Natural Language Engineering **2** (4) 367–368. Printed in the United Kingdom

© 1996 Cambridge University Press

Explanation-based learning and finite state transducers: applications to parsing lexicalized tree adjoining grammars

B. SRINIVAS

Department of Computer and Information Sciences, University of Pennsylvania
Philadelphia, PA 19104, USA
e-mail: srini@unagi.cis.upenn.edu

(*Received 27 December 1996; Revised 3 February 1997*)

There are currently two philosophies for building grammars and parsers: handcrafted, wide coverage grammars; and statistically induced grammars and parsers. Aside from the methodological differences in grammar construction, the linguistic knowledge which is overt in the rules of handcrafted grammars is hidden in the statistics derived by probabilistic methods, which means that generalizations are also hidden and the full training process must be repeated for each domain. Although handcrafted wide coverage grammars are portable, they can be made more efficient when applied to limited domains, if it is recognized that language in limited domains is usually well constrained and certain linguistic constructions are more frequent than others. We view a domain-independent grammar as a repository of portable grammatical structures whose combinations are to be specialized for a given domain. We use Explanation-Based Learning (EBL) to identify the relevant subset of a handcrafted general purpose grammar (XTAG) needed to parse in a given domain (ATIS). We exploit the key properties of Lexicalized Tree-Adjoining Grammars to view parsing in a limited domain as finite state transduction from strings to their dependency structures.

The training phase of the EBL process involves generalizing the derivation trees generated by XTAG for the training sentences and storing these generalized parses in a database under an index computed from the lexical features of the words of the sentence. In the application phase, an index using the lexical features of the words in the input sentence is computed. Using this index, a set of generalized parses is retrieved from the database created in the training phase. If the retrieval fails to yield any generalized parse then the input sentence is parsed using the full parser. However, if the retrieval succeeds, the output is a generalized parse that associates with each word the elementary tree that it anchors and the elementary tree into which it adjoins or substitutes into – an *almost parse*. The *almost parse* is input to the 'stapler' which instantiates the generalized parse to the sentence. In the full paper, we illustrate how the three key aspects of LTAG (a) lexicalization, (b) extended domain of locality, and (c) factoring of recursion from the domain of dependencies (1) lead to an *immediate* generalization of parses for the training set of sentences,

https://doi.org/10.1017/S1351324997001642 Published online by Cambridge University Press

(2) achieve generalization over recursive substructures of the parses, and (3) allow for a finite state transducer (FST) representation of the set of generalized parses.

Feature Generalization: given an LTAG parse, the generalization of the parse is truly *immediate* in that a generalized parse is obtained by (a) uninstantiating the particular lexical items that anchor the individual elementary trees in the parse, and (b) uninstantiating the feature values contributed by the morphology of the anchor and the derivation process. The feature-generalized parse of a sentence is stored in a database using the POS sequence from the training sentence. In the application phase, the POS sequence of the input sentence is used to retrieve generalized parse(s) which are then instantiated to the features of the sentence. This method of retrieving generalized parses allows for parsing of sentences of the same lengths and the same POS sequence as those in the training corpus. However, in our approach there is another generalization that falls out of the LTAG representation which allows for flexible matching of the index to allow the system to parse sentences that are not necessarily of the same length as some sentence in the training corpus.

Recursive-generalization: auxiliary trees in LTAG represent recursive structures. If there is an auxiliary tree that is used in an LTAG parse, then that tree, along with the trees for its arguments, can be repeated any number of times, or possibly omitted altogether, to get parses of sentences that differ from the sentences of the training corpus only in the number of modifiers. Due to recursive-generalization, the POS sequence covered by the auxiliary tree and its arguments can be repeated zero or more times. As a result, the index of a generalized parse of a sentence with modifiers is no longer a string but a regular expression pattern on the POS sequence and retrieval of a generalized parse involves regular expression pattern matching on the indices.

Finite State Transducer Representation: a *finite state machine* (FSM) provides the most efficient method of performing regular expression pattern matching over the space of index patterns. An index in this representation is a path in the FSM. The generalized parse associated with an index is stored along the path that represents the index. The idea is to annotate each arc of the FSM with the elementary tree associated with that POS and also indicate which elementary tree it would be adjoined or substituted into. This results in an FST representation, given the lexicalized nature and the extended domain of locality of elementary trees because of which lexical dependencies are localized to within a single elementary structure.

Stapler: a stapler is an impoverished parser that is used in conjunction with the FST to generate all possible attachment sites for all modifiers besides instantiating the features of nodes of the trees by term unification.

We have tested the performance of our approach on the ATIS corpus. Using the parses of 300 training sentences, we constructed a FST that had 1162 states and could retrieve a generalized parse for 80% of the 100 test sentences at the rate of 0.3 seconds per sentence. We also obtained a speed-up of 15 using the EBL and stapler when compared to the XTAG parser on the 100 test sentences.

https://doi.org/10.1017/S1351324997001642 Published online by Cambridge University Press