CS630 Lecture 19:Modeling syntactic structure: context-free grammars
Date: April 11th, 2006
Lecturer: Lillian Lee
Scribes: Ari Rabkin and Victoria Krafft

Today we'll be looking at syntactic structure, specifically as modeled by Context Free Grammars (CFGs).

Contents

1 CFGs

Up until now, we've only talked about structure within the corpus, and ignored structure within the document. It didn't matter what order the words in the document were in; the document was regarded as a bag of words. Today, we'll begin moving away from that.

We'll begin by looking at the structure of a sentence. We can assign a hierarchical (and ordered) constituent analysis. The constituents will be represented as subtrees. For an example of this, see the figure below, which was on the handout in class. The entire sequence of words can be assigned the label S, meaning that it forms a constituent of type "sentence". This is then split up into a noun phrase NP followed by verb phrase VP. (We will be using these syntactic categories rather than the more "functional" subject/predicate categories.)

This structure can also be represented using brackets, where each sequence of words making up a constituent is surrounded by a pair of brackets with a label. Brackets may be embedded, but may not overlap.

The smallest unit that a word will be part of is a preterminal, also known as a part of speech. For example, a noun, verb, or preposition would be a preterminal.

We observe "X-bar-like" patterns, where a phrase XP always has a subconstituent of category X, which we call the "head" of the phrase. The head's features "propagate" to the XP. So, a noun phrase NP will always contain a noun N, and a verb phase VP will always contain some verb V, and if the head V is a past-tense verb, then the VP is also past-tense.

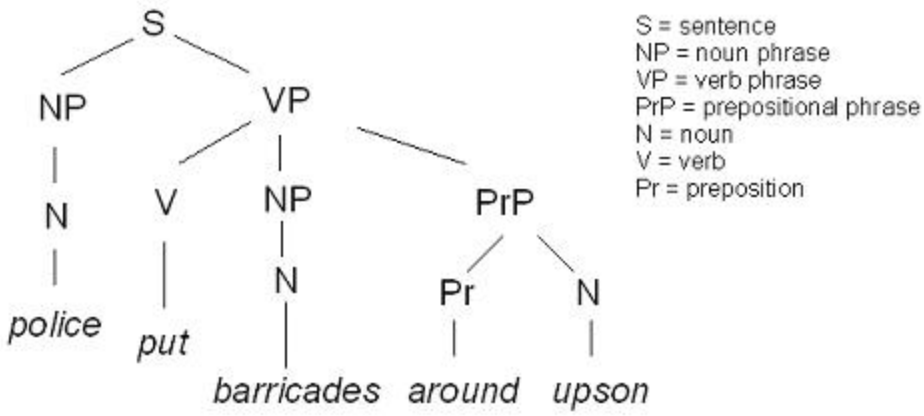

Figure 1: A sentence divided up into its constituents

In general, two constituents with the same label can be substituted for each other. For example, in bracket notation, we have:

$[[\text{police}]_{\text{NP}}\ [[\text{put}]_{\text{V}}\ [\text{barricades}]_{\text{N}}\ [[\text{around}]_{\text{Pr}}\ [\text{upson}]_{\text{N}}\ ]_{\text{PrP}}\ ]_{\text{VP}}\ ]_{\text{S}}$

With substitutability, this can become:

$[[\text{police}]_{\text{NP}}\ [[\text{put}]_{\text{V}}\ [\text{daffodils}]_{\text{N}}\ [[\text{around}]_{\text{Pr}}\ [\text{upson}]_{\text{N}}\ ]_{\text{PrP}}\ ]_{\text{VP}}\ ]_{\text{S}}$

or

$[[\text{police}]_{\text{NP}}\ [[\text{put}]_{\text{V}}\ [\text{barricades}]_{\text{N}}\ [[\text{near}]_{\text{Pr}}\ [\text{upson}]_{\text{N}}\ ]_{\text{PrP}}\ ]_{\text{VP}}\ ]_{\text{S}}$

We do not require these trees to be binary[1]. If we try to create a binary tree using placeholder categories, then problems arise. Consider a "movement-based argument". We will provisionally assume that if parts of a sentence can be rearranged, those parts are meaningful constituents. Examples I(c), I(d) and I(e) below demonstrate the results of this. In I(c), we moved 'barricades' and changed the sentence to the interrogative form, which produces a reasonable sentence in English. In I(d), we did the same thing, moving and changing a prepositional phrase. Example I(e) shows that when binary trees are used, so that "barricades around upson" is presumed to form a constituent, the result does not produce reasonable sentences in English.

Examples:

I (c) : [What] will police [[put] [around upson]]

[1]Contrast the use in formal language theory of the Chomsky normal form.

I (d) : [Where] will police [[put] [barricades]]

I (e) : [What] [where] will police [[put]] / [What] will police [[put]]

Can we use context-free grammars (CFGs) to describe exactly the set of legal syntactic analyses, and in particular, capture important regularities?

A CFG consists of the following elements:

- A finite set of terminals (lexical items such as words)
- A finite set of nonterminals (constituent labels, including a distinguished set of preterminals); this set is disjoint from the set of terminals.
- A distinguished start symbol S
- A finite set of *rewrite rules* or *productions* for the subconstituent expansions that are allowed. For example, we might allow S → NP VP.

Typically preterminal expansions, such as N → police, are stored separately in a lexicon (dictionary). This lexicon specifies, at a minimum, what part of speech each word in the vocabulary is. (Later we will see that much more information can be stored there as well.) Separating the lexicon from the "true grammar" makes it easier to add words to the vocabulary without modifying all the rules.

To generate a tree from a CFG, one does the following:

1. Start with the "degenerate" tree with a single node, labeled with the start symbol S.
2. Expand a leaf nonterminal according to the productions of the CFG.
3. Repeat above step until only terminals remain..

A CFG that generates Ib must contain, at a minimum, the following productions:
S → NP VP
VP → V NP PrP
PrP → Pr N
NP → N
N → police
N → upson
N → barricades
V → put
Pr → around

However, these productions can always be applied regardless of the surrounding context of the nonterminal to be decomposed (hence the name "context-free", so we could generate "upson put upson around upson" as well. Any CFG which generates a tree where a given

nonterminal appears more than once with a different sequence of terminals at the leaves can produce multiple trees, since subtrees with identical root nodes can be replaced with one another.

2 Problems with English

So far, there is no CFG that generates all the legal syntactic structures of English while only generating legal syntactic structures of English (even assuming that such a thing is well-defined). Indeed, there have been arguments made that CFGs are not powerful enough to model all natural-language phenomena. Regardless of the theoretical issues, there are definitely a variety of practical issues when it comes to creating a CFG for English.

1. Category proliferation: Example II on the handout shows several instances of this problem.

   IIa: police [$[\text{informed}]_{\text{V}}$ $[\text{the president}]_{\text{NP}}$ $[\text{that students had hired lawyers}]_{\text{S}'}$ $]_{\text{VP}}$

   This is a legitimate sentence. Therefore, we might imagine that VP $\rightarrow$ V NP S' is an accurate general rule for English (here we are using S' to mean "clause").

   IIb: *[2] police $[\text{informs}]_{\text{V}}$ the president that students had hired lawyers
   In this sentence, we have replaced the verb "informed" with the verb "informs". This demonstrates that the posited rule allows agreement mismatch: "informs" is singular, but "police" is plural.

   IIc: * police informed $[\text{she}]_{\text{NP}}$ that students had hired lawyers

   The NP "the president" has been replaced with the NP "she". This demonstrates that the posited rule allows a case mismatch, which is one instance of a subcategorization error. The NP "she" is a direct object, so it must be in the accusative case ("her"). Note that in English, case errors only happen with pronouns.

   IId: * police $[\text{informed}]_{\text{V}}[\text{rocks}]_{\text{NP}}$ that students had hired lawyers
   Here, the NP "the president" has been replaced with the NP "rocks". Since it is not possible to inform rocks of anything, this rule allows a selectional error: the object of "informed" must be something that can be informed.

   To fix the subcategorization problems, we can add categories. Unfortunately, we need a large number of them. Estimates put the number of subcategorization frames for the English language at > 30 [Gazder et al. '85], to > 1000 [Gross '75]. Potentially, we

[2] By convention, asterisks indicate invalid sentences

would need to indicate several (many?) subcategorization frames for each verb, which we seem to only be able to do by adding more subcategory types (one for each possible set of subcategorization frames a verb could exhibit).

2. Long-distance dependencies: In some sentences, structurally related components are some distance apart. For example, there are problems with filler-gap constructions, like "where will police put barricades"; the prepositional phrase associated with "put" has (a) moved to the front of the sentence, leaving a gap behind, and (b) transformed into a question word "where" (we refer to "where" as the *filler* for the gapped prepositional phrase). These are hard to model with CFGs. The rules below allow us to construct a correct sentence, but they also allow for incorrect sentences:

   S → NP VP

   VP → V PrP

   These rules allow "Police put around upson".

3 Questions

a) In this lecture, the terminals which we've used have been words, separated by spaces. However, in English there are meaningful lexical units that are not separated by spaces. For instance, there are a number of modifier prefixes and suffixes that have well-defined roles and meanings. Give some examples of production rules for structure within words, and comment on how easy it would be to integrate them with an (existing) grammar that generates sentences.

   Ans:

   We have lexicon entries like:
   Core Adj → "American"
   Singular Noun → "rabbit".

   Then we might use rules like:
   Adj → "un-" + Core Adj
   Adj → "anti-" + Singular Noun
   Plural Noun → Singular Noun + "s"

   The above production rules are able to generate words like "rabbits" and "anti-American". Alas, there are awkward design issues that come up when we attempt to combine word-level productions with sentence-level productions. For starters, we're going to need to tag our words with more information to enable the system to handle irregular plurals and so forth.
   Observe that the + sign here indicates concatenation without spaces in between. We're going to need to modify our parser to break words up appropriately (presumably tagging the returned prefixes to indicate that they're attached to the word, and not space

separated). Note that English words in general do not have multiple parsings, and as a result, it is seldom ambiguous whether a final "s" is part of the core word or indicates a plural. There are a few exceptions though. For instance "braces" on teeth (where the "s" is part of the word) versus the plural of "brace". Handling these cases correctly is likely to be messy–one might create new nonterminals for the former case or do some other special trick.

b) From work in the theory of computation, we have the following theorem: Determining whether two arbitrary context-free grammars are equivalent (meaning that they generate the same strings) is undecidable. What impact will this have on NLP tasks?

Ans: The theorem tells us that if different studies propose two different grammars for a language, there is no general technique to determine if the two studies agree with each other or not. While we can determine quickly if the two proposed grammars agree for any given sentence, we cannot determine if there exists a sentence for which they disagree.

This will actually have very little impact on real NLP tasks. Given two grammars, we can test them on a corpus of sample strings and determine if they agree about whether each is or isn't in the language. This will give us a very good sense whether the grammars agree for practical purposes.

Moreover, even if two CFGs do generate the same strings, they may generate different parse trees, which is an important difference between the grammars, from an NLP point of view. The set of strings generated by a grammar isn't the only important thing in the NLP world, and isn't even always the most important.