HPARSER: Extracting formal patient data from free text history and physical reports using natural language processing software

Jeffrey L. Sponsler, MD MS

Abstract

A prototype, HPARSER, processes a patient history and physical report such that specific data are obtained and stored in a patient data record. HPARSER is a *recursive transition network* (RTN) parser, and includes English and medical *grammar rules*, *lexicon*, and *database constraints*. *Medical grammar rules* augment the grammar rule base and specify common phrases seen in patient reports (e.g., “pupils are equal and reactive”). Each *database constraint* associates a grammar rule with a database *table* and *attribute*. Constraint behavior is such that if a rule is satisfied, data is extracted from the parse tree and stored into the database. Control reports guided construction of grammar and constraint rules. Test reports were processed with the control rules. 85% of test report sentences parsed and a 60% data capture rate, compared to controls, was achieved. HPARSER demonstrates use of an RTN to parse patient reports, and database constraints to transfer formal data from parse trees into a database.

Introduction

This paper describes the development of History Parser (HPARSER), a prototype that reads a patient history and physical (H&P) file, divides the words into sentences, parses each sentence, creates a parse tree for each sentence, extracts data from each parse tree, and populates database objects with the extracted data. HPARSER contains grammar rules, a lexicon, and database constraint rules (which associate parse tree patterns and database tables), a Parsing Module (PM), and a Database Constraint Processor (DCP).

Natural language processing (NLP) systems have been used to interpret free text medical reports. Sager [1] converted free text patient narratives into SNOMED International codes. Zweigenbaum [2] evaluated Menelas which analyzes discharge summaries. Friedman [3] reports on MedLEE which was trained to process radiology reports. Jain [4] used MedLEE to identify radiology reports to detect tuberculosis. Evans [5] processed drug dosing phrases with an 80% match rate. Knowledge based techniques (rules, symbolic reasoning) have been applied to problems such as automated scheduling [6, 7]. This technology is employed in HPARSER.

Hypotheses

The following hypotheses are addressed by this work: (H1) A *recursive transition network parser* is sufficient to process the sentences in a patient history and physical report. (H2) The inclusion of *medical grammar rules* that identify commonly used phrases improves the rate of phrase capture. (H3) The *database constraint* that links a specific grammar rule to a database table and attribute provides a mechanism whereby information may be extracted from parse trees and stored into a database. (H4) Grammar rules and database constraints constructed to process control H&P reports will generalize so as to be applicable to novel H&P reports.

Development of HPARSER

HPARSER consists of a Parsing Module and a Database Constraint Processor. Knowledge inputs include a Lexicon, English Grammar Rules, Medical Grammar Rules, and Database Constraints. Figure 1 illustrates the modules, the knowledge bases, and the data path of HPARSER.

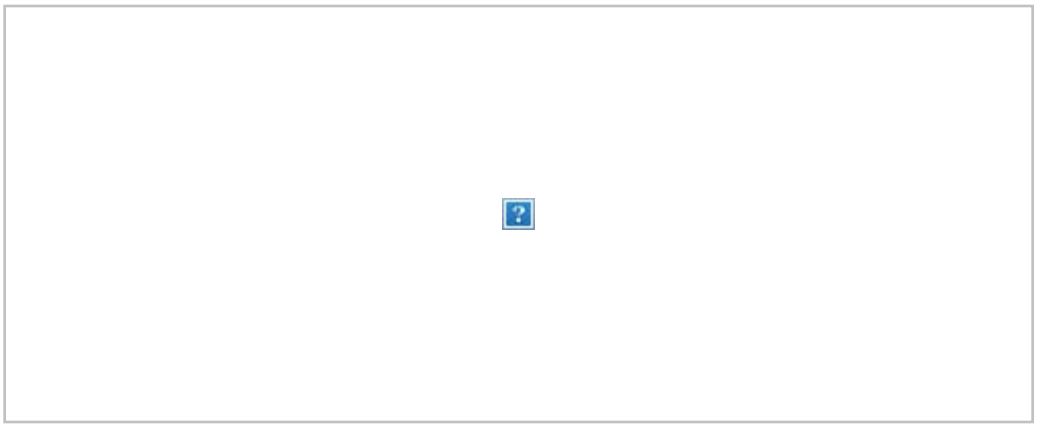

Figure 1. HPARSER Architecture.

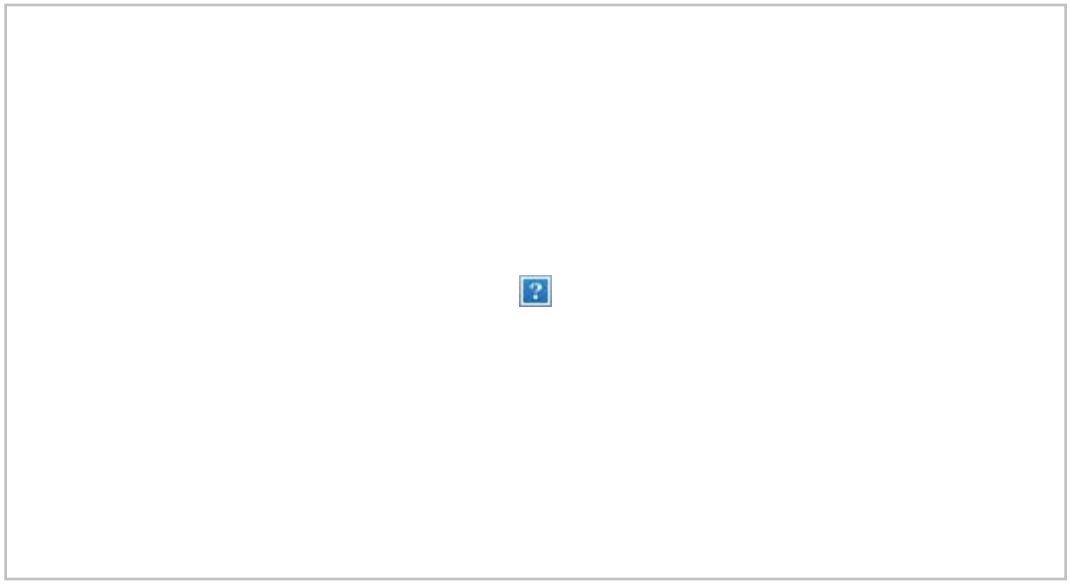

Figure 2. Recursive Transition Network Examples. The RTN's depicted above represent rules for prepositional phrases, blood pressure, and modifiers.

The Parsing Module

HPARSER has been implemented in Common Lisp and the Common Lisp Object System (CLOS). HPARSER uses the *recursive transition network* [8] approach to NLP. The recursive transition network (RTN) is a memory resident representation of grammar rules. Each rule is defined explicitly (e.g., "noun-phrase = determiner noun") and converted to a network in memory. Each grammar rule is represented as a directed graph with each arc labeled by a transition predicate such as *word class*, *literal string*, or a recursive reference to another rule.

Parsing Algorithm. The Parsing Module searches through the domain of rule networks, attempts to match a given network with the input tokens, and consumes input tokens when these tokens satisfy the transition predicates. The Parsing Module (PM) employs depth first or breadth first search and stops at the first legal parse. The Parsing Module does not attempt to determine all parse trees for a given sentence, nor to search for the "best" parse tree. The *Parsing Module Heuristic* states that sufficient information is made available by a single successful parse to attain useful data.

The output of the Parsing Module for a given sentence is a parse tree or nil if the parse fails. It is expected that some fraction of novel inputs will fail the parsing step. This prediction is made based on the ambiguous nature of English grammar, nonstandard grammar use, and rule base incompleteness. The percentage of sentences that parse is used to quantify the software's behavior and is reported in a later section.

The HPARSER Sentence Identifier

The patient report file is processed initially by the Sentence Identifier such that each word becomes a string token, punctuation characters are converted to symbolic representations (e.g., "|comma|"), and sentences are terminated by periods, question marks, colons, and semicolons (which are then discarded). Each sentence is represented as a list of string tokens. Processing of "Neuro: Cranial nerves are intact." results in the lists:

```
(neuro)
(cranial nerves are intact)
```

The HPARSER Lexicon

A lexicon has been constructed for this prototype and consists of the words that are found in the four H&P's that have been processed for this investigation. Each word entry includes part of speech and variants (for plurals and verb conjugation). New word classes have been constructed to support specific parsing goals: numeral, punctuation, date, and measurement. Examples of word definitions follow:

```
(verb "do")
(adverb "suddenly")
(conjunctive "and")
(pronoun "her")
(numeral "one")
(preposition "about")
```

During parsing, when an input token (e.g., "under") and an arc transition predicate (e.g., "preposition") match, a new object, the *token-link* is created. The token and the lexicon entry are stored in the token-link, and the token-link is added to the growing parse tree. An unsuccessful retrieval returns nil and another path through the grammar rule base is selected and investigated.

rule-name	Unique name of the rule (e.g., noun-phrase)
initial *node*	The first node in the network is *node*.
final *node*	The named *node* is a legal stopping point in the network.
from *node1* to *node2* by *label*	Defines an arc from *node1* to *node2* with transition *label*.

from *node1* to *node2* by literal "*string*"	Defines an arc from *node1* to *node2* via a specific character *string*.
= *label1 label2 ... label-n*	Defines a linear series of nodes with legal transitions which include the named *label1, label2, ... label-n.*
== *string1 string2 ... string-n*	Defines a sequence of arc transitions that are literal strings.
not *label1 label2 ... label-n*	Defines a set of arc labels that if seen on input disallows the rule from being selected.

Table 1. The syntax of Lisp rule definitions is (*rule-name* (*rule-components*)+). Components are listed in this table. Italicized terms are variable and all other terms are constants.

The HPARSER Rule Base

The parsing rule base consists of two set of rules (English grammar rules and medical grammar rules) that are employed to effect syntactic analysis of the sentences. Since rules are implemented as networks (or nets), the terms rules and nets are used synonymously in this paper. Rule syntax is seen in Table 1.

As described in Table 1, arcs can be defined explicitly via "from *node1* to *node2* by *label*" syntax or implicitly using the "=" operator. Literal string transitions can be defined using quotation marks or the "==" operator. Examples follow:

```
(prepositional-phrase
 (initial 0) (final 3)
 (from 0 to 1 by preposition)
 (from 1 to 2 by determiner)
 (from 2 to 3 by noun))
(sodium (= "sodium"))
(chloride (== chloride))
```

Grammar Rules

The English Grammar Rule base consists of definitions of sentence, noun phrase, verb phrase, prepositional phrase, and so on. Examples are included below.

```
(noun-phrase
 (= modifier noun))
(modifier
 (= determiner)
 (= possessive-pronoun))
```

A novel concept, the *medical grammar rule*, was devised to represent phrases that frequently occur in H&P's. Examples include "regular rate and rhythm," "Na 135, K 4.6," "alert and oriented times 4," etc. Examples follow.

```
(lab-value
 (= "sodium" numeral)
 (= "potassium" numeral))
(blood-pressure (= "bp"
 numeral |slash| numeral))
```

The Parse Tree

A parse is complete when all input tokens are consumed and, for each network that has been traversed, the parsing module has stopped at a final node. During parsing, a data structure called the parse tree is constructed in memory and returned. The tree is a binary tree where each branch is a subtree and each terminal is a *token-link* data structure. Each token-link associates a token with a lexicon word object. An example is shown here:

```
Class: TOKEN-LINK
 TOKEN: <TOKEN na>
 WORD: <NUMERAL 135>
```

Database Constraint Processor (DCP)

The *database constraint processor* (DCP) populates the database from parsing results. By processing each parse tree with knowledge that is stored in database constraint rules, information is extracted from the parse trees. The DCP embodies two heuristics: The parse tree has a

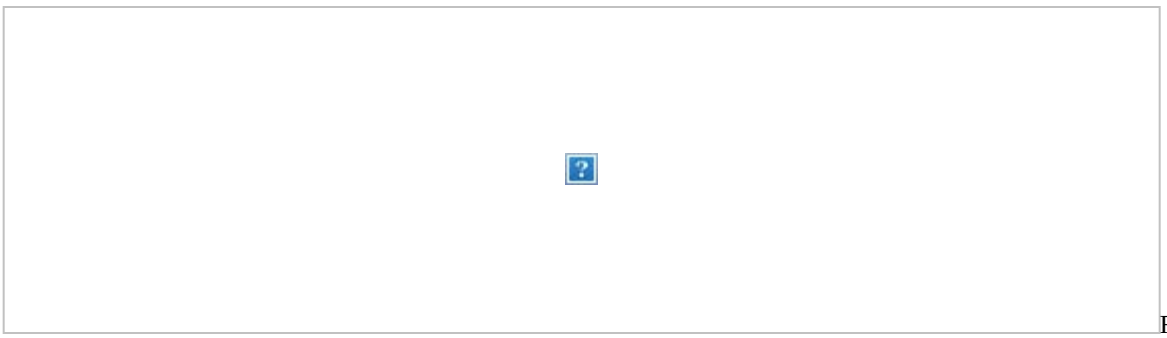

Figure 3. Parsing the phrase "Na 135, K 3.5" produces this parse tree. Several rules are represented here by name including lab-values, lab-value, sodium, and potassium. The tree captures the search path, the rules that have been satisfied, and the token-links.

predictable and known structure and information can be extracted from the tree. Each constraint rule associates a grammar rule with a database table and attribute. Several types of constraints exist and include the following: The *simple database constraint* when satisfied stores a precise value into the database. The *datatype search constraint* searches through the tree for a specific datatype (e.g., numeral) and, if found, stores the associated token. The *word search constraint* searches the tree for a string and if found stores that token. The *subnet search constraint* names a second subtree (i.e., the subnet) for which to search within the scope of the parse tree. When found, data is pulled from the subtree and stored. Examples follow.

```
(define-db-constraint
  :net 'nkda
  :store 'none
  :table 'patient-data
  :attribute 'allergies)
(define-datatype-search-constraint
  :net 'systolic-pressure
  :datatype 'numeral
  :table 'vital-signs
  :attribute 'systolic-pressure)
(define-word-search-constraint
  :net 'patient-id
  :words '(male female)
  :table 'patient-data
  :attribute 'sex)
```

Sentence Analysis Example

A sentence is processed through the major steps of the HPARSER system to illustrate its operation.

```
ID: The patient is a 15 year old male.
```

The parse tree that results is represented here in Lisp:

```
(<Net SENTENCE>
 (<Net PATIENT-ID>
  (<Net SENTENCE-OF-BEING>
   (<Net NP-CONJUNCTIVE>
    (<Net NP>
     (<Net MODIFIER>
      <DETERMINER the>)
     (<Net NOUN>
      <COMMON-NOUN patient>)))
  (<Net VP-BEING><BEING-VERB is>)
  (<Net CONJUNCTIVE-OBJECT>
   (<Net OBJECT>
    (<Net NP> (<Net MODIFIER>
                <DETERMINER a>
 (<Net AGE-IN-YEARS> <NUMERAL 15> <STRING year> <STRING old>))
  (<Net NOUN>
   <COMMON-NOUN male>)))
```

Two constraint rules seen here are activated and data (age and sex) are extracted and stored.

```
(define-word-search-constraint
  :net 'patient-id
  :words '(male female)
  :table 'patient-data
  :attribute 'sex)
(define-datatype-search-constraint
  :net 'age-in-years
  :datatype 'numeral
  :table 'patient-data
  :attribute 'age)
Table: PATIENT-DATA
  AGE: 15  SEX: male
```

Experiments and Results

Two control H&P reports guided development and two test reports were used to exercise HPARSER. No changes to grammar rules, constraint rules, or the software engines were made to accommodate these reports. Minor changes included correction of spelling errors, addition of periods to the end of all sentences, and the addition of colons to all section headings. HPARSER consists of a 1266 word lexicon, 94 English grammar rules, 212 medical grammar rules, and 104 database constraints. Results are in Table 2. For the constraints that were written, data extraction was satisfactory and included age, sex, race, history of present illness, vital signs, laboratory values, review of systems, and exam. Data capture for vital signs and laboratory values was complete and 100% accurate and this relates to the specific nature of these sentences (and thus the ease of analysis).

Incorrect Results and Problems

The correctness of the parse trees varied from sentence to sentence. Common errors included incorrect prepositional phrase attachment and incorrect conjunction (e.g. “and”) tree structure.

The word search constraint logic searches for key terms within a subtree. This constraint is the most flexible but also the least predictable and the results of the search may be incorrect. Computing and assigning a measure of probable correctness to this constraint may be a heuristic that preserves the utility while attempting to quantify the truth value of such rules.

Report	Sentences	Process Time	% Parsed	Elements Stored in Database
Control 1	186	6.0 min	97%	103
Control 2	168	9.5 min	91%	96 (93% of Control 1)
Test 1	179	12.0 min	85%	62 (60% of Control 1)
Test 2	206	15.0 min	82%	55 (53% of Control 1)

Table 2. Results from parsing and constraint processing of 2 control and 2 test reports.

Discussion

Hypotheses listed above are discussed: (H1) The evidence strongly supports this hypothesis. A large fraction of sentences in the reports parsed and even in the cases where the parse trees had erroneous structure, the word class identification and uncomplicated phrase structure identification (such as prepositional phrases) were correct. (H2) In every case where a medical grammar rule could be constructed, parsing behavior and parse tree interpretation were improved. (H3) The database constraint system was quite effective in transferring information from parse trees to the database. Since the parse tree data structure is very organized (compared to free text), constraint composition is facilitated. Further analysis of parse trees should reveal patterns upon which new constraints can be written. (H4) The processing statistics noted above suggest that HPARSER algorithms generalize and that this generalization may be related to the stereotypical nature of the H&P reports. Expanded rule bases should broaden the processing power of HPARSER such that many different reports could be processed.

In summary, this prototype demonstrates facility in extracting formal patient data from history and physical reports, includes new mechanisms (medical grammar rules and database constraints) for text analysis, and provides evidence that further evolution of this software will result in a useful medical informatics system.

Future Work

The next phase (under development) of HPARSER will employ the Unified Medical Language System Lexicon and Index [9]. The grammar and constraint rulebases will be incrementally increased in size and quality to move from a prototype to a functioning system. An algorithm to support fuzzy match (where “glioblastoma” matches “gleoblastoma”) will be included so that typographical errors do not fail a parse. Meta-rules are planned wherein a module scans a sentence (for key tokens), selects the context, and applies context sensitive rules. An algorithm to automatically detect common medical phrases and to write corresponding medical grammar rules will be considered. The design of a neurology expert system is in progress currently and HPARSER will be employed as a front-end.

References

[1] Sager, N., M. Lyman, N. Nhan, L. Tick, 1995. Medical Language Processing Applications to Patient Data Representation and Automatic Encoding, *Methods of Information in Medicine* 1995: 140-146.

[2] Zweigenbaum, P., J. Bouaud, B. Bachimont, J. Charlet, J. Boisvieux, 1997. Evaluating a normalized conceptual representation produced from natural language patient discharge summaries, *Proceedings AMI Annual Fall Symposium* 1997: 590-594.

[3] Friedman, Carol, 1997. Towards a comprehensive medical language processing system: methods and issues, *Proceedings AMI Annual Fall Symposium* 1997:595-599.

[4] Jain, Nilesh, C. Knirsch, C. Friedman, G. Hripcsak, 1996. Identification of suspected tuberculosis patients based on natural language processing of chest radiograph reports. *Proceedings AMI Annual Fall Symposium* 1996:542-546.

[5] Evans, David, N. Brownlow, W. Hersh, E. Campbell, 1996. Automating concept identification in the electronic medical record: an experiment in extracting dosage information. *Proceedings AMI Annual Fall Symposium* , 1996:388-392.

[6] Sponsler, Jeffrey L., Mark Johnston, Glenn Miller, Anthony Krueger, Michael Lucks, Mark Giuliano, 1991. An AI Scheduling Environment for the Hubble Space Telescope, *Proc of Computing in Aerospace 8*, October 21-24, 1991 (Balt, MD) pp 14-24.

[7] Sponsler, Jeffrey L., 1994. A Criterion Autoscheduler for Long Range Planning, *Proc. of the 1994 Goddard Conference on Space Applications of Artificial Intelligence* (Greenbelt, MD).

[8] Gazdar, G, C. Mellish, 1989. *Natural Language Processing in LISP* (Addison-Wesley, 1989).

[9] McCray, A., S. Srinivasan, A. Browne, 1994. Lexical methods for managing variation in biomedical terminologies. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care*, pp. 235-239.

Acknowledgments

The author wishes to thank Dr. Robert D’Alessandri, Dr. John Traubert, and Dr. Paul Brown (of the WVU School of Medicine), and Dr. Frances VanScoy (of WV EPSCOR) for their support of this work.