMARK, 1989.

Kjærulff (1990) Uffe Kjærulff. Triangulations of graphs – algorithms giving small total clique size. Technical Report R 90-09, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, Aalborg, DENMARK, 1990.

Laskey and Lehner (1988) Kathryn B. Laskey and Paul E. Lehner. Belief Maintenance: An integrated approach to uncertainty management. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 210–214, 1988.

Lauritzen and Spiegelhalter (1988) S.L. Lauritzen and David J. Spiegelhalter. Local Computations with probabilities on Graphical Structures and their Application to Expert Systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

Nilsson (1986) Nils Nilsson. Probabilistic Logic. *Artificial Intelligence*, 28:71–88, 1986.

Norvig (1987) Peter Norvig. Inference in Text Understanding. In *AAAI-87*, pages 561–565, 1987.

Palmer and Finin (1990) Martha Palmer and Tim Finin. Workshop on the Evaluation of Natural Language Processing Systems. *CL*, 16(3):175–181, September 1990.

Pearl (1988) Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, Waltham, MA, 1988.

Shimony and Charniak (1990) Solomon Shimony and Eugene Charniak. A New Algorithm for Finding MAP Assignments to Belief Networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 98–103. General Electric Corporation, 1990.

Srinivas and Breese (1989) Sampath Srinivas and Jack Breese. IDEAL: Influence Diagram Evaluation and Analysis in Lisp. Technical report, Rockwell International Science Center, Palo Alto, CA, May 1989.

Webber (1987) Bonnie Lynn Webber. The interpretation of tense in discourse. In *ACL-87*, pages 147–154, 1987.

Wilensky (1983) Robert Wilensky. *Planning and Understanding.* Addison-Wesley, AW-ADDRESS, 1983.

Wong (1981) Douglas Wong. Language comprehension in a problem solver. In *IJCAI-81*, pages 7–12, 1981.