Distribution of Words in Sentences: N-grams, Phrase Structure Syntax and Parsing

Adam Meyers
New York University

Outline

- N-Gram Analysis of English
- Context Free Grammar for part of English
- CFG as model of linguistic concepts?
- Generators, Recognizers and Parsers
- Heads & Subcategorization
- Syntactic Frameworks Similar to CFGs
- A More Thorough English Grammar
- Chunking: Simpler than Parsing

Word/Sentence Distribution?

- How do we predict sequences of words? How do we distinguish sentences from non-sentences?
- **Statistically** – N-gram probabilities predict the occurrence of words, based on the occurrence of N-1 previous words, e.g., ***the*** follows the word ***in*** about 28% of the time that ***in*** occurs (bigram model). Models distinguish more and less likely sentences.
- **Syntactically** – Phrases are sequences of words that form equivalence classes based on how all phrases fit together.
  - ***The giant chicken from Brazil*** and ***John's homework assignment*** are both Noun Phrases (NPs)
  - Replacing an NP in a well-formed sentence with another NP will probably result in a well-formed sentence.
  - A sentence is a type of phrase.

Part 1

Statistical Language Models

A Statistical Language Model

- Probability distribution over sequence of words (or other linguistic units)
- Used to rank word sequences by likelihood
  - Rank multiple output of Machine Translation, Language Generation, Voice Recognition, Spelling Correction etc.
  - Assume correct answer is Highest ranked output
- Can be a component in other NLP systems
- Generalized in Various ways
  - POS tagger (discussed earlier in the term)
  - N-grams of character sequences, phoneme sequences, etc.

Training vs Dev/Test Corpus

- Statistics on words are derived from training corpus (e.g., the Brown Corpus)
- Statistics used to predict occurrence of words in other corpora (anything)
- Assume that training corpus is representative
  - Can be a flawed assumption
- After an explanation, we explore unigram and bigram models using python code (and NLTK)
  http://cs.nyu.edu/courses/fall21/CSCI-UA.0480-057/bigram_test.py

Unigrams

- Every word, instance of punctuation, and select other items are tokens
- There are 16 tokens in this sentence, including the period
  - *A fact about the unicorn is the same as an alternative fact about the unicorn.*
- The counts of these words in the Brown Corpus using NLTK
  - *a 23195 fact 447 about 1815 the 69971 unicorn 0 is 10109 the 69971 same 686 as 7253*
  - *an 3740 alternative 34 fact 447 about 1815 the 69971 unicorn 0 . 49346*
- Probability of each token chosen randomly (and independently of other tokens)
  - This is called the unigram probability
  - *a 0.02 fact 0.000385 about 0.00156 the 0.0603 unicorn 0.0 is 0.00871 the 0.0603*
  - *same 0.000591 as 0.00625 an 0.00322 alternative 2.93e-05 fact 0.000385 about 0.00156*
  - *the 0.0603 unicorn 0.0 . 0.0425*
- Converting counts to unigram probabilities
  - count/total_words ≈ probability
  - Assumes that (Brown) corpus is representative of future occurrences

A Unigram Model of a Sentence

- Unigram probability of sentence = product of probabilities of individual words.
- If 1 word has probability of 0, than the probability of the sentence is 0, **unless we model Out-of-Vocabulary (OOV) items differently.**
- One OOV model: assume words occurring once (or less than N times) are OOV and recalculate counts, e.g., *unicorn* now has a non-zero probability
- New Unigram Probabilities:
  - a 0.02 fact 0.000385 about 0.00156 the 0.0603
  - unicorn 0.0135 is 0.00871 the 0.0603 same 0.000591
  - as 0.00625 an 0.00322 alternative 2.93e-05
  - fact 0.000385 about 0.00156 the 0.0603
  - unicorn 0.0135 . 0.0425

Bigrams

- Bigram = probability of $word_N$, given $word_{N-1}$
  - bigram(the,same) = count(the,same)/count(the)
  - count(the,same) = 628
  - count(the) = 69,971
  - bigram_probability = 628/69971 = 0.00898
  - If the previous word is out of vocabulary, the bigram probability will be:
    - count(*oov*,current_word)/count(*oov*)
- Additional steps
  - Include probability that a word occurs a the beginning of a sentence, i.e., bigram(the,START)
  - Include probability that a token occurs at the end of a sentence, e.g., bigram(END,.)
  - Include non-zero probability for case when an unknown word follows a known one (and vice versa).
- Backoff Model
  - If a bigram has a zero count, “backoff” (use) the unigram of the word
    - replace bigram(current_word,previous_word) with unigram(current_word)

NLTK bigram probability of sample sentence

start_end a 0.0182 a fact 0.000388 fact about 0.00447

about the 0.182 the *oov* 0.0293 *oov* is 0.00485

is the 0.0786 the same 0.00898 same as 0.035 as an 0.029

an alternative 0.00241

alternative fact 0.000385 (**Backing off to unigram probability for fact**)

fact about 0.00447 about the 0.182 the *oov* 0.0293

oov . 0.0865 . *start_end* 1.0

Total = product of the above probabilities = **1.12e-30**

Do Walk Through

- python3 bigram_test.py

Trigrams, 4-grams, N-grams

- Trigram Probability
  - Prob(3 token sequence | first 2 tokens)
  - $\frac{count(w-2, w-1, w)}{count(w-2, w-1)}$
  - count(*the*, *same*, *as*)/count(*the*, *same*)
- 4-gram Probability
  - Prob (4 token sequence | first 3 tokens)
  - $\frac{count(w-3, w-2, w-1, w)}{count(w-3, w-2, w-1)}$
  - count(*the*, *same*, *as*, *an*)/count(*the*, *same*, *as*)
- N-gram Probability
  - Prob(N token sequence | N-1 tokens)
  - $\frac{count(w-(n-1), \ldots, w-3, w-2, w-1, w)}{count(w-(n-1), \ldots, w-3, w-2, w-1)}$

The probability of most well-formed sentences is 0

- Many 16 grams do not occur in Brown
  - *A fact about the unicorn is the same as an alternative fact about the unicorn.*
- Some sentences don't occur in any other corpus
  - **Don’t want a model based on a known finite set of all sentences**
- Words (*unicorn*) and N-grams (*the*, *unicorn*) have probability 0 if not in training corpus
- Idealizations are needed to model sentences:
  - Probability of sentence = Some combination of N-grams, where N is a small number (2, 3 or 4)
  - Missing (Out-of-Vocabulary or OOV) words handled specially
  - Missing N-grams handled specially, e.g., backoff
- Include beginning/ending of sentences in model

Markov Assumptions

- Unigram Model: Probability of words are independent of each other
- Bigram Model: Probability of a word depends only on previous word
- Trigram Model: probability of a word depends only on previous two words
- N-gram Model: probability of a word depends only on previous N-1 words
- Probability of a sentence = Product of Probability of words

OOV Model

- Example Model for OOV words
  - Words occurring N or fewer times (N=1) in Training Corpus
    - Not counted as separate words
  - Treated as instances of one word *OOV*
  - Treat any word in test/dev corpus that was not seen in the training corpus as an instance of *OOV*
  - This is what we did in bigram_test.py
- Other models exist, but not discussed here
  - Note problem with current OOV model: ***alternative*** has lower probability than ***unicorn***, even though ***alternative*** actually occurs more than once in the corpus and ***unicorn*** does not occur even once.

Summary of bigrams

- If bigram is not found in training corpus, use unigram probability (implemented)
  - In trigram model, backoff to bigram model, then to unigram model, ..
  - In 4-gram model, backoff to trigram, then bigram, then unigram
  - Etc.
  - This is a simplification of backoff model – see J & M for more details
- If word not found in training corpus or word count = 1, use OOV probability
- For example: .01 is the probability of OOV assuming:
  - The Brown Corpus, treating the 16K items found only once as instances of OOV
- Language models are used to rank possible output
- They are successful if better output gets higher scores, even if all scores are low
- Example, several candidate translations are output by a Machine Translation System
  - They are ranked according to some N-gram model and this ranking is used to choose the best translation.
  - I am currently using a character-based (not word-based) N-gram model to predict whether some text is “garbage” (tables, bibliography, etc.) and should not be processed by the Termolator program

Summary of N-gram Model

- N-gram Language Models approximate the probability of a string of words
  - Based on Markov assumptions (idealizations)
  - Without Markov assumptions, many sentences would have zero probability
- Sentences typically have low probabilities
- Applications assume that better-formed English sentences have higher probabilities
- Such assumptions are typical of Statistical NLP

N-gram Readings/Exercises

- Jurafsky and Martin: Chapters 4.1–4.4
- Other sections of chapter 4 provide refinements to handle back-off, smoothing, etc.
- Review NLTK, chapter 1, section 3
  - http://www.nltk.org/book/ch01.html (then find section 3)
  - Analyze the bigram_test.py program in light of this section
    - https://cs.nyu.edu/courses/fall23/CSCI-UA.0480-057/bigram_test.py

Part 2

Phrase Structure Model

Phrase Structure Model of Language

- Possible sentences in a language are modeled by a set of context free phrase structure rules.
- Non-Terminals represent phrases, sequences of 1 or more terminals
- The terminals are parts of speech
- We will also include “Preterminal” rules
  - Rules that map terminals to words (via dictionary lookup)
  - This is a way of separating the grammar from the lexicon.

English'
A 'Model' of a Subset of English

- N = {S, VP, NP, POSSP, PP}
- T = {N, P, D, POSS)
- S=Sentence
  - English' only handles declarative tensed clauses and ignores other kinds of sentences
- words ={the, a, this, that, food, clam, discussion, group, table, room, Bill, Mary, sincerity, knowledge, redness, of, about, on, in, 's, ate, saw, had, put}
- R = Rules listed on next slide

Grammar for English'

- S → NP VP
- VP → V(erb)
- VP → V(erb) NP
- VP → V(erb) NP PP
- NP → POSSP N(oun) PP
- NP → POSSP N(oun)
- POSSP → NP POSS
- PP → P(reposition) NP
- NP → D(eterminer) N(oun)
- NP → N(oun)

Lexicon for English'

- D → the, D → a, D → this, D → that,
- N → food, N → clam, N → discussion, N → group, N → table, N → room, N → Bill, N → Mary, N → sincerity, N → knowledge, N → redness
- P → of, P → about, P → on, P → in, P → with
- POSS → 's
- V→ ate, V→ saw, V→ had, V→ put

A Random Sentence Generator

- Algorithm
  - Set Output to S
  - Repeat Until Output Contains Only Words
    - Replace the 1st Element Q in Output that is not a word
      - If Q is a NonTerminal
        - » Randomly choose a rule Q → α, where α is a string of terminals and/or nonterminals
        - » Replace Q with α
      - If Q is a Terminal, replace it with a word from the lexicon that is of class Q
- What kinds of objects get generated?
  - Are they all well-formed?
  - If ill-formed, how are they ill-formed?

Implementation of Generator

- random_sentence3.py (and words.py)
- Slightly different (bigger) grammar
  - Variable ***rules*** set to a list of lists
    - 1st item in each list is a nonterminal
    - Remaining items are possible expansions of nonterminal
    - Example: ***DetP --> (NP pos) | (determiner)***
  - The file: words3.py assigns POS to words
- Possible expansions for each symbol stored in table
- generate_random_phrase (optionally set **full_trace=True**)
  - Initially, start symbol (S) is the only element in stack
  - repeat until stack empty
    - remove top item
      - if word, add to result
      - Otherwise, expand and add to stack (left most item on top)
        - » Expand Nonterminal (left-hand side of rule)
          - Choose randomly from possible right-hand sides
        - » Expand terminal via dictionary look-up (random among words with given POS)

Evaluating Strings of Terminals

- A formal language is a set of strings of symbols
- Members of English' and English
  - ***Mary ate food***
- Member of English', but not member of English
  - ***Clam saw the Mary***
  - English' is an imperfect model of English
- Semantically ill-formed
  - ***The redness ate sincerity***
- Not a member of English' or English (Word Salad)
  - ***Clam discussion the the knowledge***

Phrase Structure Typically Models the Distribution of Words and Sequences

- Most linguistic theories assume ways of testing if 2 units belong to the same category or if a sequence forms a unit (or constituent). To some degree these syntactic units correspond to semantic ones.
  - **Important for modeling meaning, e.g., who did what to whom**
- There is some variation among accounts as to whether most, some or any of these tests are being modeled by the grammar
- Alternate view: the grammar is adequate iff all and only the strings of the language can be generated by the grammar. This may not necessitate that constituents or category labels model anything linguistic or semantic.
  - Example: It can be useful to convert the grammar to Chomsky normal form (CNF), where all right hand sides of rules have at most 2 constituents, e.g.,
    - Standard Phrase Structure: NP → Det Adj N
    - Example of Equivalent 2 CNF rules: (a) NP → Det N-bar; (b) N-bar → Adj N

    No guarantee that CNF rules model semantic units. Similarly, some linguistic theories impose templates on phrases and do not attempt to model semantic units.

Same Category Tests

- If 2 units A and B have the same phrase label (or POS), then exchanging A for B or B for A, should not change whether a sentence is syntactically well-formed.
  - *NP read the book* | NP ∈ {*Mary, The pianist with a hat, She, The uncooked eggplant, …*}
  - *They VP and VP* | VP ∈ {*saw a movie, wrote poetry, are politicians, ...*}
- Two words that participate in the same inflectional paradigm are probably the same part of speech
  - *{kill,kills,killed,killing}; {love,loves,loved,loving}*
  - *{big,bigger,biggest}; {silly,sillier,silliest} …*
  - *{house, houses}; {nose, noses} …*

Constituency Tests

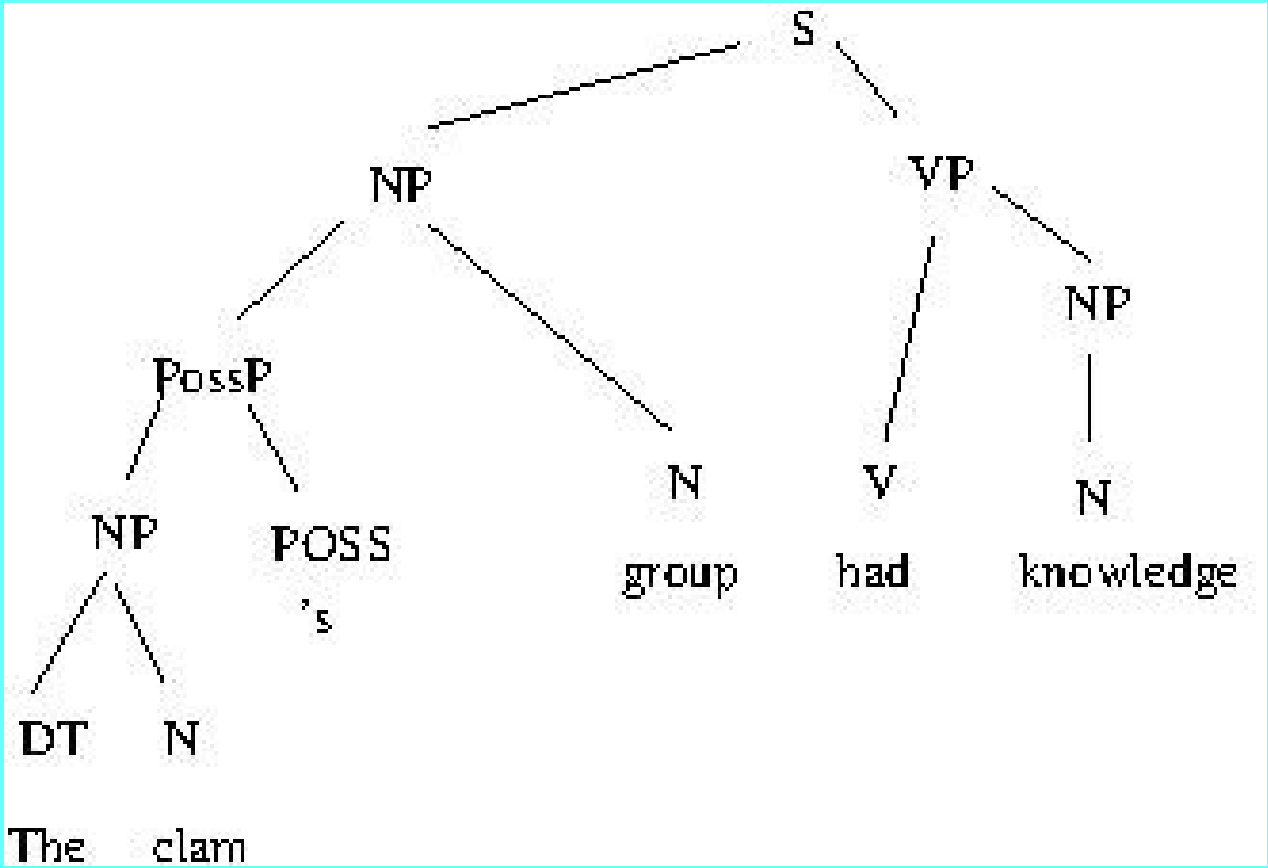
- Sequences behave as units across corresponding sentences.
  - *[In this story], there are 3 pigs ↔ There are 3 pigs [in this story]*
  - *The clam scared [the tourist with a wig] ↔ [The tourist with a wig] was scared of the clam.*
- Only units can conjoin; should be of the same type
  - *[eat] and [drink]; [read books] and [see movies]; [in the house] and [on the porch]*
  - Exceptions (with explanations):
    - *She is a thief and extremely wealthy*
    - *He is angry and in a bad mood*
    - *John did eat and Mary did not eat dinner*
- Units can occur as fragments, e.g., answers to questions
  - *What do you want?* ***A toothbrush with golden bristles***
- Anaphora resolution
  - *Fish really do* ***eat their children****. Humans wouldn't do* ***that****.*

Bracketing Representation of Linguistic Tree

(S (NP (POSSP (NP (D the) (N clam))
(POSS 's)
(N group))
(VP (V had)
(NP (N knowledge)))))

- Commonly used for treebanks and parsers

Equivalent Tree Representation



Sample Parsing with NLTK

- sample_parsing_nltk.py

Sample Sentences to Manually Parse

- *The ice cream truck ended up in New Jersey.*
- *Mary said that she knew about linguistics.*
- *The glass fell before it broke.*
- *The book and the pencil rest quietly on the table.*
- *Donald, the most powerful person in the world, likes french fries with chocolate sauce.*

Chomsky Normal Form (CNF)

- Subtype of Context-free Phrase Structure rules
  - XP → YZ ## Nonterminal → 2 Nonterminals
  - XP → x ## Nonterminal → 1 terminal
  - XP → ε ## Nonterminal goes to empty string
- Any Context-free grammar can be converted to CNF
- Some parsing algorithms (e.g., CKY) require CNF grammars
  - CKY combines, at most, 2 constituents at a time

Convert English' NonTerminal Rules to CNF

- Replace ***VP → V NP PP*** with 2 rules:
  - ***VP → VG PP***
  - ***VG → V NP***
- Replace: ***NP → POSSP N PP*** with 2 rules:
  - ***NP → NG PP***
  - ***NG → POSSP N***
- NG = Noun Group and VG = Verb Group

Approaches to Converting Any Context Free Grammar to CNF

- X → Y Z Q
  - X → A Q
  - A → YZ
- X → Y Z Q R
  - X → AB
  - A → YZ
  - B → QR

- X → Y Z Q
  - X → Y A
  - A → Z Q
- X → Y Z Q R
  - X → YA
  - A → ZB
  - B → QR

Example Chomsky Normal Form

- Example Long Rule: NP → DT JJ JJ NN NN
- Example CNF Rules:
  - NP → DT NG_1
  - NG_1 → JJ NG_1
  - NG_1 → NG_2
  - NG_2 → NN NG_2
  - NG_2 → NN
- Above rule generalizes (allowing multiple NNs or JJs). More NP over-generalizations on midterm exam is OK, even this:
  - NP → DT NG1
  - NG1 → JJ NG1
  - NG1 → NN NG1
  - NG1 → NN

CKY Parser : Tokenization

- Tokens and positions between tokens:
  - 0 ***the*** 1 ***mean*** 2 ***duck*** 3 ***quacked*** 4 ***menacingly*** 5
- Start/End numbers represent subsequences
  - [0,1] = ***the***, [1,2] = ***mean***, [2,3] = ***duck***
  - [0,2] = ***the mean,*** [3,5] = ***quacked menacingly***
  - [0,3] = ***the mean duck,*** [1,4] = ***mean duck quacked***
  - [0,5] = ***the mean duck quacked menacingly***
  - Etc.

CKY Parser: Triangular chart representing all spans

	The	mean	duck	quacked	menacingly
	1	2	3	4	5
0	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]
1		[1,2]	[1,3]	[1,4]	[1,5]
2			[2,3]	[2,4]	[2,5]
3				[3,4]	[3,5]
4					[4,5]

Grammar for
the mean duck quacked menacingly

Context Free Rules

- S(entence) → NP VP
- VP → V(erb) AdvP
- VP → V(erb)
- NP → Det NG
- NG → Adj NG
- NG → Noun
- AdvP → Adv(erb)

Lexicon

- Det: the
- Noun: duck, mean
- Adj: mean
- Verb: quacked, duck
- Adverb: menacingly

CKY: Apply lexicon and rules of the form X → x covering single words

	The	mean	duck	quacked	menacingly
	1	2	3	4	5
0	[0,1] DT	[0,2]	[0,3]	[0,4]	[0,5]
1		[1,2] N, Adj, NG	[1,3]	[1,4]	[1,5]
2			[2,3] N,V, NG,VP	[2,4]	[2,5]
3				[3,4] V,VP	[3,5]
4					[4,5] Adv,AdvP

CKY: Apply rules of the form X → YZ

If Y = [a,b] & Z = [b,c], then X = [a,c]

	The	mean	duck	quacked	menacingly
	1	2	3	4	5
0	[0,1] DT	[0,2] NP	[0,3] NP,S	[0,4] S	[0,5] S
1		[1,2] N, Adj, NG	[1,3] NG	[1,4]	[1,5]
2			[2,3] N,V, NG,VP	[2,4]	[2,5]
3				[3,4] V,VP	[3,5] VP
4					[4,5] Adv,AdvP

Example Applications of X → YZ

- **NP → DT NG**
  - [0,1] (***the***, DT) + [1,2] (***mean***, NG) → [0,2] NP (***the mean***)
  - [0,1] (***the***, DT) + [1,3] (***mean duck***) → [0,3] (***the mean duck***)
- **NG → ADJ NG**
  - [1,2] (***mean***,Adj) + [2,3] (***duck***,NG) → [1,3] NG (***mean duck***)
- **S → NP VP**
  - [0,3] (***the mean duck***, NP) + [3,4] (***quacked***, VP) → [0,4] ***the mean duck quacked***
  - [0,3] (***the mean duck***, NP) + [3,5] (***quacked menacingly***, VP) → [0,5] ***the mean duck quacked menacingly***

Final Parse: Covers whole string [0,5]

	The	mean	duck	quacked	menacingly
	1	2	3	4	5
0	[0,1] **DT**	[0,2] NP	[0,3] **NP,S**	[0,4] S	[0,5] **S**
1		[1,2] N, **Adj**, NG	[1,3] **NG**	[1,4]	[1,5]
2			[2,3] **N**,V, **NG**,VP	[2,4]	[2,5]
3				[3,4] **V**,VP	[3,5] **VP**
4					[4,5] **Adv,AdvP**

Representing Output as Tree

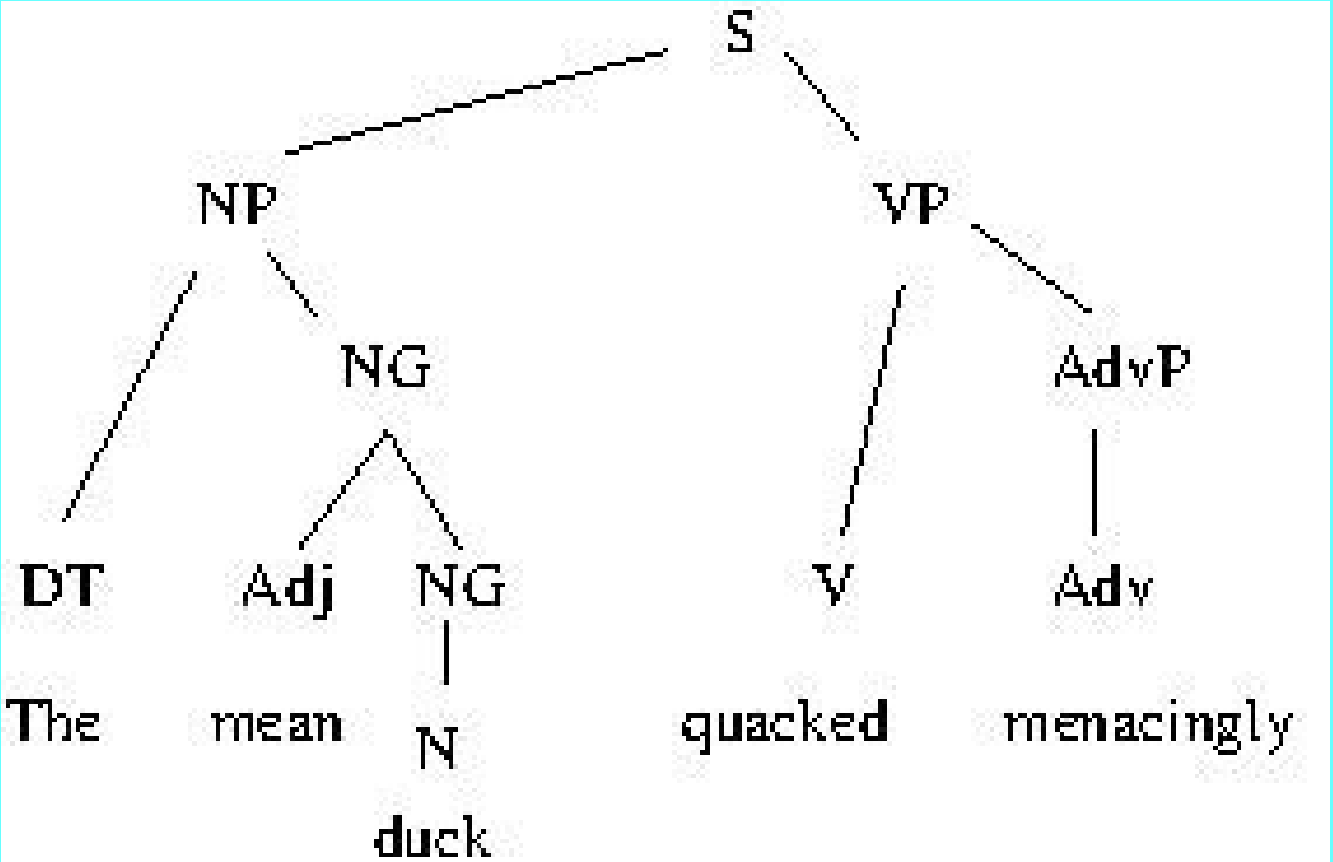


Chart Parsers/Recognizers

- States (or Edges) represent partial processing of the sentence
  - Phrase structure rule + text span + indication of part of rule is used
  - A Dot indicates how much of rule is used:
    - **NP → . det Adjp noun [0,0]** ## nothing has been recognized
    - **NP → det . Adjp noun [0,1]** ## a sentence initial det was found
- New states are created
  - to account for new or larger portions of input string
  - based on previously created states
- States are recorded in a “chart”
- A Final State
  - Represents the whole sentence
  - Instantiates a completed expansion of the start symbol

A Generic Chart Parser (13.4 in J&M)

- chart-parse(words,grammar, strategy):

  Initialize(chart,agenda,words): Add initial "edges" (or states) into chart and agenda

  while agenda is non-empty:

  current_edge ← pop_next_edge(agenda)

  process_edge(current-edge) # may change agenda & chart

  return(chart)

- process_edge(**edge**):

  Add **edge E** to chart (if not already there)

  IF incomplete(**E**):

  For each **complete_edge** that: a) starts where **E** ends and b) matches the symbol after the dot in **edge**

  Add a modified version of **E** to agenda, such that the dot in **E** is advanced past the next symbol

  ELSE IF **E** spans the input string and matches the start symbol: then do nothing

  ### Note E is complete. Possibly end processing if only 1 parse is desired.

  ELSE: ## Note E is complete

  For each **incomplete_edge** such that left_side(E**)** is after the dot:

  – Add modified **incomplete_edge** to agenda with dot advanced one symbol

  Make_prediction(**E**)

  Add additional edges to agenda based on **edge –** details depend on parsing strategy

Alternative Parsing Strategies

- Direction: Left to Right, Right to Left (order in which rules beginning with terminals are placed on agenda)
- Search Strategies
  - Bottom Up: complete constituents 1st, start w/terminals
  - Top Down: nonterminals 1st, start with start symbol
  - Breadth 1st – investigate constituents in rule in parallel
  - Depth 1st – investigate constituents in rule 1 at a time
  - Others: Left corner parsing, head-driven parsing, etc.
- Optimizations
  - Dynamic Programming

The concept of Head

- Head of a phrase – word that determines the category of the phrase
  - *the **book** about Fred* – NP because *book* is a noun
  - ***ate** a sandwich* – VP because *ate* is a verb
  - ***in** the room* – PP because *in* is a preposition
- Some phrases arguably don't have heads
  - *John and Mary*
  - *Adam Meyers*
  - *the bigger, the better*
- Heads typically subcategorize (select) other members of their phrase
  - $VP \rightarrow V_{put}\ NP\ PP{+}LOC$ *He put the book on the table*
  - $VP \rightarrow V_{laugh}\ PP\text{-}at$ *She laughed at the poodle*
- CFG representations can break up POS into subcategories which correspond to these different selectional possibilities

Typical Criteria for Head

- The single word that determines the distribution of the phrase
  - Bloomfield 1933
- The head of the phrase selects for the other members of the phrase
  - Subcategorization
  - Semantic selection, e.g., ***eat*** requires that its object be tangible (one cannot eat an idea). In NLP, a notion of statistical selection is used instead, e.g., typical objects of ***eat*** are edible (not rocks).
- Heads tend to determine the agreement properties of phrases (number and gender)
- Problem: Sometimes different items in a phrase fullfill these roles, e.g., an adjective selects the head noun, e.g., ***angry*** modifies sentient head nouns.

Subcategorization Dictionaries

- NYU built one of the widely used dictionaries of this type
  - Website including manuals and papers: http://nlp.cs.nyu.edu/comlex/
  - Latest version of Comlex dictionary (downloadable from Brightspace Google Docs):
  - Subcategorization for nouns, verbs, adjectives
  - Modification frames for adverbs
  - Owned by LDC, available from NYUClasses (COMNOM.tgz)
- Examples:

(NOUN :ORTH “authority”
:SUBC ((NOUN-FOR-TO-INF)
(NOUN-PP :PVAL (“by” “for” of” “over” …))))

(VERB :ORTH “put”
:SUBC ((NP-PP :PVAL (“on” “in” “into” “off” “out” …))
(PART-PP :ADVAL (“up” “in”) :PVAL (“with” “for”))
(NP-ADVP) …))

(ADVERB :ORTH “right”
:MODIF ((PRE-PREP :PVAL (“outside” “in” “into” “on” “out of” …))
(PRE-ADV)))

Argument Sharing Properties in the Lexicon

- Raising (subj-to-sub, subj-to-obj—aka ECM)
  - Verbs/Adjectives/Nouns link predicates and arguments
    - *The student is **likely** to leave (the student's likelihood of leaving)*
    - *The student **seemed** to leave*
    - *The student **believed** the test to have only 1 question.*
- Equi (aka control)
  - Arguments are shared between two predicates
    - *The student **desires** to leave*
    - *the student's **desire** to leave*)
    - She ***promised*** her mother to be good.

Adverbs

- Not well-defined part of speech in some systems
  - "adverb" often assigned to words not fitting other categories
- Adverbs are complement words evoking locative or evaluative concepts
  - She put the book ***there***
  - She did ***well***
- Adverbs "modify" non-nouns
  - Part of phrase, not selected by head of phrase
  - One way adverbs are classified in COMLEX is by what they modify
- Examples of modifiers of determiners or numbers
  - *all, barely, just, only, quite, scarcely*
  - *all 5 children, scarcely a solution, ...*
- Examples of modifiers of prepositions or adverbs
  - *all, just, less, quite, really, rather, right, somewhat, tightly, very*
  - *He climbed right up the wall, They walked rather slowly, ...*

Adverbs 2

- Examples of modifiers of adjectives
  - *absolutely, deeply, legally, only, not, slightly, very*
  - *He was **deeply** hurt, She was **legally** liable, They were both **very** angry*
- Some adverb modifiers of adjectives take complements as well
  - *The type was **too** small **to read***
- Examples of modifiers of verbs/sentences
  - *easily, probably, never, not, actively, then, immediately*
  - *She **actively** looked for her glasses, She **probably** found them, **Then**, I saw them at the bottom of the stairs*
  - There are further distinctions for these:
    - variation where they can occur
      - intially, finally, between subject and verb, etc.
    - Sentential adverbs are distinguished from verbal ones, but there is controversy in the details.
      - E.g., Epistemic (probably, possibly, ...) are usually considered sentential

Frameworks without traditional CFGs

- Dependency Grammar: Words are linked to form graphs
  - No concept of phrase
  - Nonheads depend on heads (so all constructions must have heads, even for arguably non-headed constructions, e.g., conjoined phrases)
- Categorial Grammar
  - There is a syntax of POS and Category names
    - Solves some of the problems with the notion head
    - An adjective is a category of type N/N because it is an item that combines with an N to produce an N
      - N/N X N = N
    - A VP is a S\NP because it combines with an NP (subject) to form an S
      - S/NP X NP = S
- Feature Structure Grammars (HPSG, LFG, etc.)
  - Generalize CFG to large attribute value structures (Covered Late in Semester)
  - See GLARF (http://nlp.cs.nyu.edu/meyers/GLARF.html)

A More Complete English Grammar

- Scope of this discussion:
  - Surface Syntax (which could conceivably be captured by phrase structure rules)
  - Subphrases of Sentences: (NP, VP, PP, ADJP, ADVP ...)
  - Ignoring for now:
    - Grammar beyond the “surface”: paraphrase relations (e.g., between active and passive, filling in of 'gaps', etc.), semantics, pragmatics, etc.
    - Nondeclarative clauses (questions, exclamations, ...) & exotic constructions
- Open Class vs Closed Class
  - Open class: parts of speech with an opened set of members. Newly coined words usually fall into these classes: nouns, verbs, adjectives, some adverb classes
  - Closed class: words with highly idiosyncratic grammatical functions. New words rarely fall into these classes. Ex: prepositions, subordinate conjunctions, coordinate conjunctions, infinitival “to”, “as”, etc.

Noun Phrases

- NP → Determiners Adjectives Nouns-as-Modifiers Head-Noun Noun-Complements Noun-post-modifiers
  - *[The happy [ice cream truck] salesman [sitting on the chair]]*
  - *[All the little piggies [that live piggy lives]]*
  - All elements except the noun are optional (sort of)
- Determiner (and DetPs) – additional rules required for multiple determiners
  - possessive phrases (NP + 's)
  - articles and demonstratives: *the, a, an, this, that, these, those*
  - numbers: *1, 2, 3, 4, 53, 175, one hundred thirty, five hundred and thirty*
  - Quantifiers: *every, all, each*, some, ….
- Adjectives (and adjective phrases and verb particples): angry, very angry
- Noun-Phrases-as-Modifiers: *the [ice cream] man, a [car insurance] policy*
  - An NP as a modifier typically: lacks a determiner and has a singular form
    - * *the [the car insurance] policy*
    - Exception: *the [three woman] band*

Right Modifers of Nouns

- Complements selected by nouns (PP or Clause)
  - *The fact [that one plus one equals two]*
  - *His anger [about the situation]*
- Non-complement PPs (ownership, location, time)
  - *The book [on the table], the book [of John's]*
- Relative clauses (missing subject, object or other)
  - *The fact [that she forgot]* --- missing object of *forgot*
- Reduced relative clauses (participles or adjectives with relative like meanings)
  - *The book [__ sitting on the table]*
  - *The people [__ angry at Simon]*
- Appositive Phrase: A second noun phrase modifying the first. It is typically separated by a pause (punctuation), and has a “that is” type of meaning.
  - *John Smith, the new president of ACME*
  - *The palamino, the horse-shaped subatomic particle*

Verb Phrases

- Verb + Complements + Adverbial Modifiers
  - *put [the book] [on the table] quietly*
- Complements: big variety see COMLEX website & dictionary:
  - http://nlp.cs.nyu.edu/comlex (Website)
  - https://drive.google.com/file/d/10YG5szTe75tRS9QFHZciMTphxiQDvnG8
    - Download from NYU Drive (ugly version entries are each on 1 line)
  - Verbs, adjectives and nouns can all take complements
- Auxilliaries: closed class words that in some grammars are labeled as types of verbs. Most theories assume a binary structure like this:
  - (VP aux (VP aux .. (VP verb …)))
  - infinitival ***to*** and modals (***may, can, could***, …) are followed by the base form of a verb
  - Forms of ***be*** are followed by either ***-ing*** forms of verbs (progressive) or past participle forms (passive) (past particples end in ***-en*** or ***-ed***)
  - Forms of ***have*** are followed by past participle forms of verbs (perfect)

Adjective Phrases

- ADJP → (ADV) ADJ Complement
  - *very angry about the situation*
- Most adjectives can occur either:
  - Attributively (as noun modifiers): ***the angry bird***
  - or as predicates (after ***be*** or other special verbs)
    - *The bird was angry, The bird got angry, They considered the bird stupid*
- Some adjectives can only occur one way
  - *The bird was alive. *That is an alive bird*
  - *He is the former president. *That president is former.*
- Prenominal ADJP do not include their complements, except for special adjectives that allow a complement after the noun
  - **The easy politician to please**
  - **??The easy to please politician** ## In written text, better if hyphenated
  - ***The angry at them people** ## Always bad

Some Other Closed Class Words

- Coordinate Conjunctions: *and*, *or*, *but*
  - Combine like constituents
    - XP → XP and XP
    - X → X and X
- Subordinate Conjunctions: *if*, *while*, *before*, ..
  - Combine clauses in special logical, temporal or discourse relationships (not equal like coordinate conjunctions)
- Prepositions:
  - Can have purely formal functions or semantic ones (similar to subordinate conjunctions)
  - Link NPs with other words/phrases

Summary Phrase Structure Rules & English Grammar

- Phrase Structure Rules (PSRs) can be created manually or via an automatic procedure
- A set of PSRs (a grammar) recognize a set of sentences which are part of a language. Strings not recognized are not part of this language.
- The best parsing (and recognition) algorithms have N^3 time complexity where N is the number of tokens in the sentence
- Writing descriptively adequate grammars of even 1 language is challenging: we wrote an OK one for a subset of English.
- Grammars typically focus only on syntactic well-formedness & recognize semantically ill-formed strings

Some Other Models of Grammar

- Categorial Grammar – Phrase structure rules are built into parts of speech.
- Dependency structures – links words instead of phrases – nonheads “depend” on heads
- I will demo these alternative frameworks using Miro

Readings/Exercises: PSRs and English Grammar

- Readings
  - Chapters 12 & 13 in J & M
  - Chapter 8 in NLTK
- Suggested exercises
  - Experiment with various parsers discussed in Chapter 8
    - Be aware that some top down parsers will not terminate given left recursive phrase structure rules like NP → NP PP
  - Choose a sentence above 20 words long and attempt to manually write a phrase structure tree based on rules. (See practice midterms).
  - Attempt to fill a CKY parse for a short sentence, assuming a grammar in CNF (see practice midterms)

Some Additional Sources

- Partee, et. al. (1990) *Mathematical Methods in Linguistics* – Gives good mathematical background to formalize linguistic concepts
- Descriptive Grammars
  - McCawley (1988) *The Syntactic Phenomena of English* (Vol 1 and 2)
  - Huddleson and Pullum (2002) *The Cambridge Grammar of the English Language*
- Introductory Books for Syntactic Theories
  - Carnie (2006) *Syntax* – Introduction to Chomskian linguistics
  - Borsley (1996) *Modern Phrase Structure Grammar* – Introduction to feature structure approaches to Syntax
  - Wood (1993) *Categorial Grammars*
  - Dependency Grammar – still looking for good intro: Prague Dependency Treebank, Kyoto Corpus, …
- Parsing and Chunking: See citations in J & M
- Interesting application of CFG generation: http://pdos.csail.mit.edu/scigen/
  - Randomly generates CS papers for submission to marginal conferences

Summary of Models of English Sentences

- 2 models for predicting what is a sentence?
  - N-gram (stochastic) model inclusion of sentence in language based on (relative) probability
  - Phrase Structure model has rules for strictly including a sentence in or excluding a sentence from a language
- N-gram model does not generate word sequences (of length N) not found in training, but phrase Structure model may generate unlikely sentences
- Phrase structure units smaller than the sentence model psychological units, N-grams probably do not
- Phase structure grammars represent a larger class of theoretical models including dependency grammars, feature structure grammars, etc. (each potentially a topic for a final project)

Probabilistic Parsing

- Probabilistic Parsing of combines stochastic and grammatical models, typically based on annotated training data
- Useful for (undergraduate) final projects as a topic or for generating features for Machine Learning systems
- Probabilistic phrase structure parsers
  - Charniak parser:http://www.aclweb.org/anthology/P01-1017
  - Collins/Bikel parser: http://www.aclweb.org/anthology/J04-4004
- J & M Version 3 about DG parsing
  - https://web.stanford.edu/~jurafsky/slp3/14.pdf
- Probabilistic Dependency Parser
  - http://www.aclweb.org/anthology/D11-1116

Final Project: Parsing

- Implement a Parser
  - Based on previous work (cite)
- Split corpus into training, development, test
  - For WSJ, use standard split (as in HW)
- Use a standard measure for parser quality
  - For constituency parsers, use Parseval
    - http://www.aclweb.org/anthology/H91-1060
    - Or other measures
- Do error analysis based on development set
- Variations: Unusual corpora, dependency parsing, languages, genres, domain adaptation, etc.