Deep Domain Models for Discourse Analysis

Leo Joskowicz  Tomasz Ksiażyk  Ralph Grishman

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street, New York, NY 10012

Abstract
This paper describes the use of an equipment simulation model to find causal relations between facts in short equipment failure messages. We argue that a simulation model is necessary for identifying implicit causal relations that are essential to fully understand the message. We present a computational model that uses a simulation model of the equipment, together with a set of hypothesis rules, to establish the relationships between facts.

Introduction
It is widely accepted among researchers in linguistics and natural language processing that detailed knowledge about the domain of discourse is necessary for a complete understanding of utterances in relation to the domain. Language processing problems such as reference resolution, disambiguation, and identification of causal dependencies can only be resolved to their full extent by having a model of the domain of discourse. The purpose of this model is to organize the relevant domain knowledge for use in syntactic, semantic, and discourse analysis. The robustness of any natural language system is heavily dependent on the extent and organization of its domain knowledge.

In this paper, we address the problem of discourse analysis, and in particular finding causal relations between facts mentioned in messages, using a detailed domain model. This work is part of the PROTEUS (PROtotype TExt Understanding System) project, whose objective is to understand short narrative messages about equipment installed in Navy ships. Casualty Reports (CASREPs) describe failures of this equipment, together with maintenance actions performed by the crew on board. To capture the domain knowledge, we built a model of the equipment installed on the ship (initially, the starting air system) and demonstrated its use in several aspects of language understanding. Previous papers ([Gris88] and [Ksia89]) describe this model-based approach and the use of the model in the interpretation of noun phrases ([Ksia87]). This paper shows how this model is used to find causal relations between facts mentioned in the messages.

Following Schank [Scha77] and Wilensky [Wile78], [Wile81], we view language understanding as an explanation driven process. A fact in a text is said to be understood only when a plausible explanation for it is found, which means, in our particular domain of equipment failures, that all its causal relations to other facts in the text are found. Causal analysis is important for many understanding tasks, such as explanation, summarization, and temporal analysis. First, causal chains constitute a possible explanation for the facts described in a message since its links provide a justification for the described facts. In our failure messages, the causal chain provides an explanation of the sequence of facts that lead to a malfunction. Furthermore, the original circumstances of the malfunction can be traced back to the first set of facts in the chain. The last set of facts constitutes the ultimate consequences of this malfunction. Using both sets, a summary of the message can be produced by stating only the original failure and the ultimate consequences. The causal chain can also be viewed as the validation of the coherence of the message since it is always consistent with the domain model; it can therefore be used to select preferred readings of an ambiguous message. Finally, the causal chain establishes temporal relations between facts, since it describes the order in which the facts occurred.

The main problem when looking for causal relations is that these relations are not always explicitly stated, nor can they be inferred from the syntactic and semantic information appearing in the text. Consider for example the following (unedited) casualty report:

CH2715-1/89/0000/0195$01.00 © 1989 IEEE
While diesel was operating with SAC disengaged, the SAC lube oil alarm sounded. Believe the coupling from diesel to SAC lube oil pump to be sheared. Pump will not turn when engine jacks over.4

The message states that the possible cause of the alarm is the shearing of the coupling between the motor and the pump. Because of this shearing, the rotation of the motor cannot be transmitted to the oil pump, which produces oil pressure for lubrication. Since the pump does not turn, no pressure is generated, and thus the low pressure alarm sounded. Note that there is a causal relation between the fact that the oil pump does not rotate and the sounding of the alarm, although this relation is not stated in the message. This relation can only be determined by knowing the function of the oil pressure alarm and the oil pump, as well as their operational specifications.

The organization of domain knowledge and its use in language understanding has been the topic of much research for over a decade. In the stories of personal encounters discussed by Schank [Scha77] and Wilensky [Wile78, Wile81], the characters of the story are assumed to have a particular goal and a plan to achieve it. The knowledge about beliefs and actions of people is encoded as a set of rules. Goals and plans to achieve them are specified as frames or scripts. Novel situations not included in the original plan may arise, and a means-ends analysis is performed to find the causal chain of facts that explains the new situation. More recently, Norvig [Norv83], [Norv87] has proposed a unified scheme for doing both parsing and inferencing based on frame-like objects that represent physical entities and inference rules. Instead of having a separate knowledge structure for scripts and plans, he introduces six general classes of inferences that are not dependent on the individual knowledge structures, but rely on patterns of connectivity between concepts. Marker-passing techniques such as the one described by Martin and Riesbeck in [Mart86] and Charniak in [Char86] are based on indexed memory structures. Inferencing is done by a memory search that attempts to find paths between memory structures. In our domain of discourse, none of these representation schemes is fully adequate. Representing the knowledge about the functioning of the equipment as a collection of plans or scripts would require an exhaustive list of all equipment failures, which is clearly infeasible for large and complex machines. Using a set of rules to describe the functioning of the equipment poses the same problems of consistency and control encountered in rule-based expert systems. As for memory structures, they are adequate only when the domain knowledge is easily represented declaratively. The operation of the equipment in our case is best described procedurally.

As an alternative representation, we propose to use a simulation model of the equipment. The novelty of our approach lies in the use of a simulation model to embody the knowledge required for natural language analysis. This model captures the causal, structural and functional relations between the different components and subsystems of the equipment. The simulation model is complemented by a set of hypothesis rules that are used to generate hypotheses about the possible causes of failures. These hypothetical causes do not appear in the original message, but are necessary to reconstruct the causal chain of facts.

Researchers at the Naval Research Laboratory have recently described an alternative approach to analyzing these messages which relies on a static equipment model and on rules to propagate failures through this model [Wauc88]. In many cases the net result is the same as using a simulation model. We believe that in some cases, however, the static model would not capture instances of failure which can be identified using the simulation.

The PROTEUS System

The PROTEUS message processing system has three major stages: syntactic analysis, semantic analysis, and discourse analysis.

The syntactic analyzer performs parsing and syntactic regularization. The parser uses a chart parsing algorithm and an augmented context-free grammar. This grammar includes procedural restrictions which enforce various grammatical constraints. It also includes regularization rules, associated with each production, which operate compositionally to produce a regularized syntactic form.

The semantic analyzer consists of a clause analyzer and a noun phrase analyzer. The clause analyzer maps each clause (and nominalization) into a proposition involving a domain-specific predicate. The noun phrase analyzer has the task of identifying the referent or possible referents (in the equipment model) of the noun phrases in the message. This is a difficult task because of the prevalence in these messages (and similar technical texts) of lengthy compound nouns. The analysis procedure relies heavily on the information in the equipment model, as we have described in [Ksio87].

The final stage is the discourse analyzer, which aims to identify the causal and temporal relations among the events in the message. This stage begins by mapping the propositions produced by semantic analysis into the elementary facts which will be used by causal analysis. In some cases, the mapping is one-to-one; in others, a single proposition (for example, describing a change of state) is decomposed.
into several facts (describing the initial state, the transition state, and the final state). This is followed by the causal (and temporal) analysis which is described in the later sections of this paper.

The Equipment Model

Structural and functional knowledge about the equipment has been captured by an equipment model. In this model, the equipment is represented as a recursive transition network in which the nodes are equipment units (like cooling system, lubrication oil pump, or relief valve) and the edges are conduits connecting these units. Conduits transmit media, such as gas, liquids, mechanical movement or electric current. Every edge is associated with the medium it transports. The recursive nature of the model implies that a node at one level of detail can be represented as a network at a lower level. A node at the deepest level is described by a function specifying the way this node changes the parameters of media flowing through it. These relations are given in qualitative terms only. For example, should we decide to represent an oil cooler as a basic node (i.e. one which has no underlying network) in our model, we would include in its description a function specifying that the temperature of oil flowing through the cooler changes to NORMAL when it is HIGH at input, or remains unchanged otherwise. The behavior of higher-level nodes is determined recursively.

Associated with each equipment unit, besides its behavior function (specified only if the unit is modelled by a basic node), is a set of attributes describing the different states in which the unit can be, the operational parameters related to it, pointers indicating its place in the model, etc. Using this information together with the behavior functions, the model is capable of carrying out simulations, i.e. propagating the consequences of events throughout the network. There are two classes of simulated events: (1) operator interventions such as turning on a control switch, and (2) unexpected disturbances such as seizure of a shaft. Simulation can be viewed as an event-driven process. A simulated scenario is an interleaved sequence of stable and unstable states of the model. Each event causes the model to enter an unstable state. Once all the consequences of the event have been propagated, the model returns to its stable state and is ready to process the next event. This mode of servicing events results in a non-real-time simulation. The dynamic aspects of the simulation can be shown using the graphical interface (icons representing rotary parts move, those depicting gases and liquids flow, etc.). Figure 1 shows the highest-level model of the ship's propulsion system. We refer to the equipment model with its simulation procedures as the simulation model (for more details, see [Ksie88]).

Figure 1: Model graph for the propulsion system

Finding Causal Relations

Two sets of facts mentioned in a message are causally related when the facts in one set are the consequence of the facts in the other. This causal relation might require an implicit sequence of facts, not originally mentioned in the message, to explain the relationship. Such sequences of facts form a causal chain, in which every node is the consequence of the previous one (except for the first node which is the initial cause of all subsequent facts). Unconnected facts never appear in the same chain since there is no causal link between them. In general, causal relations between facts in a message can be described by one or more causal chains (possibly with common links).

We have determined that the most effective way of finding causal links is to use a simulation method. The simulation method is adequate for finding causal links since it has the capability of propagating the consequences of a set of facts throughout the model. All these consequences are both logically and operationally sound since the simulation itself is sound. Nevertheless, as will be pointed out in the next section, there are cases when some additional mechanisms are needed to bind facts into causal chains.

The starting point for discourse analysis and causal chain construction is a set of propositions (predicates with arguments) extracted from the report. This set is produced by the semantic analyzer, in conjunction with the noun phrase analyzer. These propositions are then translated into elementary facts. To find the direct causal relations that will form the causal chain, we must first establish the sets of facts that will be tested pairwise to determine if a causal link exists between them. In principle, every pairwise combination of sets is a potential candidate for testing. This constitutes a large number of tests, although most of them
are not necessary to determine the coherence of the message. We can determine in advance, by the nature of the facts in the sets, whether a causal link can exist between them since certain classes of facts can never have a causal relation. For example, a fact describing the normal operation of a component cannot possibly be the cause of a fact describing the malfunction of another component. To capture this idea, we established a taxonomy of 4 classes of possible facts: (1) static states, e.g. broken gear tooth; (2) dynamic states, e.g. low oil pressure; (3) transition states, e.g. turbine changing its state from standstill to running; (4) actions, e.g. operator pushing a control button. Furthermore, each fact is classified as normal or abnormal. Using this taxonomy of fact categories and the information about their normality, we can discriminate between pairs of fact sets that require further testing by the simulation model. This additional test, in the form of a simulation query, will ultimately determine whether there is a causal link.

In order to determine whether two sets of facts are causally related, we send queries to the simulation model. There are two types of queries: dynamic simulation queries and static structural queries. Simulation queries are of the form MQ-Simulation($F_1, F_2, Context$), where $F_1$ and $F_2$ are two facts whose categories are compatible according to the taxonomy, and $Context$ is a set of facts describing the operational state of equipment (queries to the model are in the form of two facts and a context, rather than two sets of facts, for reasons of simulation efficiency). Upon receiving a simulation query, the simulation model sets the model to the facts from $Context$, and then tries to establish a causal link between $F_1$ and $F_2$ by simulating the consequences of $F_1$, and determining whether $F_2$ is included among them. If so, the result of the query is true, or false otherwise. Structural queries are used when it is necessary to determine whether two equipment units are related by a path. Structural queries are of the form: MQ-Path($N_1, N_2, Media$) where $N_1$ and $N_2$ are two equipment units, and $Media$ is any combination of rotation, oil and air. This query returns the path between the two nodes through the media indicated, or false if no such path exists. One static query is shown below:

\[
\text{if } \text{seized}(X) \text{ in the context of generator}(Y, \text{Rotation}), \\
\text{MQ-Path}(Y, Z, \text{Rotation}), \\
\text{MQ-Path}(Z, X, \text{Rotation}) \\
\text{then sheared}(Z)
\]

The interpretation of this rule is: “If an element $X$ is seized, then it is possible that an element $Z$, lying on a rotary path between the generator of rotation (a motor) $Y$ and the seized part $X$ is sheared.”

The algorithm for building causal chains starts by separating propositions that are considered as context (facts describing operational state) from those that describe abnormal facts. Then, using the taxonomy of fact categories and the explicit temporal relations in the text (conveyed by words like while, when etc.), a set of simulation queries consisting of two facts and a context is created. An ordering between queries is established, and the queries are then sent to the simulator; the results of these queries are recorded. If it is possible to build a causal chain relating all the facts (or those that can be related according to the taxonomy) with the results of the simulation queries, the algorithm stops. If, on the other hand, abnormal facts are left unrelated, the system tries to use hypothetical facts. Figure 2 illustrates the derivation of the causal chain for the message presented in the introduction. In this case no hypothetical facts were necessary.

In addition to providing a method for finding causal links, the simulation model has proved to be an effective tool for resolving semantic ambiguities. If a message has more than one possible interpretation, we can determine which interpretations are consistent by testing them against the simulation model. Interpretations that are consistent with the simulation model are preferred.

**Hypothetical Facts**

In the preceding section we have assumed that all the reported facts can be included into some causal chains. There are cases, however, where the simulation method alone, as described in the preceding section, is not sufficient to do this. Therefore, at the end of the above process, we check whether any of the reported facts are “dangling”, i.e. don’t belong to any of the causal chains established so far. If so, we try to hypothesize facts which would possibly enable us to include dangling facts into some causal chain. One such case involves the rule mentioned above which relates seizing to shearing. If both the seizing and shearing events are explicitly mentioned in the message, this rule will tie them together in a causal chain. Suppose, however, that only the seizing and an effect of the shearing are explicitly mentioned (e.g., “oil pump will not turn when engine jacks over”). In this case the two events will not be directly causally related. The system must use this rule as a hypothesis rule, and hypothesize that some part between the diesel engine (the generator of rotation) and the seized component has sheared. For each such part, it will then try to complete the causal chain by using the simulation model to relate the shearing to the effect explicitly described in the message.

As another illustration, consider the following message from our corpus:
Message: \((\text{fact1, fact2, fact3, fact4, fact5, fact6})\)

Abnormal Facts:
\(\text{fact1: sound(lube-oil-alarm)), while(fact4)}\)
\(\text{fact2: become(sheared(coupling)), } \emptyset\) ; alarm sounded
\(\text{fact3: not(rotate(lube-oil-pump)), when(fact5)}\) ; coupling sheared
\(\text{fact4: operate(diesel), with(fact6)}\) ; pump does not turn

Facts Describing Operational State:
\(\text{fact4: operate(diesel), with(fact6)}\) ; diesel was operating
\(\text{fact5: jack-over(diesel), } \emptyset\) ; engine jacks over
\(\text{fact6: disengaged(SAC), } \emptyset\) ; SAC Disengaged

Simulation Queries:
1. MQ-Simulation\((\text{fact1, fact2, \{fact4, fact6\}})\)  Answer: false
2. MQ-Simulation\((\text{fact2, fact1, \{fact4, fact6\}})\)  Answer: true
3. MQ-Simulation\((\text{fact2, fact3, \{fact5\}})\)  Answer: true
4. MQ-Simulation\((\text{fact3, fact2, \{fact5\}})\)  Answer: false

We are not testing the causal links between fact1 and fact3 because their contexts (fact4 and fact5) are incompatible: the diesel cannot operate and jack over at the same time; it can perform only one of these actions at a time.

Causal Relations:
From Query 1b:  \text{fact2 }\Rightarrow\text{fact1} \quad \text{in context } c1 = \{\text{fact5}\}
From Query 2a:  \text{fact2 }\Rightarrow\text{fact3} \quad \text{in context } c3 = \{\text{fact4, fact6}\}

Causal Chains:
1. \text{fact2 }\Rightarrow\text{fact1}
2. \text{fact2 }\Rightarrow\text{fact3}

Figure 2: An example of causal chain derivation

Oil pressure has been slowly decreasing. Failure occurred during engine start when oil pressure dropped below 60 psig. Investigation revealed excessive fine metal particles in oil.

The second sentence sets the context for the fact reported at the beginning of the message. In this case, we cannot establish a direct connection between the decrease of pressure and the finding of metal particles in oil. Both these facts are dangling ones. To make them into a causal chain, we must conjecture at least one fact not mentioned in the message, which would serve as a missing link between the two explicitly mentioned facts. To do this we use both general domain knowledge about technical equipment and specific knowledge about our model. The former tells us that metal particles in oil can be a symptom of the wearing or shearing of some rotating metal parts with which oil comes into contact. From specific knowledge retrievable from the model we can identify all such parts. Next, using a simulation procedure for each such part, we can establish which ones among them, when sheared, can cause the oil pressure to decrease. Such a relationship will be confirmed for those parts which constitute the transmission link between the diesel motor and the oil pump. Therefore, one conjecture could be that the coupling between the pump and the diesel has sheared. We would mark this fact as hypothetical, because although it explains the reported situation, it cannot be proven. What's more, in the particular situation described in the above message, this is not the only possible hypothesis consistent with our model. Between the oil pressure gauge and the oil pump is a filter. The metal particles in the oil could have clogged the filter to such an extent that it started to act as a pressure reducing element. This would provide an alternative causal connection between the facts.

To avoid the rapid proliferation of hypothetical facts, we do not allow the system to hypothesize facts which would be — even partially — based on other previously hypothesized facts.
Conclusion

The PROTEUS system has been substantially implemented and debugged and has been publicly demonstrated operating on a small set of actual CASREPs. The equipment model and its simulation has been fully implemented, as well as the noun phrase analyzer. The causal chain mechanism has been implemented to date only for causal analysis without hypothesis generation.

Adding a mechanism for hypothesizing plausible explanations for a set of observations, as described in the previous section, would enrich our language understander and incorporate many of the capabilities needed for equipment diagnosis. It would then be a natural and straightforward extension of our system to create an interactive diagnostic assistant, which could accept verbal descriptions of failures and suggest further tests to be performed. One could expect that the resulting system, when combined with the already implemented natural language and graphical interface modules, would be especially user friendly.

References


