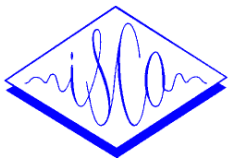

3rd International Conference on
Spoken Language Processing (ICSLP 94)
Yokohama, Japan
September 18-22, 1994

Recent results in automatic learning rules for semantic interpretation

Roland Kuhn (*) and Renato De Mori (**)

(*) Centre de Recherche Informatique de Montréal,
1801 McGill College Avenue, Suite 800, Montréal, Canada H3A 2N4 (kuhn@crim.ca)
(**) School of Computer Science, McGill University, 3480 University Avenue,
Montréal, Canada H3A 2A7 (demori@cs.mcgill.ca)

Abstract

This article describes a new method for building a natural language understanding (NLU) system, in which the system's rules are learnt automatically from training data. The method has been applied to design of a speech understanding (SU) system. Designers of such systems rely increasingly on *robust matchers* to perform the task of extracting meaning from one or several word sequence hypotheses generated by a speech recognizer; a robust matcher processes semantically important islands of words and constituents rather than attempting to parse the entire word sequence.

1 Introduction

A new method was recently introduced [7] for building a natural language understanding (NLU) system, in which the system's rules for semantic interpretation are learnt automatically from training data. The rules are encoded in a forest of specialized decision trees called **Semantic Classification Trees** (SCTs). Application of the SCT method to a speech understanding (SU) task showed that the learned rules are robust to grammatical or lexical errors in the input. By eliminating the need to code and debug a large number of such rules by hand, the methodology facilitates rapid construction of NLU systems for which annotated corpora are available.

SCTs are building blocks that can be used in a variety of ways by a system designer, subject only to the availability of training data; for instance, they can be used in conjunction with a conventional parser. In our particular application, a word string was preprocessed by a bottom-up parser that recognized and labeled certain semantically important phrase constituents while leaving most of the string unchanged [7, 8]. The partly-parsed word string then passed through a forest of SCTs, each of which generated a different aspect of the representation by recognizing patterns made up of constituents, other words, and gaps.

In contrast with a structural pattern recognition approach [4], SCTs make a decision about new patterns on the basis of statistical classification rules learned from patterns seen earlier. Operations performed on gaps lie at the heart of the SCT-growing algorithms. No grammatical inference technique that we are aware of permits gaps: the goal of grammatical inference is to account for all the symbols in a given string.

SCTs have the following properties:

- They learn rules for classifying new strings or substrings from a corpus of classified strings or substrings. To apply SCTs to a problem, one must formulate it as a classification problem.
- The questions in the nodes of an SCT involve regular expressions made up of string symbols and a special gap symbol. The string symbols could be words or higher-level constituents.
- Generation and selection of questions is completely automatic: any symbol from the symbol lexicon may appear in a question.
- There are two main types of SCT: *single-symbol* SCTs and *set-membership* SCTs. Either type can be employed to classify a whole string, or to classify substrings within a string.

Results of a recent experiment are presented in the following.

2 Building Semantic Classification Trees (SCTs)

2.1 Introduction

An SCT is a specialized classification tree that learns semantic rules. To grow classification trees, one must supply three elements [2]:

1. A set of possible yes-no questions that can be applied to data items;
2. A rule for selecting the best question at any node on the basis of training data;
3. A method for pruning trees to prevent over-training.

In our application, a data item is a symbol (usually word) sequence. The original aspect of SCTs is the way in which the set of possible questions is generated. These questions ask whether a word sequence matches certain regular expressions involving words and gaps.

To choose a question from this set, we use the Gini "impurity" $i(T)$ of a node T [2]. The best question for T is considered to be the question which brings about the greatest drop in impurity in going from T to its children. If the two children of T are denoted YES and NO, and the proportions of items at T that a question will send to the YES and NO children are denoted p_Y and p_N respectively, consider the *change in impurity* defined as

$$\Delta I = i(T) - p_Y * i(YES) - p_N * i(NO).$$

The question chosen at node T will be a question that maximizes ΔI.

To prevent over-trained trees, we use the algorithm described in [5], which involves iterative cycles of expansion and pruning on two equal-sized disjoint sets of training data.

Single-symbol SCTs tend to have more nodes in NO than in YES subtrees (because a particular sentence will probably not contain a given word from the lexicon). This implies that sentences that differ only by a synonym will be assigned to different subtrees by a single-symbol SCT.

A set-membership question asks about the presence or absence of any member of a set of words at a given position, thus allowing system-defined synonyms. Word sets are inserted into the gaps + in the known structure, similar to what happens with individual symbols in the single-symbol SCT. If S stands for an automatically generated word set, the questions considered at a given + are obtained by replacing it with S, $S+$, $+S$, and $+S+$.

The algorithms for growing SCTs (either single-symbol or set-membership) can be adapted to the task of classifying substrings. An example from the ATIS task described in Section 4 will illustrate this case. Consider the sentence "show me flights from Boston no sorry from New York to Chicago stopping over in Pittsburgh". After local parsing, this would be "show me flights from CIT no sorry from CIT to CIT stopping over in CIT". The CITs should be labelled as follows: "show me flights from CIT⇐SCRAP no sorry from CIT⇐ORI to CIT⇐DEST stopping over in CIT⇐STOP", where "ORI" is flight origin, "DEST" is destination, "STOP" is stopover, and "SCRAP" means the CIT is irrelevant.

The SCT-growing algorithms require only minor modification to grow SCTs that classify parts of strings. The key is to submit the same sentence to an SCT as many times as there are substrings to be classified. Each time, the substring being classified is marked with a special symbol such as '*'.

3 Computational Complexity of the SCT Algorithms

Let D be the total number of sentences in the training data, L the upper limit on the number of words in a sentence, and V the size of the vocabulary. The worst-case time analysis for both single-symbol and set-membership SCTs depends on whether the supply of training data or the supply of possible questions runs out first. For each of at most L positions in a sentence, at most $4V$ meaningful single-symbol or set-membership questions about this position can be asked along a path. A rough upper bound for the depth of a path is therefore $4*L*V$ (one could establish a tighter upper bound). To obtain a path of this depth, one requires more than $4*L*V$ sentences in the training data; thus, complexity results depend on whether $D < 4*L*V$ or $D > 4*L*V$.

If $D < 4*L*V$, the worst-case time complexity of growing a single-symbol SCT is $O(D^3*L^2*V)$; for growing a set-membership SCT, it is $O(D^3*L^2*V^2)$. Given the reasonable assumption that the number of iterations of the Gelfand-Delp algorithm [5] is a constant (in our experience, convergence has never required more than 4 iterations) these expressions can be divided by D. Then, growing a single-symbol SCT is $O(D^2*L^2*V)$ and growing a set-membership SCT is $O(D^2*L^2*V^2)$. Both single-symbol and set-membership SCTs require $O(D*L)$ time to classify a sentence of maximum length L.

If $D > 4*L*V$, the time complexity of growing a single-symbol SCT is $O(D^2*L^3*V^2)$ and growing a set-membership SCT is $O(D^2*L^3*V^3)$. Under the assumption of a constant number of expansion-pruning iterations, these expressions are $O(D*L^3*V^2)$ and $O(D*L^3*V^3)$ respectively. For both single-symbol and set-membership SCTs, classifying a string of maximum length L is $O(L^2*V)$.

4 Experiments on the ATIS Task

4.1 The system

The ATIS task is an ARPA-sponsored benchmark for speech recognition and understanding [9].

Two aspects of the ATIS task involve pure sentence classification and can be evaluated separately from other components of performance: generation of the set of displayed attributes ($V.A$) and classification of sentences as 'A' or 'D' ($V.B$).

CHANEL is an SCT based system that translates a word sequence into a semantic representation. Before reaching the robust matcher, the sentence is preprocessed by a local chart parser. The robust matcher itself has two parts: the part that generates a list of **displayed attributes** (the database columns and functions of them the user wants to see) and the part that generates the **constraints** (which are made up of one or more frames combined by AND or OR). Each part is built of SCTs. In Nov. 1992 the SCTs in the robust matcher were trained on 3248 class 'A' sentences from the ATIS 2 training data release; in Dec. 1993 they were trained on 5501 class 'A' sentences from ATIS 2, Feb. 1992, Nov. 1992, and ATIS 3. Class 'A' sentences can be interpreted without knowledge of previous sentences.

4.2 The Local Chart Parser

The bottom-up chart parser looks for words or phrases carrying constraints that should be incorporated into the SQL query [8]. These words or phrases are replaced by three-letter symbols in the version of the input sentence sent to the SCT-based robust matcher. The meaning associated with each symbol is stored by the chart parser, since it will have to be recovered to produce the SQL query. For instance, the chart parser would convert the sentence "Delta flights from Boston to Denver serving breakfast" into "AIL flights from CIT to CIT serving MEA", the version seen by the robust matcher. The assignments $AIL = DL$, $CIT_1 = BBOS$, $CIT_2 = DDEN$, $MEA = breakfast$ will be stored until the semantic representation is generated. In the Nov. 1992 CHANEL, the SCT-based robust matcher was entirely responsible for generating the displayed attribute list and for assigning roles to constraints. In the Dec. 1993 version, the chart parser looks for and sometimes finds groups of constraints - for instance, arrival time and date may be grouped with the destination. The resulting role assignments may overrule those made by the SCT component; however, this phase of the chart parser has incomplete coverage.

SHOW ME FLIGHTS FROM BOSTON TO DENVER ⇒ 0

ALL RIGHT WHAT I'D LIKE TO DO IS FIND THE CHEAPEST ONE WAY FARE FROM BOSTON TO DENVER ⇒ 1

I WOULD LIKE INFORMATION ON GROUND TRANSPORTATION IN THE CITY OF ATLANTA FROM AIRPORT TO DOWNTOWN ⇒ 0

SHOW ME ALL THE FLIGHTS BETWEEN DALLAS FORT WORTH AND EITHER SAN FRANCISCO OR DENVER THAT DEPART BETWEEN FIVE AND SEVEN P M ⇒ 0

Figure 1: **Part of Training Data for *fare.fare_id* SCT**

4.3 Choosing the Displayed Attributes

A given query may have any number of displayed attributes. CHANEL contains a forest of SCTs for choosing displayed attributes, each responsible for a particular attribute. The output of each displayed attribute SCT is either 1 (the attribute will be displayed) or 0 (the attribute will **not** be displayed). Each of the displayed attribute SCTs looks at the input word sequence independently - in principle, all SCTs could output 0 (giving an empty list) or 1 (giving a list containing all possible attributes). Fig. 1 shows part of the training data used to grow the *fare.fare_id* SCT. The Nov. 1992 CHANEL contained 106 displayed attribute SCTs, while the Dec. 1993 CHANEL contained 109.

4.4 Assigning Roles to Constraints

In the *constraints* component of the robust matcher, *role assignment* SCTs assign roles to constraints. Properties of the Nov. 1992 constraints component:

- There were 3 role assignment SCTs - AIP (airport name), CIT (city name), and TIM (time);
- Possible values for AIP or CIT - origin, destination, stopover, site served by an airline, or ground transport location;
- Possible TIM values - arrival time or departure time;
- Role assignment SCTs were single-symbol substring classification SCTs (*II.D*).

This version of CHANEL also contained hand-coded metarules that return "NO ANSWER" under the following conditions:

- Local constraint roles clash - e.g., two different CITs in same frame assigned origin role;
- Displayed attributes and constraints clash - e.g., displayed attributes involve the table *ground_service*, but a CIT is classified as destination of a flight.

Properties of the Dec. 1993 constraints component:

- There were 3 role assignment SCTs - one for AIP and CIT, one for TIM, and one for DAT (date) and DAY (day of the week);
- Possible values the same as before, with DAT and DAY classified as arrival, stopover, or departure day;
- Role assignment SCTs are single-symbol substring classification SCTs augmented with 2 new question types:
  1. Questions about whether a given attribute is on the displayed attribute list;
  2. For different codes handled by a single SCT, questions about what the current code is - e.g., one of the nodes in the AIP-CIT tree asks whether code is AIP.

The first new question type helps ensure consistency between displayed attributes and constraints (but requires that the displayed attribute list be generated before role assignment occurs). The second new question type allows constraints with similar semantics (AIP and CIT, DAT and DAY) to be merged safely. These new question types illustrate how SCTs can incorporate application-specific information.

5 Experimental Results

5.1 Preliminary Experiments

106 SCTs for picking displayed attributes grown on a subset of ATIS 2 class A data were tested on a disjoint subset of the same data. On the test data, a "success" occurs only when the chosen set of displayed attributes is precisely identical to the set of displayed attributes found in the canonical ATIS MIN response.

SCTs for displayed attributes can vary along several orthogonal dimensions:

- Trained on a small vs. a large amount of data;
- Trained on data where the local parser has replaced certain substrings by symbols such as CIT or TIM vs. training on the original word sequences;
- "Well-shuffled" vs. "lumpy" training data;
- Single-symbol vs. set-membership SCTs.

5.2 November 1992 and December 1993 ATIS Results

Tables 1 show ATIS class 'A' NL interpretation errors.

System	NL 'A' Error
ATT	7.4
BBN1	9.6
BBN2	16.1
CHANEL (o)	21.7
CHANEL (d)	12.3
CMU	6.0
MIT-LCS	10.0
SICSA	14.7
SRI	14.3
UNISYS	28.6

Table 1: **Dec. 1993 NL 'A' Benchmarks**

Currently, the rules in the context module, which handles 'D' sentences by deciding what information to inherit from semantic representations generated earlier, are all hand-coded. We are exploring automatic learning for some of the rules in this module. In particular, it would be nice to have a module that decided with high accuracy whether a sentence is class 'A' or 'D' ('A' sentences inherit no information).

We have grown on 5816 ATIS 2 sentences (each labelled 'A' or 'D') an SCT that classifies a sentence as either 'A' or 'D', based purely on its word sequence. This SCT is not allowed to ask any questions about the past - it does not even know when a sentence is the first in a scenario (such a sentence must be 'A'). The test results for 2373 ATIS 3 sentences and 701 Nov. 92 sentences are shown in Table 2; ("A => A" gives the number of class 'A' sentences correctly classified as 'A', "A => D" the number of class 'A' sentences misclassified as 'D', and so on). The percentage of correctly classified sentences is surprisingly high. A learning algorithm which was allowed to ask questions about the contents of semantic representations (current and previous) as well as about word sequences might yield performance competitive with that of hand-coded context modules.

Data	$A \Rightarrow A$	$D \Rightarrow D$	$A \Rightarrow D$	$D \Rightarrow A$	% Corr
ATIS 3	1308	814	104	147	89.4
Nov92	411	214	33	43	89.2

Table 2: **'A' vs. 'D' Results for ATIS 2-trained SCT**

6 Discussion

This work makes two original contributions:

1. Semantic Classification Trees (SCTs);
2. The SCT-based Robust Matcher.

The ATIS speech understanding task was used as a testbed for these ideas, but they should be widely applicable in natural language understanding systems.

The motivations for our work were as follows:

- We believe that in the long run, NLU systems whose parameters are learned from data will scale up better, and be more portable to new tasks, than purely hand-coded systems.
- The increasing availability of on-line natural language corpora, and the decreasing cost of computation relative to the cost of programmer time, will also tend to favour the automatic learning approach.

- Given that successful SU systems often incorporate robust matchers that skip irrelevant words, the word patterns learned should permit gaps.
- Decision trees have been successfully applied to other NL problems can be modified to learn patterns with gaps, and are tractable building blocks for a complete system.

The first two points are contentious. For instance, on the ATIS task the best hand-coded linguistic analyzers have out-performed those that carry out automatic learning. However, the ATIS results do not address the trade-off between human effort and performance - in the ATIS community as a whole, far more effort has been expended on rule-based than on learning-oriented systems. Conceivably, performance improvements in a hand-coded analyzer require a steeper increase in programmer-hours than such improvements in an analyzer that learns rules from training data; if so, the automatic learning approach is worth pursuing.

In other areas of NL, approaches with a strong automatic learning component have, in the long run, proved more effective than approaches based on hand-coding rules. Speech recognition provides two examples: the triumph of HMMs over the expert system approach to acoustic modeling [10] and the triumph of statistical language modeling over approaches based on deterministic grammars [6]. A recent book that looks at parsing of English text [1] describes experiments in which state of the art, hand-coded deterministic parsers are applied to randomly selected sentences from newspapers and text corpora. Performance (using a very forgiving measure of structural correctness) varied between 16% and 60% correct. A probabilistic parser whose parameters were learned from training data obtained results varying between 92% and 96% (on different but equally difficult test corpora). *A priori*, there seems no reason why an approach that has been extremely effective at most other levels of language should prove ineffective when applied to semantics.

People who find automatic learning of semantic rules intriguing may, nonetheless, have objections to SCTs in particular. One such objection we have encountered is that SCTs embody regular grammars, which are known to be inadequate as a model for the generation of natural language. This ignores the possibility that some problems in computational linguistics are best approached by treating natural language as regular to a first approximation. Furthermore, the objection applies only to systems made up purely of SCTs - in general, SCTs will be used in combination with other components. For instance, in CHANEL the grammar built into the local parser may contain rules of arbitrary power.

SCTs do have genuine limitations. Since the patterns they learn contain gaps, they will tend to work well in contexts where a few key words and phrases contain most of the semantic information. The presence of deeply nested modifiers and complicated conjunctions will confuse them - they are unlikely to deal adequately with the prose of Henry James. Even on the ATIS data, we encountered some difficulty with modifying clauses.

Class 'D' context-dependent sentences also created difficulties. CHANEL relied on a hand-coded context module to deal with class 'D' sentences; this should be replaced by a module with rules learned from training data. The SCT described in the previous section, which decides whether a sentence is 'A' or 'D', is a step in this direction. We are working on an enhanced type of SCT for the context module, whose questions would ask not only about the current sentence but about elements in semantic representations generated by previous sentences. The context module would consist of a forest of such enhanced SCTs, each making the binary decision whether a given element in a previous representation should be inherited by the current representation or not.

We have presented a new method for learning semantic rules from training data. The rules learned by SCTs are based on regular expressions involving symbols from the lexicon (words or higher-level constituents) and gaps; the questions that can be asked in an SCT node depend on the answers to questions asked earlier. Using our linguistic analyzer, CHANEL, as an example, we have shown how SCTs can form building blocks for an NLU system. They can be used in a wide variety of ways in such a system: in parallel, in series, alone or combined with any number of other components. Future work will explore both new applications of SCTs (we are currently studying their ability to classify messages on the InterNet) and improvements to SCTs (for instance, algorithms yielding better-performing set-membership SCTs).

References

[1] E. Black, R. Garside, and G. Leech, "Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach", Rodopi Press, Amsterdam-Atlanta, 1993.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees", Wadsworth Inc., 1984.

[3] *Proceedings of the 1990 DARPA Speech and Natural Language Workshop*, June 1990, Morgan Kaufmann Inc.

[4] K.S. Fu, "A Step Towards Unification of Syntactic and Statistical Pattern Recognition", *Trans. IEEE PAMI*, May 1986, V. 8, no. 3, pp. 398-404.

[5] S. Gelfand, C. Ravishankar, and E. Delp, "An Iterative Growing and Pruning Algorithm for Classification Tree Design", *Trans. IEEE PAMI*, Feb. 1991, V. 13, no. 2, pp. 163-174.

[6] F. Jelinek, "Self-Organized Language Modeling for Speech Recognition" [11 pp. 450-506].

[7] R. Kuhn and R. De Mori, "Learning Speech Semantics with Keyword Classification Trees", *Proc. ICASSP 93*, V. II, pp. 55-58, April 1993.

[8] E. Millien and R. Kuhn, "A Robust Analyzer for Spoken Language Understanding", *Eurospeech 93*, V. II, pp. 1331-1334, Berlin, Germany, September 1993.

[9] P. Price, "Evaluation of Spoken Language Systems: the ATIS Domain" [[3] pp. 91-95].

[10] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition" [[11] pp. 267-296].

[11] A. Waibel and K.-F. Lee (editors), "Readings in Speech Recognition", Morgan Kaufmann Inc., 1990.