Speech and Language Processing

by Jurafsky, Martin

13 Constituency Parsing

Parse trees are an intermediate stage in semantic analysis and can play a role in question answering and other such applications.

13.1 Ambiguity

Structural ambiguity occurs when the grammar can assign more than one parse tree to a sentence. There are two common kinds of structural ambiguity: attachment (if a constituent can be attached to the parse tree at more than one place) and coordination (if an adjective can modify one or both constituents conjoined by a conjunction).

CKY is designed to efficiently handle structural ambiguities, and can be augmented by neural methods to effect syntactic disambiguation as well.

13.2 CKY Parsing: A Dynamic Parsing Approach

The advantage that lies in dynamic programming (applied to context-free grammar) is that once a constituent is discovered, we can record it and make it part of any subsequent derivation that needs it. This is the idea that lies at the heart of the Cocke-Kasami-Younger algorithm.

13.2.1 Conversion to Chomsky Normal Form

Being in CNF is a requirement for the application of the CKY algorithm.

Rules that mix terminals and variables are handled by introducing a new variable that covers only the terminal. Rules that produce a single variable (unit productions) can be replaced by substituting the RHS with the RHSs of all the rules whose LHS is that single variable. Rules with RHSs longer than two make use of new variables that spread the longer sequences over several rules.

Therefore, in summary, we can convert a CFG to CNF by: * copying all conforming rules unaltered. * converting terminals in RHSs to dummy non-terminals. * converting unit productions. * making all rules binary and adding them to the new grammar.

13.2.2 CKY Recognition

We will work on the upper-right triangle of a 2D $(n+1) \times (n+1)$ matrix, where the cell at (i, j) contains the set of non-terminals that span positions i to j of the input (indices point to the gaps between words, or fenceposts).

The diagonal contains the part of speech for each word in the input. Each diagonal above it contains constituents that cover all the spans of increasing length in the input. We fill the table in a bottom-up fashion so that the cells containing the parts that could contribute to a certain cell have been filled before it – to do this, we fill it column by column, going from bottom to top for each column. Pseudocode for the algorithm is as follows:

```
function CKY-Parse(words, grammar) returns table
    for j <- from 1 to length(words) do
        for all {A | A -> words[j] \in grammar}
            table[j-1,j] <- table[j-1,j] + A
        for i <- from j-2 down to 0 do
            for k <- i+1 to j-1 do
                for all {A | A -> BC \in grammar and B \in table[i,k] and C \in table[k,j]}
                    table[i,j] <- table[i,j] + A
```

Here, the outer loop iterates over the columns and the middle loop over the rows. In each iteration of inner loop, the constituents under the current cell are checked to see if they form a constituent under the rules of the grammar.

13.2.3 CKY Parsing

However, this algorithm only recognises the input by checking for S in cell $(0, n)$. In order to return all possible parses for an input, we make two changes: augment the entries in the table so each non-terminal points to its children, and permit multiple version of the same non-terminal to be entered.

13.2.4 CKY in Practice

Conversion to CNF can complicate syntactic analysis; the grammar has to be converted back as a post-processing step. The CKY algorithm can be altered to handle unit productions in their original form as well.

13.3 Span-Based Neural Constituency Parsing

The CKY algorithm does not disambiguate among all the possible parses. To solve this, we use neural CKY or span-based constituency parsing, which trains a neural classifier to assign a score to each constituent and combines these scores to find the best tree.

13.3.1 Computing Scores for a Span

Consider the constituent that lies between fenceposts i and j with terminal symbol l, to which we will assign the score $s(i, j, l)$.

We use a word encoder built from a pretrained language model like BERT to obtain y_t (embeddings for each word). We convert each y_t to a value for spans

ending at this fencepost and a value for spans beginning at it, by splitting it into two halves.

A traditional way to represent a span is to then take the different between the embeddings of its start and end and re-concatenating them, thus getting the span vector $v_{i,j}$. The span vector is then prcoessed some more to obtain a score for each variable.

13.3.2 Integrating Span Scores into a Parse

Formally, a tree is represented as a set of spans: $T = \{(i_t, j_t, l_t) : t \in \{1, \ldots, |T|\}\}$. Then the score for a tree is the sum of the scores of its spans.

A simple method to produce the most likely parse is to greedily choose the highest scoring label for each span; although this is not theoretically guaranteed to find the best tree, it often does so in practice.

However, it is more common to use a variation of the CKY algorithm to do this incrementally. For spans of length one, we pick $s_{\text{best}}(i, i+1) = \max_l s(i, i+1, l)$, and for longer spans we use the recursion

$$s_{\text{best}}(i, j) = \max_l s(i, j, l) + \max_k [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)].$$

13.4 Evaluating Parsers

The standard tool for evaluating parsers is the PARSEVAL metric, which measures how much the constituents in the hypothesis tree look like those in a reference tree. Then, we label the constituents as correct or incorrect and define

$$\text{labelled recall} = \frac{\text{no. of correct constituents in hypothesis tree}}{\text{no. of correct constituents in reference tree}}$$

and

$$\text{labelled precision} = \frac{\text{no. of correct constituents in hypothesis tree}}{\text{no. of total constituents in hypothesis tree}}.$$

and report a combination of the two, $F_1 = \frac{2PR}{P+R}$.

Another metric, cross-brackets is sometimes used – the number of constituents for which the reference parse has a bracketing $((AB)C)$ but the hypothesis parse has a bracketing $(A(BC))$.

The canonical implementation of PARSEVAL, called evalb, removes information likely to be grammar-specific and calculates a simplified score.

13.5 Partial Parsing

Often a partial or shallow parse of the input is sufficient for an application. For example, chunking (identifying the non-overlapping segments of a sentence constituting the basic phrases) to find the noun phrases is common.

Since chunked texts lack a hierarchy, simple brackets suffice; for example, $[_{NP}$ The morning flight] $[_{PP}$ from] $[_{NP}$ Denver] $[_{VP}$ has arrived.]. This illustrates the two basic tasks in chunking – segmenting and labelling.

Some guidelines are usually followed: base phrases of a given type do not contain any constituents of the same type, post-head material is excluded (thus eliminating attachment ambiguities).

Chunking is generally done by supervised learning (training a BIO sequence labeller). Chunkers rely on treebanks, extracting syntactic phrases, finding the heads, and including material to its left.

13.6 CCG Parsing

13.6.1 Ambiguity in CCG

In the sentence *United diverted the flight to Reno.*, the ambiguity arises from the choice between rules. However, in CCG, we can associate *to* with distinct categories to reflect the ways it can combine with words.

If *to* has the type $(NP \setminus NP)/NP$, the PP *to Reno* modifies *the flight*. On the other hand, if we assign it $(S \setminus S)/NP$, it modifies the verb phrase *diverted* instead. A third choice is to give *diverted* the type $((S \setminus NP)/PP)/NP$, allowing it to take the PP as an argument.

13.6.2 CCG Parsing Frameworks

If we follow a brute-force parsing approach, the parsing possibilities will shoot up exponentially. Thus we need to assess the likelihood of constituents during parsing; this is called supertagging.

13.6.3 Supertagging

Supertagging is the task corresponding to PoS-tagging in lexicalised grammar frameworks. CCG supertaggers rely on treebanks to provide the overall set of categories and the assignments to each word.

The standard approach is to use supervised ML to build a sequence labeller and find the most likely sequence using a neural sequence model.

13.6.4 CCG Parsing Using the A* Algorithm

The A* algorithm is a heuristic search method that adds search states (representing partial solutions) to an agenda based on a cost function. At each iteration, the least-cost option is further explored; the search terminates when the first complete solution is selected.

The A* cost function ($f(n)$, called the f-cost) has two parts: the exact cost $g(n)$; and a heuristic approximation of the cost of a solution that uses n, $h(n)$.

The accuracy of h determines the effectiveness of A*. Search states correspond to completed constituents.

In order to define a heuristic function, we make the simplifying assumption that the probability of a derivation is just the product of the probabilities of the supertags (ignoring the rules). In order to make a lower cost indicate better derivations, we use negative log; thus $P(D,S) = \sum_{i=1}^{n} -\log P(t_i|s_i)$. This becomes our definition of the inside cost $g(n)$. For the outside cost, we assume that each of the words outside the span is assigned its most probably supertag – this will never overestimate the cost of the derivation. Thus we have the function

$$f(w_{i,j}, t_{i,j}) = \sum_{k=i}^{j} -\log P(t_k|w_k) + \sum_{k \notin [i,j]} \min_{t \in tags} (-\log P(t|w_k)).$$

13.7 Summary

1. Structural ambiguity is a significant problem in parsing. Sources are PP-attachment, coordination ambiguity and NP-bracketing ambiguity.
2. Dynaming programming approaches make use of a table of partial parses to efficiently parse ambiguous sentences.
3. CKY needs the input grammar to be in CNF.
4. Parsers are evaluated with three metrics – labelled recall, labelled precision and cross-brackets.
5. Partial parsing and chunking are methods for identifying shallow syntactic constituents in a text; they are implemented by training sequence models on annotated data.