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 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.
A Statistical Language Model

• Probability distribution over sequence of words (or other linguistic unit)
• 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 and 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
• We will explore unigram and bigram models using NLTK and the python code in http://cs.nyu.edu/courses/spring18/CSCI-UA.0480-009/bigram_test.py
Unigrams

• Each individual word (instance of punctuation, etc.) is a token
  
• 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.

• One OOV model: assume words occurring once are OOV and recalculate tcounts, 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 \( w_n \), given word \( w_{n-1} \)
  - \[ \text{bigram}(\text{the}, \text{same}) = \frac{\text{count}(\text{the}, \text{same})}{\text{count}(\text{the})} \]
  - \[ \text{count}(\text{the}, \text{same}) = 628 \]
  - \[ \text{count}(\text{the}) = 69,971 \]
  - \[ \text{bigram \_ probability} = \frac{628}{69971} = 0.00898 \]
  - If the previous word is out of vocabulary, the bigram probability will be:
    - \[ \frac{\text{count(*oov*,current \_ word)}}{\text{count(*oov*)}} \]
- Additional steps
  - Include probability that a word occurs a the beginning of a sentence, i.e., bigram(\text{the}, \text{START})
  - Include probability that a token occurs at the end of a sentence, e.g., bigram(\text{END,}.)
  - Include non-zero probability for case when an unknown word follows a known one.
- Backoff Model
  - If a bigram has a zero count, “backoff” (use) the unigram of the word
    - \[ \text{replace bigram(current \_ word,previous \_ word) with unigram(current \_ word)} \]
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
Trigrams, 4-grams, N-grams

• Trigram Probability
  – $\text{Prob}(3 \text{ token sequence} \mid \text{first 2 tokens})$
  – $\frac{\text{count}(w-2, w-1, w)}{\text{count}(w-2, w-1)}$
  – $\text{count}(\text{the, same, as})/\text{count}(\text{the, same})$

• 4-gram Probability
  – $\text{Prob}(4 \text{ token sequence} \mid \text{first 3 tokens})$
  – $\frac{\text{count}(w-3, w-2, w-1, w)}{\text{count}(w-3, w-2, w-1)}$
  – $\text{count}(\text{the, same, as, an})/\text{count}(\text{the, same, as})$

• N-gram Probability
  – $\text{Prob}(N \text{ token sequence} \mid \text{N-1 tokens})$
  – $\frac{\text{count}(w-(n-1), \ldots, w-3, w-2, w-1, w)}{\text{count}(w-(n-1), \ldots, w-3, w-2, w-1)}$
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
- 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 only once in Training Corpus
    • Not counted as separate words
  – Treated as instances of one word OOV
  – Out of Vocabulary words being tested are treated as instances of this word
    – This is what we did essentially in bigram_test.py

• Other models exist, but not discussed here
Summary of bigrams

• If bigram is not found in training corpus, use unigram probability (implemented)
  – In trigram model, backoff to bigram and if that isn't there, backoff to unigram
  – In 4-gram model, ...
  – 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
  – in Brown Corpus, where we used the items 16K found once to estimate 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.
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/spring18/CSCI-UA.0480-009/bigram_test.py
Phrase Structure Model of Language

- Possible sentences in a language are modeled by a set of context free phrase structure rules.
- The terminals are parts of speech
  - A lexicon is an additional mapping from parts of speech to individual words (tokens) of a language
  - I will fudge some of these details and sometimes refer to parts of speech and sometimes words as terminals
- Non-Terminals represent phrases, sequences of 1 or more terminals
A 'Model' of a Subset of English

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

• 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)
Grammar of English' Terminal Rules

- 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 no NonTerminals
    • Replace the 1\textsuperscript{st} Nonterminal (Q) in Current String
      – Find all phrase structure rules with Q on the left hand side
        » Choose a rule Q \rightarrow \alpha, where \alpha is a string of terminals and/or nonterminals
      – Replace Q with \alpha

• 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
    • 1\textsuperscript{st} item in each list is a nonterminal
    • Remaining items are possible expansions of nonterminal
    • Example: \( \textit{DetP} \rightarrow (\textit{NP pos}) \mid (\textit{determiner}) \)
  – The file: words3.py assigns POS to words
• Possible expansions for each symbol stored in table
• generate_random_phrase
  – Optional keyword variable: \texttt{full_trace=True} (for more detail)
  – unexpanded nonterms in stack (starting w/initial symbol)
  – repeat until stack empty
    • remove top item
      – if terminal, add to result,
      – if non-terminal, add expansion stack (left most item on top)

Sentence Level Word Distribution
2018
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.

• 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: \( NP \rightarrow \text{Det Adj N} \)
    
    • Example of Equivalent 2 CNF rules: (a) \( NP \rightarrow \text{Det N-bar} \); (b) \( \text{N-bar} \rightarrow \text{Adj N} \)

More details later, but no guarantee that CNF rules model anything linguistic. Similar views are held in linguistic theories that impose a syntactic templates on all phrases.
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.
  
  – $NP \text{ read the book} \mid NP \in \{\text{Mary, The pianist with a hat, She, The uncooked eggplant, \ldots}\}$
  
  – $They \ VP \ and \ VP \mid VP \in \{\text{saw a movie, wrote poetry, are politicians, \ldots}\}$

• Two words that participate in the same inflectional paradigm are probably the same part of speech
  
  – $\{\text{kill, kills, killed, killing}\}; \{\text{love, loves, loved, loving}\}$
  
  – $\{\text{big, bigger, biggest}\}; \{\text{silly, sillier, silliest\ldots}\}$
  
  – $\{\text{house, houses}\}; \{\text{nose, noses}\ \ldots\}$
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.
Convert English' NonTerminal Rules to CNN

• Replace $VP \rightarrow \mathbf{V} \ NP \ PP$ with 2 rules:
  – $VP \rightarrow VG \ PP$
  – $VG \rightarrow V \ NP$

• Replace: $NP \rightarrow \text{POSSP} \ N \ PP$ with 2 rules:
  – $NP \rightarrow NG \ PP$
  – $NG \rightarrow \text{POSSP} \ N$

• NG = Noun Group and VG = Verb Group
Approaches to Converting Any Context Free Grammar to CNN

- \( X \rightarrow Y Z Q \)
  - \( X \rightarrow A Q \)
  - \( A \rightarrow YZ \)

- \( X \rightarrow Y Z Q R \)
  - \( X \rightarrow AB \)
  - \( A \rightarrow YZ \)
  - \( B \rightarrow QR \)

- \( X \rightarrow Y Z Q \)
  - \( X \rightarrow Y A \)
  - \( A \rightarrow Z Q \)

- \( X \rightarrow Y Z Q R \)
  - \( X \rightarrow YA \)
  - \( A \rightarrow ZB \)
  - \( B \rightarrow QR \)
Proviso about converting to Chomsky Normal Form

• Normal for CNF form to be more productive than the original (assumed in midterm exam)

• Example Long Rule: NP → DT JJ JJ NN NN

• Example CNF Rules:
  – NP → DT NG1
  – NG1 → JJ NG1
  – NG1 → NG2
  – NG2 → NN NG2
  – NG2 → NN
  – NG2 → NN NG2
Math Terms: Set Theory

• Quick Overview: set, member of, subset, superset, proper subset, proper superset

• Cartesian Product
  • Set A = {1,2,3}, Set B = {a,b,c}
  • $A \times B = \{ \{1,a\}, \{1,b\}, \{1,c\}, \{2,a\}, \{2,b\}, \{2,c\}, \{3,a\}, \{3,b\}, \{3,c\} \}$

• Partition = division into multiple non-overlapping subsets, including all members of original set
Math Terms: Set Theory 2

- Binary Relation = set of ordered pairs
- Properties of Relations
  - Reflexive: (X,X) is always in relation
  - Irreflexive: (X,X) is never in relation for any X
  - Transitive: (X,Y) and (Y,Z) → (X,Z)
  - Antisymmetric: (X,Y) and (Y,X) → X=Y
  - Asymetric: If (X,Y), then not (Y,X)
Linguistic Phrase Structure Tree

- Phrase Structure Tree = <N,Q,D,P,L>
  - N = set of nodes
  - Q = set of labels
  - D = weak partial order on N X N (the dominance relation)
    - Reflexive, transitive & antisymmetric (X & Y dominate each other iff X = Y)
  - P = strict partial order on N X N (the precedence relation)
    - Irreflexive, transitive & asymmetric (if X precedes Y & Y cannot precede X)
  - L = function from N into Q (labeling function)

- Conditions
  - Single Root (tree dominated by a single (start) nonterminal symbol)
  - A node pair cannot be both part of D and part of P
    - Only nonterminals symbols can label nonterminal nodes
  - Branches cannot cross
    - Branch A dominates nodes a1 and a2
    - Branch B dominates nodes b1 and b2
      - If a1 precedes b1, b2 cannot precede a2
  - [A CFG and a simple generator should not produce a tree violating these]
Sample English' Linguistic Phrase Structure Tree

Sentence Level Word Distribution
2018
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
English'' NonTerminal Rules Used in Example

- $S(\text{entence}) \rightarrow \text{NP } \text{VP}$
- $\text{VP} \rightarrow V(\text{erb})$
- $\text{VP} \rightarrow V(\text{erb}) \text{ NP}$
- $\text{NP} \rightarrow \text{POSSP } N(\text{oun})$
- $\text{POSSP} \rightarrow \text{NP } \text{POSS}$
- $\text{PP} \rightarrow P(\text{reposition}) \text{ NP}$
- $\text{NP} \rightarrow D(\text{eterminer}) \text{ N(oun)}$
- $\text{NP} \rightarrow N(\text{oun})$
CKY Recognizer and Parser

- Create a (triangular) table representing all spans in the sentence from 0 (the position before the first word) to N the position after sentence of length N
- For j from 1 to N do:
  - Fill in one span of length 1 using a POS rules, e.g., V → ate
    - On different iterations these will be [0, 1], [1, 2], …, [N-1, N]
  - For i from 0 to j-2 do:
    - for k from i+1 to j-1:
      - Add all matching nonterminals to [i, j] in table
      - ## fill in rest of column j
- * A nonterminal matches iff
  - There is some rule of the form A --> BC in the grammar
  - [i,k] includes the label B and [k,j] includes the label C
- Note that the inner 2 loops identify all [i,k] amd [k,j] such that i is between 0 and j-2, i<k and k<j, thus identifying the possible binary partitions of [0,j]
- Parser needs 2 additional things:
  - Differentiate different expansions of same nonterminal
  - Record pointers to matched children of nonterminals
Python Code Trace

• Do trace with python-CKY-loop.py
• Code outlines lookup instance in the CKY algorithm:
  – Looking up lexical items in outer loop
  – Looking up rule matches in loop 3
• Algorithm hinges on binary branching rules
• Adjustments of pseudo-code for python
  – range(N) of python is a list of numbers that is N long starting at 0; range(M,N) is the subrange, starting at M.
    • Range(5) = [0,1,2,3,4]
    • Range(1,5) = [1,2,3,4]
Recognize with CKY & Grammar'
Outer Loop 1^{st} Iteration

<table>
<thead>
<tr>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

| 0   | D \([0,1]\) |      |       |     |           |

| 1   |       |      |       |     |           |
| 2   |       |      |       |     |           |
| 3   |       |      |       |     |           |
| 4   |       |      |       |     |           |
| 5   |       |      |       |     |           |
2\textsuperscript{nd} Iteration

<table>
<thead>
<tr>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>D [0,1]</td>
<td>N, NP [0,1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N, NP [0,1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sentence Level Word Distribution
2018
The clam's group had knowledge

<table>
<thead>
<tr>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

0  
D [0,1]  
NP [0,2]  
POSSP [0,3]  

1  
N, NP [1,2]  
POSSP [1,3]  

2  
POSS [2,3]  

3  

4  

5  

Sentence Level Word Distribution  
2018
### 4th Iteration

<table>
<thead>
<tr>
<th></th>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>D [0,1]</td>
<td>NP [0,2]</td>
<td>POSSP [0,3]</td>
<td>NP [0,4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>N, NP [1,2]</td>
<td>POSSP [1,3]</td>
<td>NP [1,4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>POSS [2,3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>N, NP [3,4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sentence Level Word Distribution
2018
The clam's group had knowledge

<table>
<thead>
<tr>
<th></th>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>D [0,1]</td>
<td>NP [0,2]</td>
<td>POSS [0,3]</td>
<td>NP [0,4]</td>
<td>S [0,5]</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>N, NP [1,2]</td>
<td>POSS [1,3]</td>
<td>NP [1,4]</td>
<td>S [1,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>POSS [2,3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>N, NP [3,4]</td>
<td>S [3,5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>V, VP [4,5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sentence Level Word Distribution
2018
<table>
<thead>
<tr>
<th>The</th>
<th>clam</th>
<th>'s</th>
<th>group</th>
<th>had</th>
<th>knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>D [0,1]</td>
<td>NP [0,2]</td>
<td>POSSP [0,3]</td>
<td>NP [0,4]</td>
<td>S [0,5]</td>
</tr>
<tr>
<td>1</td>
<td>N, NP [1,2]</td>
<td>POSSP [1,3]</td>
<td>NP [1,4]</td>
<td>S [1,5]</td>
<td>S [1,6]</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>POSS [2,3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>V, VP [4,5]</td>
<td>VP [4,6]</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N,NP [5,6]</td>
</tr>
</tbody>
</table>
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:
    - $NP \rightarrow \cdot \text{det Adjp noun} \left[0,0\right]$  ## nothing has been recognized
    - $NP \rightarrow \text{det} \cdot \text{Adjp noun} \left[0,1\right]$  ## 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
  - \[ VP \rightarrow V_{\text{put}} \quad NP \quad PP+LOC \quad \text{He put the book on the table} \]
  - \[ VP \rightarrow V_{\text{laugh}} \quad PP-\text{at} \quad \text{She laughed at the poodle} \]

- 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 fulfill 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
  - [http://nlp.cs.nyu.edu/comlex/](http://nlp.cs.nyu.edu/comlex/)
  - 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

• Mostly “adverb” refers that “modify” a non-noun and/or express locative, verbal, evaluative concepts.
  – by modify, I mean 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 (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 participles): 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 Modifiers 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 http://nlp.cs.nyu.edu/comlex
  - Complements for verbs can be obligatory
- 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
Chunking

• Linguistic transduction that selects short approximations of linguistic phrases or chunks

• Most Common Chunks:
  – Verb Group: A unit beginning at an auxiliary verb and ending at a matrix verb or particle (that immediately follows the matrix verb) – semantic justification possible, but contrary to most syntactic analyses:
    • *The cars [may have been slowly stripped down]*
    • *They [could have stolen] the cars instead*
  – Noun Group: Unit consisting of a noun & its pre-modifiers
    • *[The big bad wolf] that huffed and puffed*
    • *[President G. W. Bush], [a goofy president] who ...*

• It is possible to create quick & accurate processors
Chunking 2: Methodology

- Finite state transducer
  - Using a left regular grammar similar to English'
- ML Chunker described in J & M (Similar to Homework 5)
  - Assumes a pre-processing stage of POS tagging
  - Assumes an annotated corpus for “training”
    - Four types of tags for each chunk type:
      - Beginning (B_NG)
      - Interior (I_NG)
      - Outside (O)
    - *The/B_NG big/I_NG cat/I_NG without/O a/B_NG tail/I_NG caught/O a/B_NG mouse/I_NG*
  - 13-tuple for each word includes: Chunk tag, word, word's POS, 2 words to left, 2 words to right, and POS info for 2 words to left and right
  - Frequency of subsets of these fields are recorded in training and used for final system to predict tags (most frequent matching pattern is used)
Chunking 3: Sample Regexp Rules for Chunks

- \( NG \rightarrow (\text{Det})* (\text{AdjP})* (\text{Noun})* \text{Noun} \)
- \( \text{AdjP} \rightarrow (\text{Adv})* \text{Adj} \)
- \( \text{VG} \rightarrow (\text{Adv})* (\text{Aux})* (\text{Adv})* (\text{Verb} \mid \text{VerbPart}) \)
- \( \text{VerbPart} \rightarrow (\text{call up}) \mid (\text{break up}) \mid (\text{call out}) \mid \ldots \)
  - disjunction of 500 verb particle combinations and their inflectional variants
- \( \text{Det} \rightarrow \text{the} | \text{a} | \text{each} | \text{every} | \text{this} | \text{that} | \text{these} | \ldots \)
  - Noun, Adv, Adj, Aux, Verb are also defined as disjunctions of words taken from a dictionary
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.
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

• Introductory Books for Syntactic Theories
  – Carnie (2006) *Syntax* – Introduction to Chomskian linguistics
  – Wood (1993) *Categorial Grammars*
  – Dependency Grammar – still looking for good intro: Prague Dependency Treebank, Kyoto Corpus, …

• Parsing and Chunking: See citations in J & M

  – 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 combines stochastic and grammatical models, typically based on annotated training data.
- Useful for 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
- Probabilistic Dependency Parser
  - http://www.aclweb.org/anthology/D11-1116