From Text to Horn Clauses: Combining Linguistic Analysis and Machine Learning

Sylvain Delisle [*], Ken Barker [**], Jean-François Delannoy [**], Stan Matwin [**], Stan Szpakowicz [**]

[*] Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Québec, Canada G9A 5H7
Sylvain_Delisle@uqtr.uquebec.ca

[**] Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
{kbarker, delannoy, stan, szpak}@csi.uottawa.ca

Abstract

The paper describes a system that extracts knowledge from technical English texts. Our basic assumption is that in technical texts syntax is a reliable indication of meaning. Consequently, semantic interpretation of the text starts from surface syntax. The linguistic component of the system uses a broad-coverage, domain-independent parser of English, as well as a user-assisted semantic interpreter that memorizes its experience. The resulting semantic structures are translated into Horn clauses, a representation suitable for Explanation-based Learning (EBL). An EBL engine performs symbol-level learning on representations of both the domain theory and the example provided by the linguistic part of the system. Our approach has been applied to the Canadian Individual Income Tax Guide and examples from it are used in the presentation.

1. Introduction

The goal of our project is knowledge extraction from texts. We are building a system that accumulates knowledge using the smallest possible domain-specific kernel prepared in advance. In the case of texts that we characterize as technical, our approach performs this extraction with *no* advanced, precoded knowledge, assuming the assistance of a cooperative user. The system applies natural language processing and machine learning techniques to process English technical text. The result is a Horn clause rule base representing knowledge about semantic relations among concepts presented by the text. We think that this can be done from scratch, provided the user trains the system in the initial phase of knowledge extraction. User intervention should then decrease with time as the system effectively learns from previous interactions.

This work is supported by a strategic grant from the Natural Sciences and Engineering Research Council of Canada. Thanks to Terry Copeck for commenting on a very late draft of the paper.

Because much knowledge is communicated via textbooks, manuals, handbooks etc., a system like ours will be an extremely useful alternative to "traditional" knowledge acquisition tools. In domains where an expository text is available, its user-assisted processing with our system will provide a first, perhaps unpolished version of the knowledge base. This version may then be debugged and improved by the user.

We note the main features of our approach, along with the ensuing conditions of its applicability.

- Detailed surface-syntactic analysis of a fragment of a technical text[1] precedes semi-automatic analysis of clause-level relations, Case relations and relations inside noun phrases (work in progress). The resulting semantic structure is transformed semi-automatically into Horn clauses. Our research hypothesis is that, in technical texts, syntax gives a reliable indication of meaning: literal interpretation based on surface syntax is usually appropriate [Kieras, 1985] and a high degree of compositionality is possible. A standard, linguistic-theory-neutral grammar of English based on Quirk *et al.* [1985] underlies the parser. Details of parsing and Case-based semantic analysis are presented in Delisle [1994].

- The system needs to be trained by the user. It remembers and generalizes the user's additions and changes to its semantic pattern dictionaries. Saturation with domain-dependent *linguistic* knowledge is gradually achieved as the user moves forward in the given text; the intervention is eventually reduced to simple approval. We are now working on the details of a measure of user interaction to be used as the yardstick of our system's performance. It will consider the amount of interaction (for example, the number of updates of the pattern dictionaries) and the ease

[1] It is widely accepted that technical texts are somehow easier to process, despite the absence of any commonly acceptable definition of what constitutes a technical text. We have assembled a checklist of linguistic properties that make a text technical; a working paper is forthcoming.

of interaction (for example, using Angluin's [1988] oracle types).

- We assume rich syntactic knowledge and detailed semantics of function words. We also rely on publicly available domain-independent lexical knowledge. The Collins dictionary [Karp *et al.*, 1992] is used by the parser for part-of-speech information. WordNet [Miller, 1990] will be used for disambiguation and for semantic clustering [Feng *et al.*, 1994]. No domain-specific knowledge need be assumed: the system can be run with its semantic pattern dictionaries initially empty. Early experiments confirm that Case analysis of a text segment starting with empty dictionaries can produce an acceptably high percentage of the system's hypotheses merely confirmed by the user [Delisle, 1994].

- The system acquires knowledge incrementally during linear processing of the input text. Although the system can return to previously processed fragments just as human readers do, essentially it leads the user forward and learns useful facts even after a single pass over a fragment. The system performs learning at the symbol level in order to acquire accurate and sufficient knowledge that will be used to represent the meaning of the text.

2. Organization of the System

The organization of the system is summarized in Figure 1 (ovals represent modules, light rectangles denote data passed between modules, heavy rectangles are permanent repositories). The MaLTe[2] system receives its linguistic data from the TANKA[3] system. DIPETT[4] is TANKA's noncommittal surface-syntactic parser (Jacobs and Rau [1993] briefly discuss the role of inexact parsing in text processing). A parse tree of the current sentence produced by DIPETT may be reorganized by the phrase reattachment module. The structurally correct parse tree is processed by HAIKU, a three-step interactive semantic analysis module. HAIKU suggests semantic relations among clauses in the sentence (for example, causality, enablement, precedence), then Case patterns in clauses and finally relations inside noun phrases; the user confirms or overrides those suggestions. Relations not encountered earlier are added to HAIKU's semantic dictionaries (not shown in Figure 1).

The result, a composite graph containing syntactic and semantic information about the sentence, is called a *protonetwork* ("early representation of the network"). Within TANKA, it is passed on to the Network Fragment Builder that turns it into a fragmentary conceptual network. This fragmentary network will then be merged with a growing conceptual network representation [Yang and Szpakowicz, 1991] of the part of the source text processed so far.

The protonetwork is simultaneously passed on to the first module of MaLTe, which translates it into a set of Horn clauses. Translations of the narrative part of a text and its examples are distinguished. The Machine Learning module, which includes an EBL[5] engine, organizes these Horn clauses into a domain theory.

There are two main tasks in this operation. First, the domain theory is accumulated and organized by a hierarchization of Horn clauses into a stratified rule set in which levels of rules are clearly delineated. This is achieved by transformations of sets of clauses (for example, absorption) used in Inductive Logic Programming to reorganize clause sets into logic programs. The second task, essential to the MaLTe approach, applies EBL to the clausal representation of the examples. The clausal representation of the explanatory text plays the role of domain theory. This gives a compiled, generalized and operational rendering of the examples which includes the knowledge necessary to explain them.

Horn clauses representing the results of EBL are turned into a (simplified) protonetwork and fed back into the Network Fragment Builder. When the domain theory is sufficiently rich, it may be transmitted to a performance task external to the TANKA/MaLTe system. One example of such a performance task is a rule-based program producing income tax returns. The skeletal rule base containing the knowledge part of this program would be acquired by TANKA/MaLTe directly from the Income Tax Guide.

The system is being implemented in Quintus Prolog on Sun SparcStations. The parser and the Case analyzer are fully implemented. Prototypes of the machine learning mechanisms, the clause-level relationship analyzer and the protonetwork to Horn clause translator are close to completion. The reattachment module, the Network Fragment Builder and the noun-modifier relationship analyzer have been designed.

3. Syntactic and Semantic Analysis

3.1. The Parser

The parser accepts most sentences found in a technical text. Such a broad-coverage parser ensures that acquisition of knowledge from text is reasonably complete. Without a rich semantic model, syntax is the only support for meaning. DIPETT [Delisle and Szpakowicz, 1991; Delisle, 1994] handles, fully or partially, about 90% of the sentences in sample *unedited* texts.

---

2 Machine Learning from Text

3 Text Analysis for Knowledge Acquisition

4 Domain-Independent Parser of English Technical Texts

5 Explanation-Based Learning

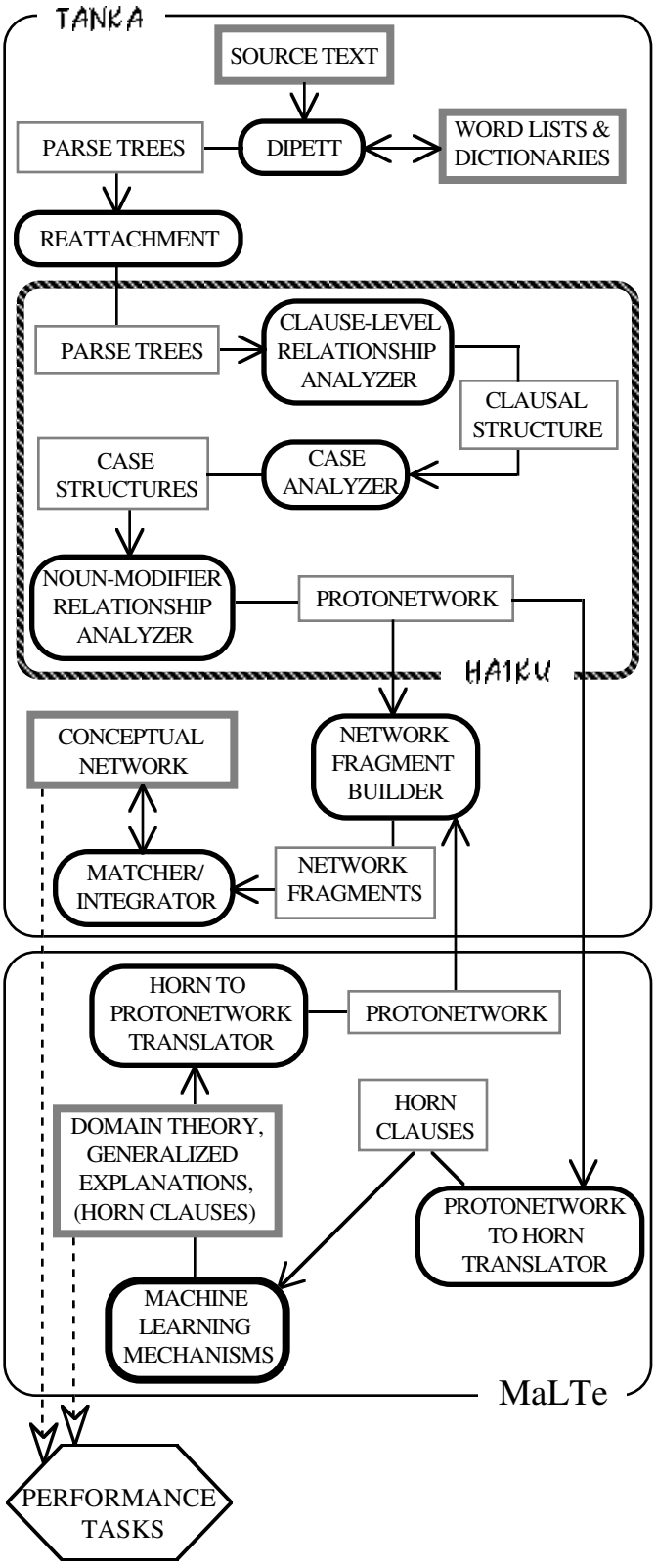

Figure 1. The TANKA and MaLTe Systems

Linguistic theories of syntax such as GPSG, HPSG and LFG use feature structures to encode linguistic objects: these structures' form and content depend on the underlying theory. DIPETT is not committed to any particular linguistic theory. Its DCG grammar formalism is theory-neutral (as are other formalisms, for example, PATR-II). Most of the grammar's rules are based on Quirk *et al.* [1985]. DIPETT may be considered as a functional grammar, that is, one in which syntactic analysis is based on syntactic roles instead of only word order.

In addition to standard parsing functions, the parser contains several subsidiary non-standard components: a simple tagger, a dynamic dictionary expansion facility, a memorizing device (that is, a well-formed substring table or passive chart) and an error explanation mechanism.

3.2. The Case Analyzer

We have defined a Case system which is used by HAIKU. Six Cases appear in this paper's examples: *Agent* (AGT), *Accompaniment* (ACMP), *Beneficiary* (BENF), *Location to* (LTO), *Object* (OBJ), *Time at* (TAT). The complete list of 28 Cases, as well as the motivation and the discussion of other published Case lists are presented in Barker *et al.* [1993].

The Case Analyzer (CA) takes a parse tree produced by DIPETT and semi-automatically extracts the Case pattern that best represents its meaning. Cases are signaled by Case markers, which are realized in two ways: *lexically*, for example, a preposition that introduces a prepositional phrase, and *positionally*, as subject (`psubj`), direct object (`pobj`), indirect object (`piobj`).

CA accumulates Case patterns in its semantic dictionaries and refers to them when processing new sentences. A sentence that has little in common with previously encountered patterns may introduce new elements of knowledge. These are integrated into the incrementally growing dictionaries. For a sentence similar to previously analyzed sentences, CA suggests a semantic interpretation for the user to confirm or reject. CA is instrumental to our intent of turning knowledge-based text understanding into knowledge acquisition. Knowledge acquired during this operation constitutes an important part of the conceptual model of a text's domain.

The main data structures are stored in the Meaning Dictionary, the Case-Marker Pattern Dictionary and the Case Pattern Dictionary. All dictionaries may be empty when the system is first used on a new text. CA does not need any seed knowledge to work properly, though it does require significant involvement of the user in the initial phase. The amount of work decreases as CA processes more sentences and acquires more patterns.

The following definitions describe CA terms used in the rest of this paper.

A **Case-Marker Pattern (CMP)** is an ordered list of Case markers, representing the markers appearing in a clause. In CA, a clause has only one CMP, that is, only one syntactic analysis from which a unique CMP is derived.

A **Case Pattern (CP)** is an ordered list of Case abbreviations, representing the Cases appearing in a clause. In CA, a clause normally has only one CP, although it may have more if the clause is semantically ambiguous.

The **Meaning Dictionary** has entries for individual words. A verb entry contains a list of CMPs found with this verb in the text, and a list of Cases associated with each marker in these CMPs. The Meaning Dictionary also contains entries for prepositional and adverbial Case markers. Such an entry contains a fixed list of Cases the marker can realize; details are in Barker *et al.* [1993].

The **Case-Marker Pattern Dictionary** has entries for CMPs. An entry contains a list of CPs already associated with this CMP. Each CP is illustrated by an example sentence. This dictionary may be initialized with entries for a number of common CMPs. In CA, a CP may be associated with one or more examples since many syntactically different clauses can have the same Case pattern.

The **Case Pattern Dictionary** has CP entries, each containing a list of verbs associated with this CP in the text.

All three dictionaries are continuously updated. As mentioned above, a parsable clause should normally be associated with a unique CP. To accomplish this, CA first searches its dictionaries for the *target CP* that matches the CMP of the input sentence most closely, if not perfectly [Delisle *et al.*, 1993a]. Next, an example sentence to illustrate the target CP is fetched from the dictionary. If the CP and example sentence are not acceptable, the system asks for the user's help.

Note that order does not matter in patterns: it is only their semantic interpretation that counts. Thus, `subj-obj-at-by` is equivalent to `subj-obj-by-at`, and, similarly, AGT-OBJ-LAT-TAT is equivalent to AGT-OBJ-TAT-LAT.

Sentences from the following paragraph will be used as examples to illustrate CA as well as other processes described in the rest of this paper.

> "Jim is a member of the Canadian Armed Forces and was posted to Lahr in 1989. Jim's wife moved with him to Lahr. He broke all residential ties with Canada. Jim is a resident of Canada because he is serving abroad in the armed forces."

The fourth sentence above ("Jim is a resident of Canada because he is serving abroad in the armed forces.") contains two clauses, each of which will be analyzed separately by CA. The main verb of the first clause is the stative verb "be". Although stative verbs introduce facts about objects, activities and their properties, they do not have Cases. Consequently, clauses with stative main verbs are not treated by CA but are passed on for semantic processing by subsequent modules—for a description of the treatment of stative verbs, see Delisle *et al.* [1993b]. The second clause is Case Analyzed. The CMP `psubj-adv-in` is associated with the clause's main verb "serve". CA first checks the Meaning Dictionary to determine if the verb "serve" has appeared previously in the text and what CMPs and CPs have been associated with it. It also checks the CMP Dictionary to see if the CMP `psubj-adv-in` has occurred previously and what CPs have been associated with it. Based on this historical data as well as information about which Cases the individual Case Markers mark, CA suggests a CP to the user. The user may accept the suggested CP or reject it and supply a new CP for this clause. The three dictionaries are then updated to reflect this Case assignment. The output for the sentence (after successful Case assignment) would consist of the following Case structures:

```
case_structure(*statement1*, be, psubj-pobj-of,
 nil, 'Jim', resident, 'Canada')
case_structure(*statement2*, serve, psubj-adv-in,
 agt-lat-benf, 'Jim', abroad, 'the armed forces')
```

3.3. The Clause-Level Relationship Analyzer

Case Analysis deals with semantic interpretation of the relationships between a verb and its arguments within a clause. Semantic information is also conveyed by relationships between clauses. In particular, the causal links which are vital to the construction of rules from text are commonly found at the inter-clausal level. We are completing an extension of the semantic analyzer onto Clause-Level Relationships (CLRs). The design of the CLR Analyzer (CLRA) closely mirrors that of the Case Analyzer. First, a list of semantic relationships was constructed based on an exhaustive study of the lexical items signaling them. This set was then checked for completeness against a number of works in traditional and computational linguistics. The current list of CLRs is: *Causation*, *Enablement*, *Entailment*, *Prevention*, *Detraction*, *Conjunction*, *Disjunction*, *Location*, *Temporal Precedence*, *Temporal Co-occurrence*—details have been presented in Delisle *et al.* [1993b].

During interactive semantic analysis, CLRA is activated when the current input sentence contains syntactically connected clauses. The connective (a conjunction) is matched against a list of potential CLR markers and the CLRs they typically mark. One or more CLRs from this list are suggested to the user who is given the option of accepting a suggested CLR or assigning a new one.

Consider again the sentence "Jim is a resident of Canada because he is serving abroad in the armed forces". CLRA recognizes two clauses in the input connected by the conjunction "because". It finds in its dictionary of CLR markers that "because" often marks Causation, Enablement and Entailment. The user can choose one of these or enter a new CLR. If Entailment is chosen, the analysis will be stored as:

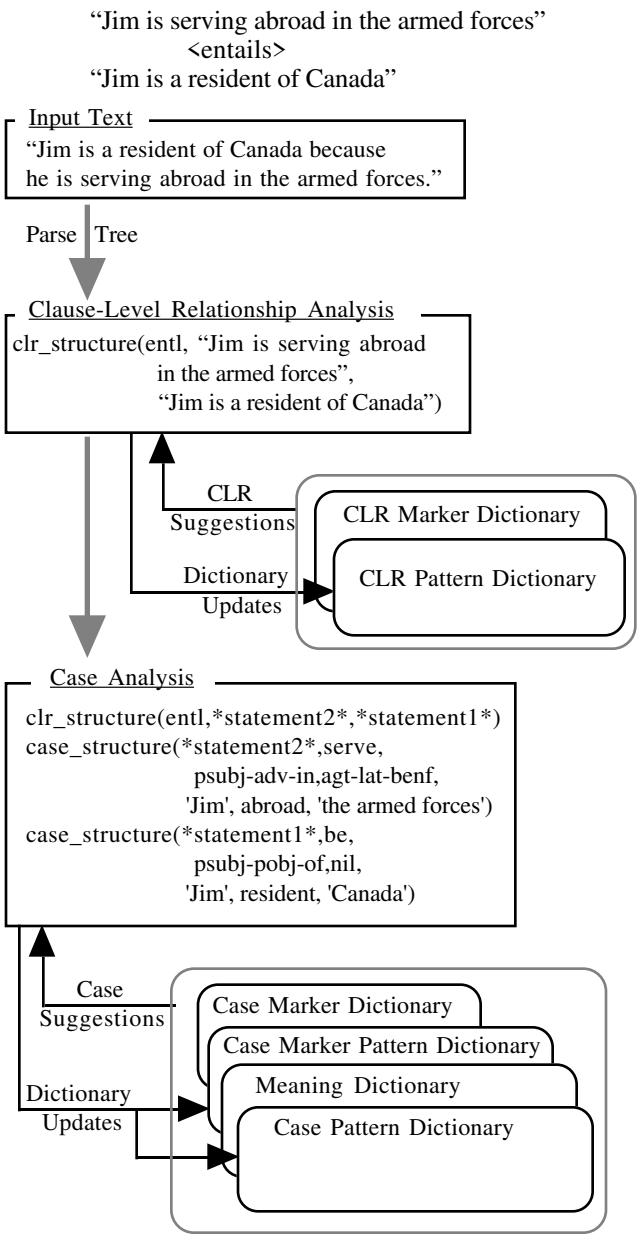

Figure 2. A Condensed Illustration of HAIKU Semantic Processing

The whole process of semantic analysis is summarized in Figure 2.

We have recently incorporated additional linguistic clues in the semantic analysis of CLRs. The syntactic makeup of clauses themselves helps identify the semantic relationships between them. For example, the modality of their main verbs is useful in distinguishing between different causal relationships, as the following example sentences suggest: "If you claim expenses then you *must* have paid money for services" (*Entailment*); "If you paid money for services then you *may* claim expenses" (*Enablement*).

4. The Protonetwork-to-Horn Processor

4.1. Construction of Horn Clauses

In contrast to several other approaches (see section 5), the conversion chain followed in MaLTe involves deriving a linguistically justified semantic encoding, which is then translated into first-order logic. For a moderate additional computational cost, the resulting principled process is also general and portable. Generality relies on the separation of processing tasks from user interaction. The source of domain knowledge is the user, who makes domain-dependent choices in disambiguation and Case or CLR assignment, while the processes of parsing and semantic processing are inherently domain-independent.

Translation from semantics to logic is done after all phrase reattachment and pronoun resolution have been completed. The input is the protonetwork material, which includes the CLR structure of a sentence, the detailed Case structure of its clauses and the internal description of the semantics of noun phrases.

The aspect of a clause (stative versus nonstative) is a key factor. Stative clauses in a technical context are taken to be definitional (for example, "An eligible child could be your child, your spouse's child, etc."), or they can describe an attribute of an instance ("Jim is a member of the Canadian armed forces."). HAIKU assigns no Cases to stative clauses. We translate them with a predicate based on the attribute. Nonstative clauses ("Jim moved to Lahr") are assigned Cases whose labels are attached to the verb to form a specific predicate name (such as `serve_agt_lat_benf`).

The algorithm follows a declarative description of the tree structure produced by HAIKU. The structural organization corresponds roughly to the input grammar of DIPETT, except that the structure to analyze is not a token sequence, but Prolog terms. These terms correspond to the sentence and its clauses; their nodes should be seen as mere functors, not predicates, since they can have variable "arity" according to the presence or absence of optional syntactic constituents.

At sentence level, the clause-level relationship is the most important criterion to determine how to piece together the elements of the Horn clause: with *Causation*, *Enablement* and *Entailment* (the latter especially important in a tax guide), the module asserts a rule. When clauses are simply conjoined, it asserts two independent facts. The user is asked for confirmation in all situations, which include more ambiguous relations like temporal precedence (which may or may not denote causality).

Thus, the fourth sentence:

> "Jim is a resident of Canada because he is serving abroad in the armed forces"

which is represented by :

```
clr_structure(entl, *statement2*, *statement1*)
case_structure(*statement1*, be, psubj-pobj-of,
 nil, 'Jim', resident, 'Canada')
case_structure(*statement2*, serve, psubj-adv-in,
 agt-lat-benf, 'Jim', abroad, 'the armed forces')
```

will be transformed into:

```
is_resident_of(jim, canada) :-
   serve_agt_lat_benf(jim, abroad, armed_forces).
```

If a clause has a negative polarity ("X is not eligible...") then we have to assert a rule involving explicit negation. The most frequent situation, and the simplest, is to simply assert a fact because there is no entailment or opposition relationship between the clause and other clauses, as in the other sentences of the example:

```
is_member(jim, canadian_armed_forces).
post_obj_lto_tat(jim, lahr, 1989).
move_agt_acmp_lto(wife, jim, lahr).
break_agt_obj_benf(jim, residential_ties, canada).
```

Note that some inference or interaction may be needed to relate different encoding of the same concepts, as canadian_armed_forces and armed_forces in the example.

4.2. Learning

The logical translation of both the narrative and the example sections of a text is fed to the learning module which performs Explanation-based Learning (EBL).

EBL is a learning method that generalizes a concept or procedure description from a single example. Rather than discriminating and generalizing from features of a large number of examples, as in the standard inductive approach in the spirit of ID3 [Quinlan, 1986], EBL uses an explanation of just one training instance as the basic learning tool. Explanation is usually a deduction that justifies (for example, through a Prolog-style proof) the statement "this specific instance is an instance of the concept we are learning". The explanation is used for two purposes. Firstly, it identifies the relevant features of the example, which are sufficient conditions for describing the concept [Minton *et al.*, 1990]. Secondly, generalization in EBL is performed by regressing the concept definition through the explanatory structure (for example, an AND tree). Consequently, the generalization process often turns constants of the example into terms, rather than just variables [Mitchell *et al.*, 1986]. Those terms bring into the explanation certain relevant parts of domain knowledge. In order to produce an explanation, an EBL learner must have a domain theory. If the theory is represented in the Horn clause format, it can be easily used to produce an explanation of the example. The concept can be treated as a top-level Prolog goal, the example—as a conjunction of Prolog facts, and the domain theory acts as a Prolog program. If the goal, properly instantiated with the constants of the example, can be proven by a Prolog-like interpreter, EBL succeeds and the goal tree is treated as an explanation. In the system described in this paper, the narrative text is converted—in several steps—into a domain theory, and the examples in the text are used as the training instances.

As described elsewhere [Delannoy *et al.*, 1993], we rely on transformations such as abstraction and absorption from Inductive Logic Programming to organize clauses into a meaningful, hierarchical knowledge base. The EBL process takes as input the Horn clause base produced by the Protonetwork-to-Horn (PtH) module, as well as the clausal representation of the examples from the text, and performs EBL on them. We shall illustrate this process below. Suppose that the domain theory (acquired in a manner described above) contains the following rules:

```
claim_child_care_expenses(P, C, E) :-
   person_deduct_expenses(P),
   eligible_child(C),
   deduct_amount_expenses(E).
person_deduct_expenses(P) :-
   is_resident_of(P, canada), eligible(P).
...
```

Let further facts from the example be produced by the PtH module as shown above. The EBL process continues, producing first the proof tree as in Figure 3 (only a fragment is shown).

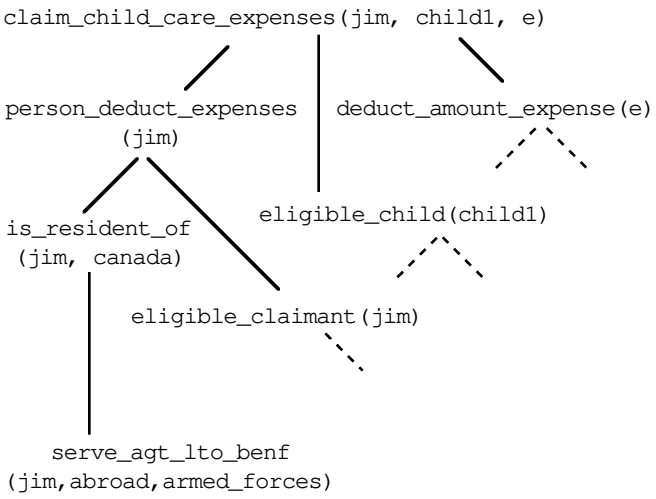

Figure 3. Fragment of the Proof Tree Produced by the EBL Process

EBL extracts from the whole collection of facts available in a given example exactly those that are necessary to prove that the example satisfies the concept definition (here, that it is an instance of the concept `claim_child_care_expenses`). EBL also puts together (compiles) all the knowledge necessary to show the membership of the example in the concept. Moreover, there

is generalization in the second phase of EBL that consists in regression of the domain theory rules through the proof tree. Generalization will produce a useful ("operational", in the EBL terminology) generalization, that is:

```
claim_child_care_expenses(P, C, E) :-
   serve_agt_lat_benf(P, abroad, armed_forces),
   ...
```

Such a rule is then added to the domain theory. Consistent with the EBL paradigm, symbol-level learning has been achieved, and it has made the theory more useful than its general rendering (it applies now *directly* to all armed forces personnel stationed overseas), and at the same time more general than the specific example provided in the text. Unlike in simple inductive systems, the generalization obtained is fully justified by the existing domain theory.

5. Related Work

Work on classification tasks, useful in knowledge base (KB) construction, uses text processing as a means of extracting classification knowledge from various textual material. Silvestro [1988] and Gomez [1989] are two examples of that approach. Moulin and Rousseau [1992] describe a system which looks for predetermined, fixed patterns (for example, 'if', 'because', 'when') in the input sentences and decompose the original sentences into representations that stand for production rules or KB elements. Ciravegna *et al.* [1992] present the SINTESI system. It extracts knowledge from short (4 or 5 sentences) descriptive diagnostic reports written in Italian, in order to summarize their technical content and support the constructing of a KB on car faults. All objects that may be of interest in a text are described in a KB that is available *a priori* and there is no incremental KB augmentation.

Kim and Moldovan [1993] describe PALKA, a semi-automatic KA system designed to facilitate the construction of a large KB of semantic patterns accumulated from corpora. PALKA requires much *a priori* knowledge: a general and domain-specific concept hierarchy and frame definitions telling the system what to look for in a text (along with the relevant keywords). Liu and Soo [1993] have implemented a system that attempts to assign thematic roles to sentence elements using minimal *a priori* knowledge. The system uses syntactic clues to propose an initial set of potential thematic roles. This set is then pruned by applying heuristics and consulting the user.

Given the knowledge-intensive character of Natural Language Processing, surprisingly little work has been done in applying machine learning techniques in the context of NLP systems. Hauptmann [1993] describes how a relatively unsophisticated, rote-learning mechanism helps in the acquisition of a mapping from syntax to the meaning of sentences in a given domain. Zelle and Mooney [1993] and Aliprandi [1993] show how different learning techniques, inductive logic programming and standard induction, apply in the resolution of the propositional phrase attachment problem. A separate research community [Powers, 1989] focuses on the difficult questions of applying machine learning in the attempt to understand the cognitive aspects of language learning. Cohen [1990] shows how crucially learning from texts relies on a flexible definition of operationality. His work concentrates on the learning aspects without attempting to develop an integrated NLP-ML system.

6. Conclusion

We propose a combination of partially automated text processing and explanation-based learning for knowledge extraction from unedited technical texts. Early experiments show that this alliance yields more knowledge than standard language processing methods alone; at the same time, novel learning problems and opportunities originate from the fact that the domain theory is semi-automatically constructed directly from the text. Such mutual enrichment of natural language processing and machine learning lies in a largely uncharted territory. Challenging questions arise from the design of the first version of the system. The transformation of text fragments into a domain theory (expressed in first order logic) adequate for explanation-based learning requires intensive user participation, if the knowledge acquisition exercise is to be meaningful. However, learning (inductive generalization of user's interventions) is expected to decrease the amount of user interaction. The characterization of this amount as a function of the properties of the text and of the gradual saturation of the knowledge base through experience is an open research problem. Reliance on surface syntax as the carrier of meaning is vindicated by the ability of the linguistic subsystem to work up from an empty domain-specific knowledge base. Research problems in learning include explanation-based learning in an unavoidably incomplete domain theory, dynamic modification of the operationality criterion according to the changing performance task, and extension of learning from a first order logic rule base onto sorted logic.

References

[Aliprandi and Saviozzi, 1993] G. Aliprandi and G. Saviozzi, "A Supervised Learning Method to Solve PP-Attachment Ambiguities in Natural Language", *Proceedings Machine Learning and Text Analysis Workshop, ECML-93*, Vienna 1993, 45-52.

[Angluin, 1988] D. Angluin, "Queries and Concept Learning", *Machine Learning*, **2**(4): 319-342, 1988.

[Barker *et al.*, 1993] K. Barker, T. Copeck, S. Delisle and S. Szpakowicz, "An Empirically Grounded Case System", submitted to the *International Journal of Lexicography*, 36 pages.

[Ciravegna *et al.*, 1992] F. Ciravegna, P. Campia and A. Colognese, "Knowledge Extraction from Texts by SINTESI", *Proceedings 15th International Conference on Computational Linguistics—COLING-92*, Nantes 1992, 1244-1248.

[Cohen, 1990] W. W. Cohen, "Learning from Textbook Knowledge: A Case Study", *Proceedings AAAI-90*, 743-748, 1990.

[Delannoy *et al.*, 1993] J.-F. Delannoy, C. Feng, S. Matwin and S. Szpakowicz, "Knowledge Extraction from Text: Machine Learning for Text-to-Rule Translation", *Proceedings Machine Learning and Text Analysis Workshop, ECML-93,* Vienna 1993, 1-7.

[Delisle and Szpakowicz, 1991] S. Delisle and S. Szpakowicz, "A Broad-Coverage Parser for Knowledge Acquisition from Technical Texts", *Proceedings 5th International Conference on Symbolic and Logical Computing — ICEBOL5* (Madison, S.D., USA), April 1991, 169-183.

[Delisle, 1994] S. Delisle, "Text Processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic Analysis for Incremental Knowledge Acquisition", Ph.D. thesis, Department of Computer Science—Ottawa-Carleton Institute for Computer Science, TR-94-02, University of Ottawa, 1994.

[Delisle *et al.*, 1993a] S. Delisle, T. Copeck, S. Szpakowicz and K. Barker, "Pattern Matching for Case Analysis: A Computational Definition of Closeness". O. Abou-Rabia, C. K. Chang and W. W. Koczkodaj (eds.) *Proceedings ICCI-93*, 310-315.

[Delisle *et al.*, 1993b] S. Delisle, K. Barker, T. Copeck and S. Szpakowicz, "Interactive Semantic Analysis of Technical Texts: Case Pattern Acquisition", submitted to *Computational Intelligence*, 67 pages, 1993.

[Feng *et al.*, 1994] C.Feng, T. Copeck, S. Szpakowicz and S. Matwin, "Semantic Clustering. Acquisition of Partial Ontologies from Public Domain Lexical Sources". *Proceedings AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff 1994, 1-15.

[Gomez, 1989] F. Gomez, "Knowledge Acquisition from Natural Language for Expert Systems Based on Classification Problem-Solving Methods", *Proceedings 4thAAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff 1989, 15.1-15.18.

[Hauptmann, 1993] A. G. Hauptmann, "Meaning from Structure in Natural Language Interfaces", Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University, 1993.

[Jacobs and Rau, 1993] P. S. Jacobs and L. F. Rau, "Innovations in Text Interpretation", *AI Journal* 63(1-2) (Special Issue on Natural Language Processing), October 1993, 143-191.

[Karp *et al.*, 1992] D. Karp, Y. Schabes, M. Zaidel and D. Egedi, "A Freely Available Wide Coverage Morphological Analyzer for English", *Proceedings 15th International Conference on Computational Linguistics—COLING-92*, Nantes 1992, 950-955.

Kieras, 1985] D. E. Kieras, "Thematic Processes in the Comprehension of Technical Prose", in B. K. Britton and J. B. Black (eds.), *Understanding Expository Text (A Theoretical and Practical Handbook for Analyzing Explanatory Text)*, LEA, 89-105, 1985.

[Kim and Moldovan, 1993] J.-T. Kim and D. I. Moldovan, "Acquisition of Semantic Patterns for Information Extraction from Corpora", *Proceedings 9thIEEE Conference on AI Applications*, 171-176, 1993.

[Liu and Soo, 1993] R.-L. Liu and V.-W. Soo, "An Empirical Study on Thematic Knowledge Acquisition based on Syntactic Clues and Heuristics", *Proceedings 31st Annual Meeting of the ACL*, Columbus, Ohio, 1993, 243-250.

[Miller, 1990] G. A. Miller, (ed.), "WordNet: An On-Line Lexical Database", *International Journal of Lexicography*, **3**(4), 1990.

[Minton et al., 1990] S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni and Y. Gil, "Explanation-Based Learning: A Problem Solving Perspective", Machine Learning (Paradigms and Methods), J. Carbonell (ed.), MIT Press, 63-118, 1990.

[Mitchell et al., 1986] T. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-based Generalization: A Unifying View", *Machine Learning*, **1**(1): 47-80, 1986.

[Moulin and Rousseau, 1992] B. Moulin and D. Rousseau, "Automated Knowledge Acquisition from Regulatory Texts", *IEEE Expert*, October 1992, 27-35.

[Powers and Turk, 1989] D. M. Powers and C. R. Turk, *Machine Learning of Natural Language*, Springer-Verlag, 1989.

[Quinlan, 1986] J. R. Quinlan, "Induction of decision trees", *Machine Learning* 1: 81-106, 1986.

[Quirk *et al.*, 1985] R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, *A Comprehensive Grammar of the English Language*, Longman, 1985.

[Silvestro, 1988] K. Silvestro, "Using Explanations for Knowledge-Based Acquisition", *International Journal of Man-Machine Studies*, **29**: 159-169, 1988.

[Yang and Szpakowicz, 1991] L. Yang L. and S. Szpakowicz, "Inheritance in Conceptual Networks". Karen S. Harber (ed.) *Proceedings Sixth International Symposium on Methodologies for Intelligent Systems (Poster Session)*. Charlotte, NC, 1991, 191-202.

[Zelle and Mooney, 1993] J. M. Zelle and R. Mooney, "ILP Techniques for Learning Semantic Grammars", *Proceedings ILP Workshop, IJCAI-93*, Chambery (France), 83-92, 1993.