Concept Map

Speech and Language Processing (3rd ed. draft)
Dan Jurafsky and James H. Martin

Chapter 13
Constituency Parsing

CSE 431

Task 3
Group 13 (Section 2)

19101442 Sharnit Saha
19101106 Md. Arban Rahman
18101374 Nuzhat Islam

13.1 Ambiguity

Ambiguity is the most serious problem faced by syntactic parsers where introduced the notions of part-of-speech ambiguity and part-of-speech disambiguation. This chapter introduced Structural Ambiguity, which occurs when the grammar can assign more than one parse to a sentence. Structural ambiguity, appropriately enough, comes in many form. Two common kinds of ambiguity are attachment ambiguity and coordination ambiguity. A sentence has an attachment ambiguity if a particular constituent can be attached to attachment ambiguity the parse tree at more than one place.

13.2 CKY Parsing: A Dynamic Programming Approach

Dynamic programming provides a powerful framework for addressing the problems caused by ambiguity in grammars.

13.2.1 Conversion to Chomsky Normal Form

The CKY algorithm requires grammars to first be in Chomsky Normal Form (CNF). CNF are restricted to rules of the form A → B C or A → w. That is, the right-hand side of each rule must expand either to two non-terminals or to a single terminal. Restricting a grammar to CNF does not lead to any loss in expressiveness, since any context-free grammar can be converted into a corresponding CNF grammar that accepts exactly the same set of strings as the original grammar.

13.2.2 CKY Recognition

CKY recognition consists of filling the parse table in the right way. To do this, we'll proceed in a bottom-up fashion so that at the point where we are filling any cell [i, j], the cells containing the parts that could contribute to the entry.

13.2.3 CKY Parsing

Recognizer can tell us whether a valid parse exists for a given sentence based on whether or not if finds an S in cell [0,n], but it can't provide the derivation, which is the actual job for a parser. To turn it into a parser capable of returning all possible parses for a given input, we can make two simple changes to the algorithm: the first change is to augment the entries in the table so that each non-terminal is paired with pointers to the table entries from which it was derived, the second change is to permit multiple versions of the same non-terminal

13.2.4 CKY in Practice

While the restriction to CNF does not pose a problem theoretically, it does pose some non-trivial problems in practice. Obviously, as things stand now, our parser isn't returning trees that are consistent with the grammar
given to us by our friendly syntacticians. In addition to making our grammar developers unhappy, the conversion to CNF will complicate any syntax-driven approach to semantic analysis

13.3.1 Computing Scores for a Span

The input word tokens are embedded by passing them through a pretrained language model like BERT. Because BERT operates on the level of subword (wordpiece) tokens rather than words, we'll first need to convert the BERT outputs to word representations. One standard way of

Chapter 13: Constituency Parsing

Syntactic parsing is the task of assigning a syntactic structure to a sentence. Parse trees can be used in applications such as grammar checking: sentence that cannot be parsed may have grammatical errors. Parse trees can be an intermediate stage of representation for semantic analysis and thus play a role in applications like question answering.

Recall that a dynamic programming approach systematically fills in a table of solutions to sub-problems. The dynamic programming advantage arises from the context-free nature of the grammar rules

doing this is to simply use the last subword unit as the representation for the word (using the first subword unit seems to work equivalently well). The embeddings can then be passed through some postprocessing layers;

13.3 Span-Based Neural Constituency Parsing

This parser learns to map a span of words to a constituent, and, like CKY, hierarchically combines larger and larger spans to build the parse-tree bottom-up. But unlike classic CKY, this parser doesn't use the hand-written grammar to constrain what constituents can be combined, instead just relying on the learned neural representations of spans to encode likely combinations.

13.3.2 Integrating Span Scores into a Parse

The parser is using the max label for span (i, j) + the max labels for spans (i, k) and (k, j) without worrying about whether those decisions make sense given a grammar. The role of the grammar in classical parsing is to help constrain possible combinations of constituents (NPs like to be followed by VPs). By contrast, the neural model seems to learn these kinds of contextual constraints during its mapping from spans to non-terminals. Also. on span-based parsing, including the margin-based training algorithm

13.4 Evaluating Parsers

The standard tool for evaluating parsers that assign a single parse tree to a sentence is the PARSEVAL metrics. The PARSEVAL metric measures how much the constituents in the hypothesis parse tree look like the constituents in a hand-labeled, reference parse.

13.5 Partial Parsing

A partial parse, or shallow parse, of input sentences information extraction systems generally do not extract all the possible information from a text: they simply identify and classify the segments in a text that are likely to contain valuable information.

13.6 CCG Parsing

Lexicalized grammar frameworks such as CCG pose problems for which the phrasebased methods we've been discussing are not particularly well-suited. To quickly review, CCG consists of three major parts: a set of categories, a lexicon that associates words with categories, and a set of rules that govern how categories combine in context.

13.6.1 Ambiguity in CCG

The difficulties with CCG parsing arise from the ambiguity caused by the large number of complex lexical categories combined with the very general nature of the grammatical rules. While CCG parsers are still subject to ambiguity arising from the choice of grammar rules, including the kind of spurious ambiguity, it should be clear that the choice of lexical categories is the primary problem to be addressed in CCG parsing.

13.6.2 CCG Parsing Frameworks

Since the rules in combinatory grammars are either binary or unary, a bottom-up, tabular approach based on the CKY algorithm should be directly applicable to CCG parsing. Unfortunately, the large number of lexical categories available for each word, combined with the promiscuity of CCG's combinatoric rules, leads to an explosion in the number of (mostly useless) constituents added to the parsing table.

13.6.3 Supertagging

Supertagging is the corresponding task for highly lexicalized grammar frameworks, where the assigned tags often dictate much of the derivation for a sentence. CCG supertaggers rely on treebanks such as CCGbank to provide both the overall set of lexical categories as well as the allowable category assignments for each word in the lexicon.

13.6.4 CCG Parsing using the A* Algorithm

The A* algorithm is a heuristic search method that employs an agenda to find an optimal solution. Search states representing partial solutions are added to an agenda based on a cost function, with the least-cost option being selected for further exploration at each iteration. The A* cost function, f(n), is used to efficiently guide the search to a solution. The f-cost has two components: g(n), the exact cost of the partial solution represented by the state n, and h(n) a heuristic approximation of the cost of a solution that makes use of n.