CS 630 Notes: Lecture 21
Lecturer: Lillian Lee

Notes by Matt Connolly, David Lin, and Danish Mujeeb

April 18^{th}, 2006

1 Interlude: On the importance of syntactic analysis

Today we continue our discussion of tree-adjoining grammars (TAGs). But first, as a reminder, we ask: why are we looking at this topic?

Our motivation is to construct an "elegant" model of syntactic structure (compared to CFGs) and something a little more powerful – at a somewhat higher cost ($O(n^6)$ for the time to parse an n-word sentence).

The point behind studying models of syntactic structure is expressed in the following **claim:** Documents or speech cannot be fully understood without (some) analysis of their structure – although there is a lot of information available just in a bag-of-words representation.

A canonical example of why we need this structural information, the pp-attachment problem, is exemplified by the sentence "show me all flights on Tuesday". The default or standard assumption is that "on Tuesday" modifies "flights". However, the "unexpected" interpretation, in which "on Tuesday" describes when the showing is to take place, is also legal and at least somewhat plausible. (On the other hand, there is apparently no reasonable interpretation in which "flights on" is treated as a constituent.) Thus, we need to know which structures are legal and which are illegal; also which are preferred or "dispreferred".

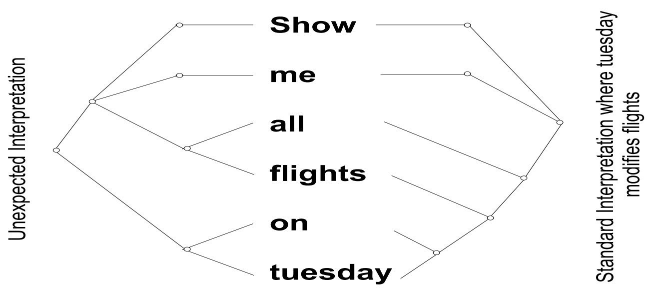


Figure 1: Two grammatical interpretations of "show me all flights on Tuesday".

A more baroque example:

I saw her duck with a telescope

Possible interpretations:

- telescope = mine, duck = verb
- telescope = mine, duck = noun-animal
- telescope = mine, duck = noun-cloth/material
- telescope = duck's, duck = noun
- saw = verb, duck = dead

However, not all combinations are possible:

- telescope = hers, duck = verb .... **YES**
- telescope = hers, duck = noun .... **NO**

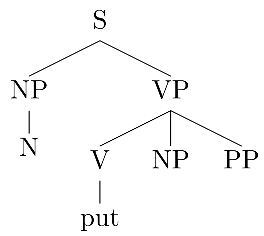
So there is some structure involved in our intuitive analysis of this sentence's meaning that rules out some combinations.

2 TAGs vs. CFGs

Recall that in feature-based context-free grammars some redundancy was observed. In the structure **VP → V-wants-an-NP-and-PP NP PP**, the first element (**V-wants-an-NP-and-PP**) specifies two arguments ("put" being such a verb). But the arguments that follow also specify two arguments by their very presence!

TAGs solve this problem by putting all the structural requirements *in one tree structure*. Suppose we have the following initial trees:

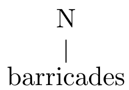
The original α_{put}:



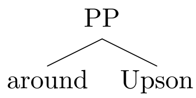
α_{np}:

α_{pol}:

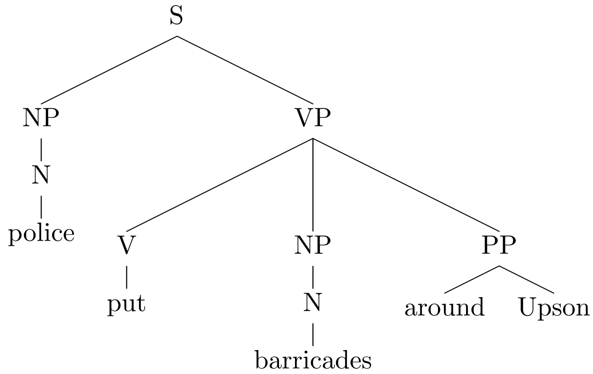
α_b:



α_{loc} (for brevity):



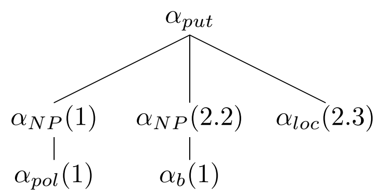
(In practice, the nodes of these trees may be supplemented with feature information. For example, node 1 of α_{put} might be labeled "NP (animate)" and node 2.3 specified as "PP (location)" to force an appropriate choice.) We represent substitutions (into initial trees only) using a derivation tree. In this example, we can create the following derived tree:



via the following operations:

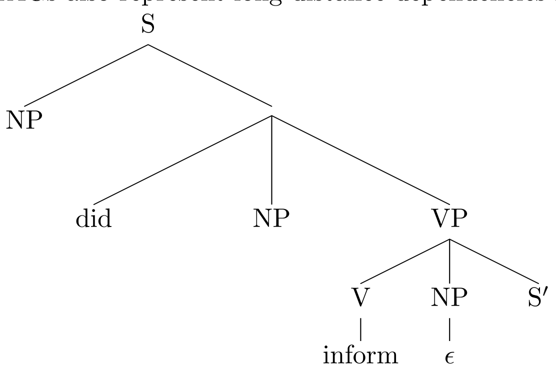
(put α_{pol} into α_{np} node 1) into α_{put} node 1
(put α_b into α_{NP} node 1) into α_{put} node 2.2
(put α_{loc} into α_{put} node 2.3

(The proper order of operation is left-associated, as indicated by the parentheses.) These operations are represented by the following *derivation* tree:



So far TAGs are weakly, but not strongly, equivalent to CFGs. That is, they will produce the same sets of sentences, but can produce sets of trees that CFGs cannot produce.

TAGs also represent long-distance dependencies in a concise manner. For example:



This single tree includes both the filler and gap, and so there is no need to traverse different grammar trees and rules to pass information between them; we can simply specify in this single tree that the noun phrase at node 1 must be animate and accusative (and a pronoun).

3 Adjunction

3.1 The issue

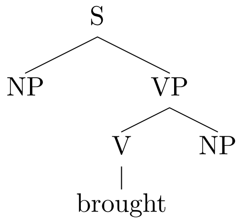
We still have a problem, however. Initial trees can specify all the required arguments for lexical items as leaves where substitution needs to occur (i.e., leaves with non-terminal labels). But consider the following example:

I [brought [cookies]$_{NP}$]$_{VP}$

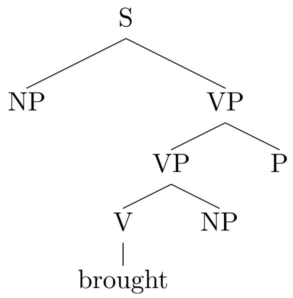
One can't simply say "I brought", so the NP argument is required. In the case of:

I [brought cookies [to the picnic]$_{PP}$]$_{VP}$

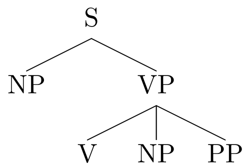
the phrase "to the picnic" is optional. How do you model optional arguments in TAGs? The same initial tree can't be used; in



there is no place to hook on the prepositional phrase. This suggests that we would need another initial tree, such as



or

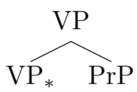


Will we face a combinatorial explosion of trees for all possible combinations of optional arguments?

We can avoid this outcome by introducing a new operation for modeling modification, **adjunction**. This also necessitates a new type of elementary tree, the **auxiliary tree**.

3.2 Auxiliary trees

An auxiliary tree contains a single, distinguished foot node (indicated by *) that bears the same label as the root:



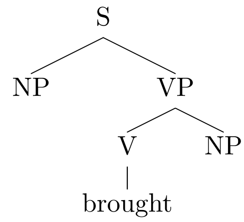
Previously we used α to label our initial trees, so auxiliary trees will be identified by β – thus, β_{pp} for the above example.

3.3 Adjunction

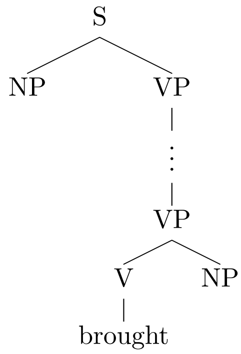
The adjunction operation splices an auxiliary tree into a non-leaf node of an elementary tree. (The "non-leaf" clause distinguishes this from substitution. Also, substitution is typically used for required arguments as opposed to optional modifiers.)

Example: adjoin β_{pp} into $\alpha_{br}(2)$:

α_{br}:

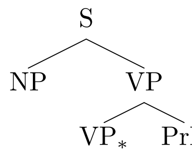


Procedure:



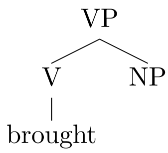
(The tree is "split" at node 2.)

→



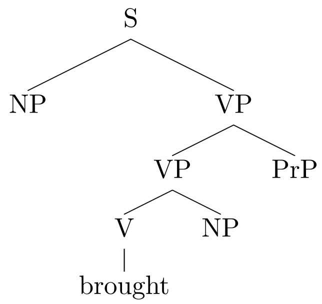
("Substitute" β_{pp} at temporary-leaf node 2.)

+



(The subtree of α_{br} that was at node 2 needs to be attached somewhere here.)

$\rightarrow$



(The displaced subtree of α_{br} is "substituted" into the foot of β_{pp}.)

Now a modifying prepositional phrase can be added at node 2.2 of the (current) derived tree. However, by our ordering rules, the modifying PP (e.g., "to the picnic") should actually have been added to β_{pp} *before* β_{pp} is adjoined into α_{br}.

Note that most initial trees are "centered around" individual lexical items and would encode lexically-based information. As such, despite the fact that we are using large structural units, in a sense TAGs are actually like a bag-of-words representation – at least compared to (vanilla) CFGs.

Not only are TAGs are "good" for syntactic modeling, but they turn out also to be good for semantic modeling! To see this, let's consider the example of idioms, which are "fixed" phrases with non-compositional meanings. For instance, the phrase "Bob kicked the bucket" offers two distinct interpretations: Bob died, or Bob literally kicked a bucket or pail. Generally speaking, modification is not allowed for idioms: "Bob kicked the red bucket" does not convey the same idiomatic meaning. Only very limited changes are allowed, e.g. "Bob kicked the proverbial bucket" – hence the description of idioms as "fixed". The "non-compositional" aspect acknowledges the fact that an idiom's meaining does not seem to be built up out of the meanings of the sub-items (words or phrases) comprising the expression.

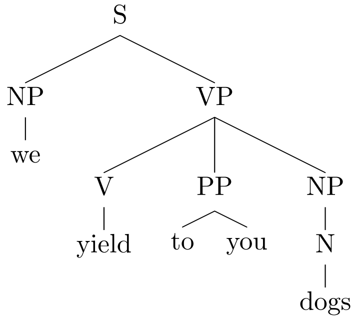
We will continue this train of thought in the next lecture.

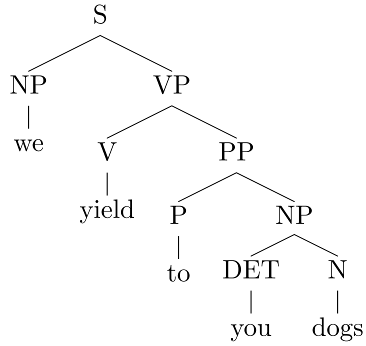
4 Questions

1. Construct two plausible analysis trees for the following sentence:

   We yield to you dogs

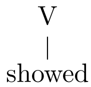
solution:



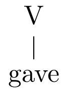


2. Given the following initial trees α_{NP}:

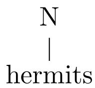
α_{V1}:



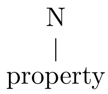
α_{V2}:



α_h:



α_{prop}:

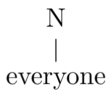


α_g:

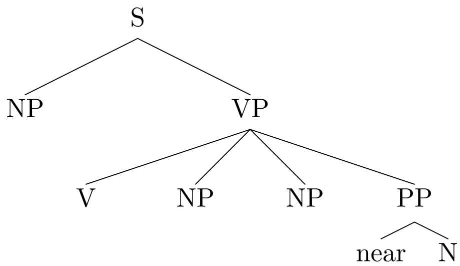
α_{na}:

α_f:

α_e:



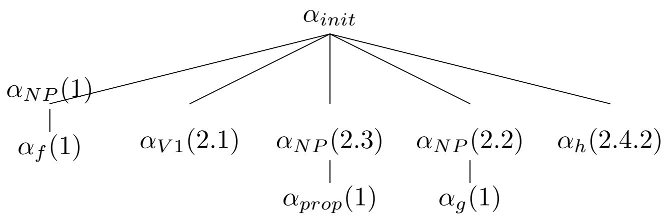
α_{init}:

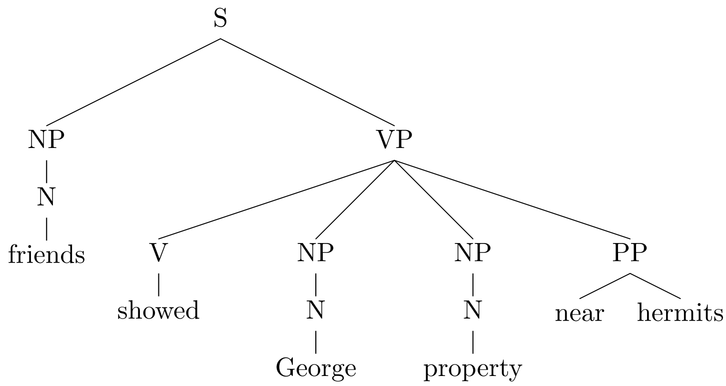


Construct a meaningful sentence, showing the substitution process in shorthand, the derivation tree, and the resulting derived sentence. (Note: there is more than one possible solution! Also, from a linguistics point of view, α_{init} is not entirely sound because the verb is not instantiated. This theoretically allows the insertion of verbs that do not conform to the pattern of arguments presented – e.g., "die".)

(Possible) solution:

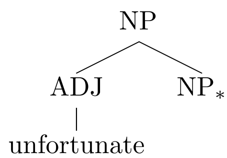
- Put (α_f into α_{NP} node 1) into α_{init} node 1
- Put (α_{V1} into α_{init} node 2.1
- Put (α_g into α_{NP} node 1) into α_{init} node 2.2
- Put (α_{prop} into α_{NP} node 1) into α_{init} node 2.3
- Put (α_h into α_{init} node 2.4.2



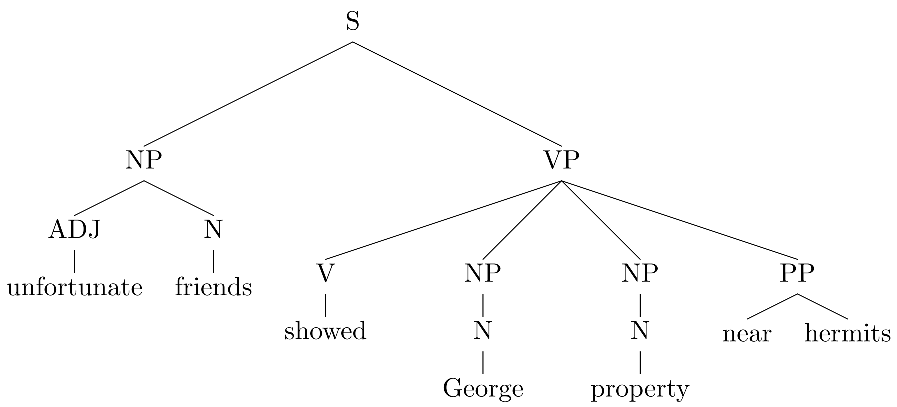


friends showed George property near hermits

3. Adjoin the following auxiliary tree into the tree derived in the previous problem:



Again there is more than one solution. Here is one possibility:



unfortunate friends showed George property near hermits

4. Consider the following sentence fragments that we want to describe with a Tree Adjoining Grammar:

   - stops for Continental 1046
   - flights from Australia to England
   - arrival in San Jose before noon from Vancouver
   - flights serving dinner
   - planes arriving within an hour

   What auxiliary trees would we need to represent these sentence fragments? For simplicity, let's only consider auxiliary trees with a single parent node and two children (though auxiliary trees can be larger). Recall that under this simplification, an auxiliary tree is of the form [Label1* Label2]$_{Label1}$. Also, recall that a verb such as "serving" or "arriving" is considered a gerund, and a verb phrase containing a gerund has a different function in a sentence than a verb phrase with a normal verb.

   Answer:

   At first, it seems like two auxiliary trees are needed to represent chains of prepositional phrases from nouns:

   [NP PP]$_{NP}$ i.e. [stops$_{NP}$ [for Continental 1046]$_{PP}$]$_{NP}$
   [PP PP]$_{PP}$ i.e. [[from Australia]$_{PP}$ [to England]$_{PP}$]$_{PP}$

   However, the single auxiliary tree [NP PP]$_{NP}$ suffices since we can apply it multiple times in succession to produce a chain of propositional phrases, i.e., [[flights$_{NP}$ [from Australia]$_{PP}$]$_{NP}$ [to England]$_{PP}$]$_{NP}$

   In addition, we need:

   [NP GerundVP]$_{NP}$ i.e. [flights$_{NP}$ [serving dinner]$_{GerundVP}$]$_{NP}$

5. Recall this ambiguous sentence:

“Show me all flights on Tuesday.”
[Show me [all [flights [on Tuesday]]]]: correct interpretation
[[Show me [all flights]] [on Tuesday]]: incorrect interpretation

What is one augmentation (besides adding features) we could make to our CFG model of syntactic structure that would give us the *correct interpretation* of this sentence? Assume we have a large corpus of examples from sentences related to air travel, and we consider a *correct interpretation* of the sentence to be that interpretation that is the most *probable* from the sets of productions that can produce this sentence in our grammar. Thus, instead of being concerned with whether our CFG allows productions that are not grammatical (from an English standpoint), we are now concerned with whether our CFG assigns low probabilities to those productions.

Answer:

One simple augmentation of the CFG is called a Probabilistic Context-Free Grammer (PCFG). In a PCFG, we not only keep track of a set of productions allowed by the grammar, but we also assign probabilities to each production. Because we have a corpus of examples, we can learn these probabilities of productions from the corpus for the PCFG such that finding the most probable set of productions that produced this sentence also determines the correct interpretation of this ambiguous sentence. (More on this topic in later lectures.)