Parsing and Case Analysis in TANKA

TERRY COPECK, SYLVAIN DELISLE, STAN SZPAKOWICZ

Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
{terry, sylvain, szpak}@csi.uottawa.ca

ABSTRACT

The TANKA project seeks to build a model of a technical domain by semi-automatically processing unedited English text that describes this domain. Each sentence is parsed and conceptual elements are extracted from the parse. Concepts are derived from the Case structure of a sentence, and added to a conceptual network that represents knowledge about the domain. The DIPETT parser has a particularly broad coverage of English syntax; its newest version can also process sentence fragments. The HAIKU subsystem is responsible for user-assisted semantic interpretation. It contains a Case Analyzer module that extracts phrases marking concepts from the parse and uses its past processing experience to derive the most likely Case realizations of each with almost no *a priori* semantic knowledge. The user must validate these selections. A key issue in our research is minimizing the number of interactions with the user by intelligently generating the alternatives offered.

BACKGROUND

This project is a long-term undertaking of the Knowledge Acquisition Lab. Previously we presented its overall design (Szpakowicz 1990), discussed elements of the Conceptual Knowledge Processor (Szpakowicz & Koperczak 1990; Yang & Szpakowicz 1990, 1991a, 1991b), and described the parser and Case Analyzer (Delisle 1990; Delisle & Szpakowicz 1991; Copeck *et al.* 1990). This paper updates and summarizes the last three publications. TANKA (Text ANalysis for Knowledge Acquisition) is implemented in Quintus Prolog on Sun workstations.

THE DIPETT PARSER

TANKA requires a broad-coverage parser because it uses a limited semantic model based on Case relations, and domain-specific knowledge is not available to it *a priori*. Without rich semantics, syntax is the only basis for inferring meaning. In TANKA, the broader the parser's coverage, the more accurate the ultimate knowledge representation can be. This is in opposition to approaches in which semantic knowledge is fed in beforehand, and syntax is limited to restricted patterns or even just keywords. Our approach lies at the other end of the spectrum: we are concerned with realistic large-scale texts and need realistic syntactic coverage. This enables HAIKU, the interactive semantic interpreter, to extract overt meaning from DIPETT's detailed parse trees, and helps organize interaction with the user.

DIPETT (Domain-Independent Parser for English Technical Texts) is a linguistic-theory-neutral parser with a broad surface-syntactic coverage of English. It handles most sentences in our unedited sample text, a guide to the fourth generation database language Quiz. DIPETT's coverage encompasses every fundamental syntactic structure in the language, including coordination, and most syntactic phenomena encountered in typical expository technical texts. The core of its grammar is based on general and NLP-oriented English grammars, in particular, Quirk *et al.* (1985) and Winograd (1983).

DIPETT's major components are a dictionary, a lexical analyzer, a syntactic analyzer, a memorizing device with a helper mechanism, plus its own trace mechanism. An input is usually given to the lexical analyzer and then to the syntactic analyzer; this makes for conceptually clear and easily implemented models. More than half of the parser's 5000 lines of code are DCG rules. A 15-word sentence can typically be processed in 15 to 20 seconds CPU on a Sun SparcStation. Novel features of DIPETT are a dynamic dictionary expansion facility, its memorizing device (well-formed substring table), a helper (error explanation mechanism), and an internal trace mechanism for debugging.

The parser's surface-syntactic *dictionary* contains most English function words. It includes a table that associates legal adverbial particles with verbs (this is used to disambiguate particles and prepositions). Another table contains word groups such as "as much as" or "even if" that usually play the same role as single function words. The dictionary will be expanded with semantic information when it is integrated with the Case Analyzer. The *lexical analyzer* builds a list of annotated words with the root form and the syntactic parameters. If the input contains a word for which the dictionary has no entry, this module allows the user to augment the dictionary dynamically. Such temporary additions are saved on a file for future permanent addition.

DIPETT's *grammar* recognizes the following major syntactic units: sentence (simple, complex and multiply-coordinated), question, verb phrase (simple and conjoined), verbal clause, complement, subordinate clause, adverbial clause, noun phrase (simple and conjoined) and their substantive forms, that-clause, relative clause, ing-clause, to-infinitive clause, whether-if clause, noun phrase post-modifier (e.g. appositive), prepositional phrase (simple and conjoined), noun pre- and post-modifier, determinative, adjectival phrase.

The purpose of the *memorizer* is to minimize the reparsing of syntactic substructures that are reconsidered on backtracking. The *helper* shows the user information that may help identify the reasons for an input's rejection. Both features can be switched on or off for the session. These two modules use *notes*—assertions that record essential syntactic information about major well-formed substrings that constitute the prepositional, noun and verb phrases. A note stores a substring, its type and its syntactic structure produced by the parser. Corresponding DCG rules contain Prolog assertions invoked if the user has activated the memorizer or the helper.

Testing and fine-tuning a complex parser can be difficult. Prolog debugging facilities are often cumbersome for logic grammars where it is only interesting to know what rule is being examined by the parser, for which part of the input string, and what has been successfully recognized. We have therefore implemented our own trace mechanism which employs trace instructions (activated by a flag) inserted in all rules related to prepositional, noun and verb phrases. The parser implementor can activate and control the trace mechanism through a menu interface.

Conjoined verb phrases and sentences are usually very expensive to parse. We have devised two look-ahead mechanisms to treat coordination efficiently. These mechanisms check the lexical categories of tokens ahead in the input string. The first looks for coordinated clauses, while the second checks inputs that are supposed to contain at least one verb (such as the to-infinitive clause). This information is used by the parser to identify potential joining-points for conjoined sentences and to avoid applying rules that cannot succeed. The parser also handles elided modals and auxiliaries in conjoined verb phrases. For example, "John *has printed* the letters *and read* the report" is analyzed as "[[John] [[has printed the letters] and [has printed the report]]]". Scoping of negation and adverbs in conjoined verbs is handled, too. For example "John *did not accidentally print and read* my personal messages" is analyzed as "[[John] [[did not accidentally print] and [did not accidentally read]] [my personal messages]]".

DIPETT does not have access to semantic knowledge, so prepositional phrase (PP) attachment must use syntax-based heuristics. Two examples: an 'of' PP is attached to the preceding noun by default; if a PP which is not an initial modifier occurs in a pre-verbal position, it is attached to the noun (whatever the preposition may be).

CURRENT WORK IN DIPETT

It is our experience that sooner or later an extra-grammatical or highly ambiguous input will engage the parser in an excessively lengthy computation. We must be able to deal with such extreme situations because our knowledge acquisition method requires finding, for any input, the first parse tree that is linguistically reasonable. The reshuffling of the tree's components is left to HAIKU. At present, we discontinue a parse operation that exceeds the time allowed for a single parse (specified by the user at the beginning of a session). Timing-out in this manner causes loss of information from a partially parsed sentence, but it is preferable to the user's waiting unrealistically long for the system's feedback. DIPETT also applies look-ahead and heuristics to fail unpromising partial parses quickly (e.g. it will not try verb phrase analysis if there is no verb). This helps produce the first reasonable parse tree as fast as possible.

The ultimate goal of the TANKA system is to process free-form technical texts. Texts often contain non-textual material such as tables or examples (e.g. data, programs, results). We assume all non-textual elements have been removed from our source texts, but each removal leaves a "hole" behind. Most holes are located between sentences and do not affect the structure of the text, but some cause fragments to appear in the text. Fragments are valid sub-structures of English sentences, such as "For example" in "`For example, > SORT ON DATEJOINED D.`"

DIPETT can parse such fragments.

Three areas of grammar are currently under active development in DIPETT:

1) **References:** the parser will be capable of resolving simple references, in particular anaphora, on syntactic grounds alone (we mean references whose resolution requires little or no semantic knowledge)—see Hobbs (1978).

2) **Topic and focus:** the parser will maintain some knowledge about topic and focus. As a first indication, a text's title should tell us about its topic while the current input indicates focus; this could benefit the Conceptual Knowledge Processor in TANKA by tentatively relating the topic to a cluster in the conceptual network.

3) **Paragraph parsing:** the parser's default mode of operation is one sentence at a time. Parsing longer inputs, a number of consecutive sentences or even paragraphs, means much more elaborate processing than parsing single sentences. Nothing is gained by simply finding a sequence of parse trees—one for each sentence, in order; see Jensen (1989) for a similar statement. We have plans for a more intelligent type of parsing that would be able to summarize the contents of these longer inputs by highlighting the main conceptual elements more closely related to the current topic (see Zadrozny & Jensen (1991) for a theory of the paragraph). Topic and focus information will probably help here.

CLASS	CASE	ABBR.
PARTICIPANT		
1	Agent	AGT
2	Beneficiary	BENF
3	Experiencer	EXPR
4	Instrument	INST
5	Object	OBJ
6	Recipient	RECP
SPACE		
7	Direction	DIR
8	Location_at	LAT
9	Location_from	LFRM
10	Location_to	LTO
11	Location_through	LTRU
12	Orientation	ORNT
TIME		
13	Frequency	FREQ
14	Time_at	TAT
15	Time_from	TFRM
16	Time_to	TTO
17	Time_through	TTRU
CAUSALITY		
18	Cause	CAUS
19	Contradiction	CNTR
20	Effect	EFF
21	Purpose	PURP
QUALITY		
22	Accompaniment	ACMP
23	Content	CONT
24	Manner	MANR
25	Material	MATR
26	Measure	MEAS
27	Order	ORD
28	Value	VAL

Figure 1. Cases Used in TANKA

CASE ANALYSIS WITH LEARNING

In TANKA, knowledge is expressed in terms of entities engaged in acts that serve to link them into a graph; see Sowa (1984) for a general discussion of this type of representation. This graph is the conceptual network that TANKA will gradually build for a technical text. It is constructed from Case frames of verbs recognized in the sentence. We have put together a set of Cases suitable for our class of domains; it is inspired by lists found in Fillmore (1968), Bruce (1975), Grimes (1975), Cook (1979), Larson (1984) and Sparck Jones & Boguraev (1987). This set (Figure 1) is not entirely settled; we continue to review the work of other authors and we are currently testing our selections against those Somers (1987) presents in his Case grid.

Case Analysis (CA) extracts the acts and Case constellations around them from the structure produced by the parser on a sentence-by-sentence basis. Only one parse is used, but the system will allow the user to override all its suggestions. Subsequent processing can adjust the understanding of a sentence enough to encompass most alternative parses and only fails to cover situations when a word can be legitimately parsed twice as different parts of speech. Items extracted from the parse are mapped quite directly into Case structures. A verb denotes an act. A Case is marked most often by a preposition or an adverb, and a noun or nominalization (marked by a preposition) serves as a Case object. Initial processing of a parse tree identifies elements of interest; others such as noun modifiers are not used by CA but are kept in the representation for the Conceptual Knowledge Processor. Two questions must then be answered for each Case-Marker in TANKA: to which verb does it attach, and which Case does it realize?

The HAIKU module does not attempt to answer these questions itself, at least not in a definitive way. It asks the user to answer by selecting among alternatives in a list, which may include syntactic elements from the original sentence copied exactly, illustrative phrases and sentences, and possibly short descriptions of the meaning of Cases. Our goal is to minimize the number of interactions the user must engage in to give the right answer. This can be done by letting all answers be specified in one interaction, and that in turn is possible if HAIKU proposes correct Case-Marker attachments and semantics at the outset. In practice a minimum of two interactions per complex sentence appear to be necessary, one to correctly link Case Markers to verbs and a second to validate Case Marker semantics for each verb. Our work on HAIKU thus concentrates on ensuring it produces the correct configuration, preferably on the first interaction.

Attachment of Case-Markers to verbs is inferred solely from the parse structure. Semantics could help were they known in advance (a verb has only one Case of a given type) but semantic inference is also aided by knowledge of syntax and something must come first. Once the user has endorsed an assignment of Case-Markers to verbs, each clause in the nested structure of coordinated and subordinated clauses received from the parser is considered in isolation. Because the pattern of Case-Markers (CMP) associated with a given verb is known when the second user interaction is undertaken, HAIKU can check a dictionary of these patterns to see if this particular one has been encountered earlier with any verb. If it has, the matching CMPs will be ordered according to a closeness metric discussed below. Otherwise HAIKU will use this closeness metric to search its CMP dictionary for the pattern that most nearly resembles the input CMP. This pattern may lack certain Case-Markers, have extra ones, or not match on both grounds. However a candidate pattern will *always* be found, it will be the *best possible*, and HAIKU can provide *additional,* next-best patterns should the first be deemed unsatisfactory.

For example, the sentence "`The parcel was moved from the house to the car`" has the CMP SUBJ-OBJ-FROM-TO (where SUBJ is *nil* here), associated with the verb *move*. A dictionary of CMPs is searched to see if this pattern has previously been associated with *move*. If not, the analyzer will look at the entry for *move*. Suppose it finds {SUBJ-OBJ, SUBJ-OBJ-WITH, SUBJ-FROM-AT}. It could try to add Case alternatives realized by FROM and TO to the SUBJ-OBJ pattern, or it might return to the CMP dictionary and seek an instance of SUBJ-OBJ-FROM-TO associated with a different verb. Eventually the algorithm selects the CMP closest to the input pattern. Closeness is a metric based on factors such as the number, types and agreement of CMs in each pattern and the verb associated with each (Copeck *et al.* 1992). It may be extended to use a very simple noun semantics for Case Objects or counts of the frequency of previous selection.

The HAIKU dictionaries—an incrementally growing store of verb-CMP associations, Case Patterns and examples—are searched for sentences that exemplify the Case Patterns associated with the CMPs. For example, if SUBJ-OBJ-FROM-TO is associated with *take*, the sentence might be "`Our guests took the train from Montreal to Ottawa`". The sentence is shown to the user, who can accept the underlying Case Pattern as correct, edit it by invoking a mode whereby a new Case is associated with a selected Case-Marker, or ask to see the next sentence in the list. The decision to view another sentence will probably be dictated by the number of changes required in the pattern illustrated by the current example. The user's selections are used to update the HAIKU dictionaries and to freeze the sense and structure of the conceptual fragment expressed by the clause which the pattern represents: the system has learned a new pattern of Case Markers, associated them with a particular verb, and recorded the meaning they convey in this instance. The resulting conceptual fragment is then passed on to the Conceptual Knowledge Processor to be integrated into the main Conceptual Network.

The representation produced by HAIKU is essentially a reorganized parse tree, augmented with elements of meaning. Discourse relations communicated by conjunctions (e.g. causality) are not analyzed by CA. The representation also includes constituents irrelevant to the overall Case structure of the sentence, e.g. adjectives, relative clauses, PPs attached to nouns, clauses with stative verbs expressing noun-noun relations, and so on. These are passed to the next module of TANKA, the Mini-Network Builder.

FUTURE RESEARCH

The new version of DIPETT is operational. It is now being integrated into the INTELLA system (Delisle *et al.* 1991) which combines text analysis with explanation-based learning. A Case Analysis prototype is running and work in this area is actively under way. It includes investigating the character of technical texts, validating the set of Cases used in TANKA, refining the process of mapping CMPs onto example sentences and confirming the design principles behind example-driven interaction with the user by experiment. A re-implementation of the HAIKU module will be completed in the coming months.

CONCLUSION

We have presented the DIPETT parser and the Case Analyzer—the main elements of the linguistic part of the TANKA system. TANKA will process unedited technical text and acquire knowledge about its domain. We want to analyze complete documents with as little user assistance as possible. This means that we must consider incomplete and problematic inputs, although their rate of occurrence should be low in a well-edited text. We have ensured robust low-level processing of text in order to facilitate almost automatic recognition of its structure. At the other end of the spectrum, we plan to handle free segments of text. In contrast with other approaches to language understanding, we do not assume a complete semantic model *a priori*. This imposes certain limitations on what can be processed automatically; we will minimize user interaction. We hope that we have made clear our interest in practical NLP, which we regard as important given the increasing interest in using NLP techniques to assist in acquiring knowledge from text. We believe such techniques will be used more and more commonly for knowledge acquisition tasks and may establish a new trend in the design of tools for knowledge engineers.

ACKNOWLEDGMENTS

This work has been supported by the Natural Sciences and Engineering Research Council of Canada and Cognos Inc.

REFERENCES

BRUCE, B. (1975). “Case Systems for Natural Language”, *Artificial Intelligence*, 6(4), 293-326.

COOK, W.A. (1979). *Case Grammar: Development of the Matrix Model (1970-1978)*, Georgetown Univ. Press, Washington DC.

COPECK, T., Delisle, S. & Szpakowicz, S. (1990). “Intelligent Case Analysis in the TANKA System”, Univ. of Ottawa, Dept. of Computer Science, TR-90-24.

COPECK, T., Delisle, S. & Szpakowicz, S. (1992). “Semantic Analysis in TANKA” (in preparation).

DELISLE, S. & Szpakowicz, S. (1991). “A Broad-Coverage Parser for Knowledge Acquisition from Technical Texts”, *Proc of the Fifth Int Conf on Symbolic and Logical Computing*, Madison, SD, 169-183.

DELISLE, S. (1990). “A Parser for Processing Technical Texts with a Large Coverage of English”, Univ. of Ottawa, Dept. of Computer Science, TR-90-25.

DELISLE, S., Matwin, S., Wang, J. & Zupan, L. (1991). “Explanation-based Learning Helps Acquire Knowledge from Natural Language Texts”, *Proc Sixth Int Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, Oct. 1991, 326-337.

FILLMORE, C. (1968). “The Case for Case”, in E. Bach and R.T. Harms (eds.), *Universals in Linguistic Theory*, Holt, Reinhart and Winston, Chicago, IL.

GRIMES, J. (1975). *The Thread of Discourse*, Mouton, The Hague.

HOBBS, J. (1978). “Resolving Pronoun References”, *Lingua* 44, 311-338.

JENSEN, K. (1989). “A Broad-coverage Natural Language Analysis System”, *Proc Int Workshop on Parsing Technologies* (Pittsburgh, PA), 425-441.

LARSON, M. (1984). *Meaning-Based Translation: A Guide to Cross-language Equivalence*, University Press of America, Lanham, NY.

QUIRK, R., Greenbaum, S., Leech, G. & Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*, Longman.

SOMERS, H.L. (1987) *Valency and Case in Computational Linguistics*. Edinburgh University Press.

SOWA, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

SPARCK-JONES, K. & Boguraev, B. (1987). “A Note on the Study of Cases”, *Computational Linguistics*, 13(1-2), 65-68.

SZPAKOWICZ, S. & Koperczak, Z. (1990). “Mixed-Strategy Matching in Conceptual Networks”. Z. W. Ras, M. Zemankova and M. L. Emrich (eds.) *Methodologies for Intelligent Systems 5*. North-Holland, 321-328.

SZPAKOWICZ, S. (1990). “Semi-automatic acquisition of conceptual structure from technical texts”, *Int J of Man-Machine Studies*, 33, 385-397.

WINOGRAD, T. (1983). *Language as a Cognitive Process (Syntax)*, Addison-Wesley.

YANG, L. & Szpakowicz, S. (1990). “Path-finding in Networks”. *Proc SEARCC ’90 South East Asia Regional Computer Confederation Conf*, Manila, Dec. 1990.

YANG, L. & Szpakowicz, S. (1991a). “Inheritance in Conceptual Networks”. *Proc Sixth Int Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, 191-202.

YANG, L. & Szpakowicz, S. (1991b). “Planning in Conceptual Networks”. F. Dehne, F. Fiala and W.W. Koczkodaj (eds.) *Advances in Computing and Information - ICCI ‘’91. Lecture Notes in Computer Science,* vol. 497, Springer-Verlag, 669-671.

ZADROZNY, W. & Jensen, K. (1991). “Semantics of Paragraphs”, *Computational Linguistics*, 17(2), 171-209.