QUERY EXPANSION, RELEVANCE FEEDBACK, POS TAGGING AND SYNTACTIC PARSING

Heng Ji

jih@rpi.edu
February 19, 2014
Outline

- Query Expansion and Relevance Feedback
- POS Tagging and HMM
- Formal Grammars
  - Context-free grammar
  - Grammars for English
  - Treebanks
- Parsing and CKY Algorithm
Query expansion

• A query contains part of the important words
• Add new (related) terms into the query
  • Manually constructed knowledge base/thesaurus (e.g. Wordnet)
    • Q = information retrieval
    • Q’ = (information + data + knowledge + …)
      (retrieval + search + seeking + …)
• Corpus analysis:
  • two terms that often co-occur are related (Mutual information)
  • Two terms that co-occur with the same words are related (e.g. T-shirt and coat with wear, …)
Query expansion

• A query contains part of the important words
• Add new (related) terms into the query
  • Manually constructed knowledge base/thesaurus (e.g. Wordnet)
    • Q = information retrieval
    • Q’ = (information + data + knowledge + …) (retrieval + search + seeking + …)

• Corpus analysis:
  • two terms that often co-occur are related (Mutual information)
  • Two terms that co-occur with the same words are related (e.g. T-shirt and coat with wear, …)
Automatic thesaurus generation

- Attempt to generate a thesaurus automatically by analyzing the distribution of words in documents
- Fundamental notion: similarity between two words
- Definition 1: Two words are similar if they co-occur with similar words.
  - “car” ≈ “motorcycle” because both occur with “road”, “gas” and “license”, so they must be similar.
- Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
  - You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence is more robust, grammatical relations are more accurate.
Co-occurrence-based thesaurus:

Examples

<table>
<thead>
<tr>
<th>Word</th>
<th>Nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolutely</td>
<td>absurd whatsoever totally exactly nothing</td>
</tr>
<tr>
<td>bottomed</td>
<td>dip copper drops topped slide trimmed</td>
</tr>
<tr>
<td>captivating</td>
<td>shimmer stunningly superbly plucky witty</td>
</tr>
<tr>
<td>doghouse</td>
<td>dog porch crawling beside downstairs</td>
</tr>
<tr>
<td>makeup</td>
<td>repellent lotion glossy sunscreen skin gel</td>
</tr>
<tr>
<td>mediating</td>
<td>reconciliation negotiate case conciliation</td>
</tr>
<tr>
<td>keeping</td>
<td>hoping bring wiping could some would</td>
</tr>
<tr>
<td>lithographs</td>
<td>drawings Picasso Dali sculptures Gauguin</td>
</tr>
<tr>
<td>pathogens</td>
<td>toxins bacteria organisms bacterial parasite</td>
</tr>
<tr>
<td>senses</td>
<td>grasp psyche truly clumsy naive innate</td>
</tr>
</tbody>
</table>
Query expansion at search engines

- Main source of query expansion at search engines: query logs
- Example 1: After issuing the query [herbs], users frequently search for [herbal remedies].
  - “herbal remedies” is potential expansion of “herb”.
- Example 2: Users searching for [flower pix] frequently click on the URL photobucket.com/flower. Users searching for [flower clipart] frequently click on the same URL.
  - “flower clipart” and “flower pix” are potential expansions of each other.
Relevance feedback: Basic idea

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as nonrelevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.
Relevance feedback

- We can iterate this: several rounds of relevance feedback.
- We will use the term *ad hoc retrieval* to refer to regular retrieval without relevance feedback.
- We will now look at three different examples of relevance feedback that highlight different aspects of the process.
Initial query:
[new space satellite applications] Results for initial query: (r = rank)

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>Score</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>0.539</td>
<td>NASA Hasn’t Scrapped Imaging Spectrometer</td>
</tr>
<tr>
<td>+</td>
<td>2</td>
<td>0.533</td>
<td>NASA Scratches Environment Gear From Satellite Plan</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.528</td>
<td>Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.526</td>
<td>A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.525</td>
<td>Scientist Who Exposed Global Warming Proposes Satellites for Climate Research</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.524</td>
<td>Report Provides Support for the Critics Of Using Big Satellites to Study Climate</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.516</td>
<td>Arianespace Receives Satellite Launch Pact From Telesat Canada</td>
</tr>
<tr>
<td>+</td>
<td>8</td>
<td>0.509</td>
<td>Telecommunications Tale of Two Companies</td>
</tr>
</tbody>
</table>

User then marks relevant documents with “+”.
Expanded query after relevance feedback

<table>
<thead>
<tr>
<th>Term</th>
<th>Term</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>space</td>
<td>2.074</td>
</tr>
<tr>
<td>satellite</td>
<td>application</td>
<td>30.816</td>
</tr>
<tr>
<td>nasa</td>
<td>eos</td>
<td>5.991</td>
</tr>
<tr>
<td>launch</td>
<td>aster</td>
<td>4.196</td>
</tr>
<tr>
<td>instrument</td>
<td>arianespace</td>
<td>3.516</td>
</tr>
<tr>
<td>bundespost</td>
<td>ss</td>
<td>3.004</td>
</tr>
<tr>
<td>rocket</td>
<td>scientist</td>
<td>2.790</td>
</tr>
<tr>
<td>broadcast</td>
<td>earth</td>
<td>2.003</td>
</tr>
<tr>
<td>oil</td>
<td>measure</td>
<td>0.836</td>
</tr>
<tr>
<td>measure</td>
<td></td>
<td>0.646</td>
</tr>
</tbody>
</table>

query: [new space satellite applications]
### Results for expanded query

<table>
<thead>
<tr>
<th>r</th>
<th>Score</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.513</td>
<td>NASA Scratches Environment Gear From Satellite Plan</td>
</tr>
<tr>
<td>2</td>
<td>0.500</td>
<td>NASA Hasn't Scrapped Imaging Spectrometer</td>
</tr>
<tr>
<td>3</td>
<td>0.493</td>
<td>When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own</td>
</tr>
<tr>
<td>4</td>
<td>0.493</td>
<td>NASA Uses ‘Warm’ Superconductors For Fast Circuit</td>
</tr>
<tr>
<td>5</td>
<td>0.492</td>
<td>Telecommunications Tale of Two Companies</td>
</tr>
<tr>
<td>6</td>
<td>0.491</td>
<td>Soviets May Adapt Parts of SS-20 Missile For Commercial Use</td>
</tr>
<tr>
<td>7</td>
<td>0.490</td>
<td>Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers</td>
</tr>
<tr>
<td>8</td>
<td>0.490</td>
<td>Rescue of Satellite By Space Agency To Cost $90 Million</td>
</tr>
</tbody>
</table>
Relevance feedback on initial query

Initial query

Revised query

x known non-relevant documents
o known relevant documents
Watson Question Analysis

• Mistake @ this step => $P(\text{wrong answer}) \approx 1$

• Elements of Question analysis are:
  
  1. Focus detection
     Part of the question that is the reference to the answer.

  2. Lexical Answer Types (LATs)
     Strings in the clue that indicate what type of entity is being asked for

  3. Question Classification
     Logical categorization of question in definite class to narrow down the scope of search.
     Example: Why, Definition, Fact

  4. Question decomposition
     Breaking question in the logical sub parts
POETS & POETRY: He was a bank clerk in the Yukon before he published Songs of a Sourdough in 1907.

FICTIONAL ANIMALS: The name of this character, introduced in 1894, comes from the Hindi for bear. (Answer: Baloo).

→ Sub-question 1: Find the characters introduced in 1894.
→ Sub-question 2: Find the words that come from hindi for bear.

Evidence for both of the sub-questions are combined for Scoring.
Watson Question Analysis

• Provides an analytical structure of questions posed and textual knowledge.
Outline

• Query Expansion and Relevance Feedback
• POS Tagging and HMM
• Formal Grammars
  • Context-free grammar
  • Grammars for English
  • Treebanks
• Parsing and CKY Algorithm
What is Part-of-Speech (POS)

- Generally speaking, Word Classes (=POS):
  - Verb, Noun, Adjective, Adverb, Article, ...

- We can also include inflection:
  - Verbs: Tense, number, ...
  - Nouns: Number, proper/common, ...
  - Adjectives: comparative, superlative, ...
  - ...

Parts of Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...
  - Lots of debate within linguistics about the number, nature, and universality of these
    - We’ll completely ignore this debate.
7 Traditional POS Categories

- **N** noun chair, bandwidth, pacing
- **V** verb study, debate, munch
- **ADJ** adj purple, tall, ridiculous
- **ADV** adverb unfortunately, slowly,
- **P** preposition of, by, to
- **PRO** pronoun I, me, mine
- **DET** determiner the, a, that, those
POS Tagging

• The process of assigning a part-of-speech or lexical class marker to each word in a collection.

<table>
<thead>
<tr>
<th>WORD</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>koala</td>
<td>N</td>
</tr>
<tr>
<td>put</td>
<td>V</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>keys</td>
<td>N</td>
</tr>
<tr>
<td>on</td>
<td>P</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>table</td>
<td>N</td>
</tr>
</tbody>
</table>
Penn TreeBank POS Tag Set

- Penn Treebank: hand-annotated corpus of *Wall Street Journal*, 1M words
- 46 tags
- Some particularities:
  - *to* /TO not disambiguated
  - Auxiliaries and verbs not distinguished
### Penn Treebank Tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordin. Conjunction</td>
<td><em>and</em>, <em>but</em>, <em>or</em></td>
<td>SYM</td>
<td>Symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td><em>one</em>, <em>two</em>, <em>three</em></td>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td><em>a</em>, <em>the</em></td>
<td>UH</td>
<td>Interjection</td>
<td><em>ah</em>, <em>oops</em></td>
</tr>
<tr>
<td>EX</td>
<td>Existential ‘there’</td>
<td><em>there</em></td>
<td>VB</td>
<td>Verb, base form</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td><em>mea culpa</em></td>
<td>VBD</td>
<td>Verb, past tense</td>
<td><em>ate</em></td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td><em>of</em>, <em>in</em>, <em>by</em></td>
<td>VBG</td>
<td>Verb, gerund</td>
<td><em>eating</em></td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td><em>yellow</em></td>
<td>VBN</td>
<td>Verb, past participle</td>
<td><em>eaten</em></td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td><em>bigger</em></td>
<td>VBP</td>
<td>Verb, non-3sg pres</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td><em>wildest</em></td>
<td>VBZ</td>
<td>Verb, 3sg pres</td>
<td><em>eats</em></td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td><em>1</em>, <em>2</em>, <em>One</em></td>
<td>WDT</td>
<td>Wh-determiner</td>
<td><em>which</em>, <em>that</em></td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td><em>can</em>, <em>should</em></td>
<td>WP</td>
<td>Wh-pronoun</td>
<td><em>what</em>, <em>who</em></td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td><em>llama</em></td>
<td>WPS</td>
<td>Possessive wh-</td>
<td><em>whose</em></td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td><em>llamas</em></td>
<td>WRB</td>
<td>Wh-adverb</td>
<td><em>how</em>, <em>where</em></td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td><em>IBM</em></td>
<td>$</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td><em>Carolin</em>as</td>
<td>#</td>
<td>Pound sign</td>
<td>#</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td><em>all</em>, <em>both</em></td>
<td>“</td>
<td>Left quote</td>
<td>‘ or “</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td><em>’s</em></td>
<td>”</td>
<td>Right quote</td>
<td>’ or ”</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
<td><em>I</em>, <em>you</em>, <em>he</em></td>
<td>(</td>
<td>Left parenthesis</td>
<td>[, (, {, &lt;</td>
</tr>
<tr>
<td>PRPS</td>
<td>Possessive pronoun</td>
<td><em>your</em>, <em>one’s</em></td>
<td>)</td>
<td>Right parenthesis</td>
<td>], ), }, &gt;</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td><em>quickly</em>, <em>never</em></td>
<td>,</td>
<td>Comma</td>
<td>,</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td><em>faster</em></td>
<td>.</td>
<td>Sentence-final punc</td>
<td>. ! ?</td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td><em>fastest</em></td>
<td>:</td>
<td>Mid-sentence punc</td>
<td>; ... -- -</td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
<td><em>up</em>, <em>off</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.6** Penn Treebank part-of-speech tags (including punctuation).
Why POS tagging is useful?

- Speech synthesis:
  - How to pronounce “lead“?
  - INsult inSULT
  - OBject obJECT
  - OVERflow overFLOW
  - DIScount disCOUNT
  - CONtent conTENT
- Stemming for information retrieval
  - Can search for “aardvarks” get “aardvark”
- Parsing and speech recognition and etc
  - Possessive pronouns (my, your, her) followed by nouns
  - Personal pronouns (I, you, he) likely to be followed by verbs
  - Need to know if a word is an N or V before you can parse
- Information extraction
  - Finding names, relations, etc.
- Machine Translation
Open and Closed Classes

• Closed class: a small fixed membership
  • Prepositions: of, in, by, …
  • Auxiliaries: may, can, will had, been, …
  • Pronouns: I, you, she, mine, his, them, …
  • Usually function words (short common words which play a role in grammar)
• Open class: new ones can be created all the time
  • English has 4: Nouns, Verbs, Adjectives, Adverbs
  • Many languages have these 4, but not all!
Open Class Words

• Nouns
  • Proper nouns (Boulder, Granby, Eli Manning)
    • English capitalizes these.
  • Common nouns (the rest).
  • Count nouns and mass nouns
    • Count: have plurals, get counted: goat/goats, one goat, two goats
    • Mass: don’t get counted (snow, salt, communism) (*two snows)

• Adverbs: tend to modify things
  • Unfortunately, John walked home extremely slowly yesterday
  • Directional/locative adverbs (here, home, downhill)
  • Degree adverbs (extremely, very, somewhat)
  • Manner adverbs (slowly, slinkily, delicately)

• Verbs
  • In English, have morphological affixes (eat/eats/eaten)
Closed Class Words

Examples:
- prepositions: on, under, over, ...
- particles: up, down, on, off, ...
- determiners: a, an, the, ...
- pronouns: she, who, I, ...
- conjunctions: and, but, or, ...
- auxiliary verbs: can, may should, ...
- numerals: one, two, three, third, ...
## Prepositions from CELEX

<table>
<thead>
<tr>
<th>Preposition</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>540,085</td>
</tr>
<tr>
<td>in</td>
<td>331,235</td>
</tr>
<tr>
<td>for</td>
<td>142,421</td>
</tr>
<tr>
<td>to</td>
<td>125,691</td>
</tr>
<tr>
<td>with</td>
<td>124,965</td>
</tr>
<tr>
<td>on</td>
<td>109,129</td>
</tr>
<tr>
<td>at</td>
<td>100,169</td>
</tr>
<tr>
<td>by</td>
<td>77,794</td>
</tr>
<tr>
<td>from</td>
<td>74,843</td>
</tr>
<tr>
<td>about</td>
<td>38,428</td>
</tr>
<tr>
<td>than</td>
<td>20,210</td>
</tr>
<tr>
<td>over</td>
<td>18,071</td>
</tr>
<tr>
<td>through</td>
<td>14,964</td>
</tr>
<tr>
<td>after</td>
<td>13,670</td>
</tr>
<tr>
<td>between</td>
<td>13,275</td>
</tr>
<tr>
<td>under</td>
<td>9,525</td>
</tr>
<tr>
<td>per</td>
<td>6,515</td>
</tr>
<tr>
<td>among</td>
<td>5,090</td>
</tr>
<tr>
<td>within</td>
<td>5,030</td>
</tr>
<tr>
<td>towards</td>
<td>4,700</td>
</tr>
<tr>
<td>above</td>
<td>3,056</td>
</tr>
<tr>
<td>near</td>
<td>2,026</td>
</tr>
<tr>
<td>off</td>
<td>1,695</td>
</tr>
<tr>
<td>past</td>
<td>1,575</td>
</tr>
<tr>
<td>worth</td>
<td>1,563</td>
</tr>
<tr>
<td>toward</td>
<td>1,390</td>
</tr>
<tr>
<td>plus</td>
<td>750</td>
</tr>
<tr>
<td>till</td>
<td>686</td>
</tr>
<tr>
<td>amongst</td>
<td>525</td>
</tr>
<tr>
<td>via</td>
<td>351</td>
</tr>
<tr>
<td>amid</td>
<td>222</td>
</tr>
<tr>
<td>underneath</td>
<td>164</td>
</tr>
<tr>
<td>versus</td>
<td>113</td>
</tr>
<tr>
<td>amidst</td>
<td>67</td>
</tr>
<tr>
<td>sans</td>
<td>20</td>
</tr>
<tr>
<td>circa</td>
<td>14</td>
</tr>
<tr>
<td>pace</td>
<td>12</td>
</tr>
<tr>
<td>nigh</td>
<td>9</td>
</tr>
<tr>
<td>re</td>
<td>4</td>
</tr>
<tr>
<td>mid</td>
<td>3</td>
</tr>
<tr>
<td>o’er</td>
<td>2</td>
</tr>
<tr>
<td>but</td>
<td>0</td>
</tr>
<tr>
<td>ere</td>
<td>0</td>
</tr>
<tr>
<td>less</td>
<td>0</td>
</tr>
<tr>
<td>midst</td>
<td>0</td>
</tr>
<tr>
<td>o’</td>
<td>0</td>
</tr>
<tr>
<td>thru</td>
<td>0</td>
</tr>
<tr>
<td>vice</td>
<td>0</td>
</tr>
</tbody>
</table>
# English Particles

<table>
<thead>
<tr>
<th>aboard</th>
<th>aside</th>
<th>besides</th>
<th>forward(s)</th>
<th>opposite</th>
<th>through</th>
</tr>
</thead>
<tbody>
<tr>
<td>about</td>
<td>astray</td>
<td>between</td>
<td>home</td>
<td>out</td>
<td>throughout</td>
</tr>
<tr>
<td>above</td>
<td>away</td>
<td>beyond</td>
<td>in</td>
<td>outside</td>
<td>together</td>
</tr>
<tr>
<td>across</td>
<td>back</td>
<td>by</td>
<td>inside</td>
<td>over</td>
<td>under</td>
</tr>
<tr>
<td>ahead</td>
<td>before</td>
<td>close</td>
<td>instead</td>
<td>overhead</td>
<td>underneath</td>
</tr>
<tr>
<td>alongside</td>
<td>behind</td>
<td>down</td>
<td>near</td>
<td>past</td>
<td>up</td>
</tr>
<tr>
<td>apart</td>
<td>below</td>
<td>east, etc.</td>
<td>off</td>
<td>round</td>
<td>within</td>
</tr>
<tr>
<td>around</td>
<td>beneath</td>
<td>eastward(s), etc.</td>
<td>on</td>
<td>since</td>
<td>without</td>
</tr>
</tbody>
</table>
## Conjunctions

<table>
<thead>
<tr>
<th>Conjunction</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>514,946</td>
</tr>
<tr>
<td>that</td>
<td>134,773</td>
</tr>
<tr>
<td>but</td>
<td>96,889</td>
</tr>
<tr>
<td>or</td>
<td>76,563</td>
</tr>
<tr>
<td>as</td>
<td>54,608</td>
</tr>
<tr>
<td>if</td>
<td>53,917</td>
</tr>
<tr>
<td>when</td>
<td>37,975</td>
</tr>
<tr>
<td>because</td>
<td>23,626</td>
</tr>
<tr>
<td>so</td>
<td>12,933</td>
</tr>
<tr>
<td>before</td>
<td>10,720</td>
</tr>
<tr>
<td>though</td>
<td>10,329</td>
</tr>
<tr>
<td>than</td>
<td>9,511</td>
</tr>
<tr>
<td>while</td>
<td>8,144</td>
</tr>
<tr>
<td>after</td>
<td>7,042</td>
</tr>
<tr>
<td>whether</td>
<td>5,978</td>
</tr>
<tr>
<td>for</td>
<td>5,935</td>
</tr>
<tr>
<td>although</td>
<td>5,424</td>
</tr>
<tr>
<td>until</td>
<td>5,072</td>
</tr>
<tr>
<td>yet</td>
<td>5,040</td>
</tr>
<tr>
<td>since</td>
<td>4,843</td>
</tr>
<tr>
<td>where</td>
<td>3,952</td>
</tr>
<tr>
<td>nor</td>
<td>3,078</td>
</tr>
<tr>
<td>once</td>
<td>2,826</td>
</tr>
<tr>
<td>unless</td>
<td>2,205</td>
</tr>
<tr>
<td>why</td>
<td>1,333</td>
</tr>
<tr>
<td>now</td>
<td>1,290</td>
</tr>
<tr>
<td>whenever</td>
<td>913</td>
</tr>
<tr>
<td>whereas</td>
<td>867</td>
</tr>
<tr>
<td>except</td>
<td>864</td>
</tr>
<tr>
<td>till</td>
<td>686</td>
</tr>
<tr>
<td>provided</td>
<td>594</td>
</tr>
<tr>
<td>whilst</td>
<td>351</td>
</tr>
<tr>
<td>suppose</td>
<td>281</td>
</tr>
<tr>
<td>cos</td>
<td>188</td>
</tr>
<tr>
<td>supposing</td>
<td>185</td>
</tr>
<tr>
<td>considering</td>
<td>174</td>
</tr>
<tr>
<td>lest</td>
<td>131</td>
</tr>
<tr>
<td>albeit</td>
<td>104</td>
</tr>
<tr>
<td>providing</td>
<td>96</td>
</tr>
<tr>
<td>whereupon</td>
<td>85</td>
</tr>
<tr>
<td>seeing</td>
<td>63</td>
</tr>
<tr>
<td>directly</td>
<td>26</td>
</tr>
<tr>
<td>ere</td>
<td>12</td>
</tr>
<tr>
<td>notwithstanding</td>
<td>3</td>
</tr>
<tr>
<td>according as</td>
<td>0</td>
</tr>
<tr>
<td>as if</td>
<td>0</td>
</tr>
<tr>
<td>as long as</td>
<td>0</td>
</tr>
<tr>
<td>as though</td>
<td>0</td>
</tr>
<tr>
<td>both and</td>
<td>0</td>
</tr>
<tr>
<td>but that</td>
<td>0</td>
</tr>
<tr>
<td>but then</td>
<td>0</td>
</tr>
<tr>
<td>but then again</td>
<td>0</td>
</tr>
<tr>
<td>either or</td>
<td>0</td>
</tr>
<tr>
<td>forasmuch as</td>
<td>0</td>
</tr>
<tr>
<td>however</td>
<td>0</td>
</tr>
<tr>
<td>immediately</td>
<td>0</td>
</tr>
<tr>
<td>in as far as</td>
<td>0</td>
</tr>
<tr>
<td>in so far as</td>
<td>0</td>
</tr>
<tr>
<td>inasmuch as</td>
<td>0</td>
</tr>
<tr>
<td>insomuch as</td>
<td>0</td>
</tr>
<tr>
<td>insomuch that</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>0</td>
</tr>
<tr>
<td>neither nor</td>
<td>0</td>
</tr>
<tr>
<td>now that</td>
<td>0</td>
</tr>
<tr>
<td>only</td>
<td>0</td>
</tr>
<tr>
<td>provided that</td>
<td>0</td>
</tr>
<tr>
<td>providing that</td>
<td>0</td>
</tr>
<tr>
<td>seeing as</td>
<td>0</td>
</tr>
<tr>
<td>seeing as how</td>
<td>0</td>
</tr>
<tr>
<td>seeing that</td>
<td>0</td>
</tr>
<tr>
<td>without</td>
<td>0</td>
</tr>
</tbody>
</table>
POS Tagging
Choosing a Tagset

• There are so many parts of speech, potential distinctions we can draw
• To do POS tagging, we need to choose a standard set of tags to work with
• Could pick very coarse tagsets
  • N, V, Adj, Adv.
• More commonly used set is finer grained, the “Penn TreeBank tagset”, 45 tags
  • PRP$, WRB, WP$, VBG
• Even more fine-grained tagsets exist
Using the Penn Tagset

- The grand jury commented on a number of other topics.
- Prepositions and subordinating conjunctions marked IN ("although I...")
- Except the preposition/complementizer “to” is just marked “TO”.
POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB

- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin
## How Hard is POS Tagging? Measuring Ambiguity

<table>
<thead>
<tr>
<th></th>
<th>87-tag Original Brown</th>
<th>45-tag Treebank Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unambiguous (1 tag)</strong></td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td><strong>Ambiguous (2–7 tags)</strong></td>
<td>5,490</td>
<td>8844</td>
</tr>
<tr>
<td><strong>Details:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 tags</td>
<td>4,967</td>
<td></td>
</tr>
<tr>
<td>3 tags</td>
<td>411</td>
<td>1621</td>
</tr>
<tr>
<td>4 tags</td>
<td>91</td>
<td>357</td>
</tr>
<tr>
<td>5 tags</td>
<td>17</td>
<td>90</td>
</tr>
<tr>
<td>6 tags</td>
<td>2 (well, beat)</td>
<td></td>
</tr>
<tr>
<td>7 tags</td>
<td>2 (still, down)</td>
<td></td>
</tr>
<tr>
<td>8 tags</td>
<td></td>
<td>4 (‘s, half, back, a)</td>
</tr>
<tr>
<td>9 tags</td>
<td></td>
<td>3 (that, more, in)</td>
</tr>
</tbody>
</table>
Current Performance

• How many tags are correct?
  • About 97% currently
  • But baseline is already 90%
  • Baseline algorithm:
    • Tag every word with its most frequent tag
    • Tag unknown words as nouns

• How well do people do?
Quick Test: Agreement?

• the students went to class
• plays well with others
• fruit flies like a banana

DT: the, this, that
NN: noun
VB: verb
P: preposition
ADV: adverb
Quick Test

• the students went to class
  DT  NN   VB   P   NN

• plays well with others
  VB  ADV P   NN
  NN  NN   P   DT

• fruit flies like a banana
  NN  NN   VB  DT  NN
  NN  VB  P  DT  NN
  NN  NN   P  DT  NN
  NN  VB  VB  DT  NN
How to do it? History

1960
- Brown Corpus Created (EN-US) 1 Million Words

1970
- Greene and Rubin Rule Based - 70%
- Brown Corpus Tagged

1980
- HMM Tagging (CLAWS) 93%-95%
- Transformation Based Tagging (Eric Brill) Rule Based – 95%
- Trigram Tagger (Kempe) 96%+

1990
- Combined Methods 98%+
- Tree-Based Statistics (Helmut Schmidt) Rule Based – 96%+
- Neural Network 96%+

2000
- British National Corpus (tagged by CLAWS)
- Penn Treebank Corpus (WSJ, 4.5M)
- LOB Corpus Created (EN-UK) 1 Million Words
- POS Tagging separated from other NLP
Two Methods for POS Tagging

1. Rule-based tagging
   • (ENG_TWOL)

2. Stochastic
   1. Probabilistic sequence models
      • HMM (Hidden Markov Model) tagging
      • MEMMs (Maximum Entropy Markov Models)
Rule-Based Tagging

• Start with a dictionary
• Assign all possible tags to words from the dictionary
• Write rules by hand to selectively remove tags
• Leaving the correct tag for each word.
Rule-based taggers

- Early POS taggers all hand-coded
- Most of these (Harris, 1962; Greene and Rubin, 1971) and the best of the recent ones, ENGTWOL (Voutilainen, 1995) based on a two-stage architecture
  - Stage 1: look up word in lexicon to give list of potential POSs
  - Stage 2: Apply rules which certify or disallow tag sequences
- Rules originally handwritten; more recently Machine Learning methods can be used
Start With a Dictionary

• she: PRP
• promised: VBN, VBD
• to: TO
• back: VB, JJ, RB, NN
• the: DT
• bill: NN, VB

• Etc… for the ~100,000 words of English with more than 1 tag
Assign Every Possible Tag

She promised to back the bill
Write Rules to Eliminate Tags

Eliminate VBN if VBD is an option when VBN|VBD follows “<start> PRP”

```
   NN
   RB
   VBN
   JJ
   VB
   PRP
   VBD
   TO
   VB
   DT
   NN

She promised to back the bill
```
POS tagging

The involvement of ion channels in B and T lymphocyte activation is supported by many reports of changes in ion fluxes and membrane 

Unseen text

We demonstrate that …

Machine Learning Algorithm

We demonstrate that …
Goal of POS Tagging

- We want the best set of tags for a sequence of words (a sentence)
- $W$ — a sequence of words
- $T$ — a sequence of tags

$$
\hat{T} = \arg\max_T P(T | W)
$$

- Example:
  $P((\text{NN NN P DET ADJ NN}) | (\text{heat oil in a large pot}))$
But, the Sparse Data Problem ...

- Rich Models often require vast amounts of data
- Count up instances of the string "heat oil in a large pot" in the training corpus, and pick the most common tag assignment to the string..
- Too many possible combinations
POS Tagging as Sequence Classification

- We are given a sentence (an “observation” or “sequence of observations”)
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$. 
Getting to HMMs

- We want, out of all sequences of $n$ tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

\[
\hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n | w_1^n)
\]

- Hat $\hat{}$ means “our estimate of the best one”
- Argmax$_x$ f(x) means “the x such that f(x) is maximized”
Getting to HMMs

• This equation is guaranteed to give us the best tag sequence

\[
\hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n | w_1^n)
\]

• But how to make it operational? How to compute this value?

• Intuition of Bayesian classification:
  • Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute
Reminder: Apply Bayes’ Theorem (1763)

\[ P(T \mid W) = \frac{P(W \mid T)P(T)}{P(W)} \]

Our Goal: To maximize it!

Reverend Thomas Bayes — Presbyterian minister (1702-1761)
How to Count

\[ \hat{T} = \arg\max_T P(T | W) \]

\[ = \arg\max_T \frac{P(W | T)P(T)}{P(W)} \]

\[ = \arg\max_T P(W | T)P(T) \]

- \( P(W|T) \) and \( P(T) \) can be counted from a large hand-tagged corpus; and smooth them to get rid of the zeroes
Count $P(W|T)$ and $P(T)$

- Assume each word in the sequence depends only on its corresponding tag:

$$P(W | T) \approx \prod_{i=1}^{n} P(w_i | t_i)$$
Count $P(T)$

\[ P(t_1, \ldots, t_n) = P(t_1) \cdot P(t_2 \mid t_1) \cdot P(t_3 \mid t_1t_2) \cdot \ldots \cdot P(t_n \mid t_1, \ldots, t_{n-1}) \]

- Make a Markov assumption and use N-grams over tags ...
- $P(T)$ is a product of the probability of N-grams that make it up

\[ P(t_1, \ldots, t_n) = P(t_1) \cdot \prod_{i=2}^{n} P(t_i \mid t_{i-1}) \]
Part-of-speech tagging with Hidden Markov Models

\[ P(t_1...t_n \mid w_1...w_n) = \frac{P(w_1...w_n \mid t_1...t_n)P(t_1...t_n)}{P(w_1...w_n)} \]

\[ \propto P(w_1...w_n \mid t_1...t_n)P(t_1...t_n) \]

\[ \approx \prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1}) \]
Analyzing

Fish sleep.
A Simple POS HMM
Word Emission Probabilities

\[ P(\text{word} | \text{state}) \]

- A two-word language: “fish” and “sleep”
- Suppose in our training corpus,
  - “fish” appears 8 times as a noun and 5 times as a verb
  - “sleep” appears twice as a noun and 5 times as a verb
- Emission probabilities:
  - Noun
    - \( P(\text{fish} | \text{noun}) : 0.8 \)
    - \( P(\text{sleep} | \text{noun}) : 0.2 \)
  - Verb
    - \( P(\text{fish} | \text{verb}) : 0.5 \)
    - \( P(\text{sleep} | \text{verb}) : 0.5 \)
Viterbi Probabilities

0    1    2    3

start
verb
noun
end
start 1
verb 0
noun 0
end 0
Token 1: fish

0 1 2 3

start 1 0
verb 0 .2 * .5
noun 0 .8 * .8
end 0 0
Token 1: fish

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Token 2: sleep
(if ‘fish’ is verb)
Token 2: sleep
(if ‘fish’ is verb)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Token 2: sleep
(if ‘fish’ is a noun)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td>.64*.8*.5</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td>.64*.1*.2</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Token 2: sleep
(if ‘fish’ is a noun)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td>.256</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td>.0128</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
Token 2: sleep
take maximum,
set back pointers

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td>.005</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.256</td>
<td>.256</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>.0128</td>
<td>.0128</td>
</tr>
</tbody>
</table>
Token 2: sleep
take maximum,
set back pointers

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>0.1</td>
<td>0.256</td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>0.64</td>
<td>0.0128</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Token 3: end

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td>-</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td>-</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>.256*.7</td>
</tr>
</tbody>
</table>
Token 3: end
take maximum, set back pointers

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td>-</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td>-</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>.256*.7</td>
</tr>
</tbody>
</table>
Decide:
fish = noun
sleep = verb

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td>-</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td>-</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>.256*.7</td>
</tr>
</tbody>
</table>
POS taggers

- Brill’s tagger
  - http://www.cs.jhu.edu/~brill/
- TnT tagger
  - http://www.coli.uni-saarland.de/~thorsten/tnt/
- Stanford tagger
- SVMTool
  - http://www.lsi.upc.es/~nlp/SVMTool/
- GENIA tagger
  - http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/
- More complete list at:
  http://www-nlp.stanford.edu/links/statnlp.html#Taggers
Outline

• Query Expansion and Relevance Feedback
• POS Tagging and HMM
• Formal Grammars
  • Context-free grammar
  • Grammars for English
  • Treebanks
• Parsing and CKY Algorithm
Syntax

• By grammar, or syntax, we have in mind the kind of implicit knowledge of your native language that you had mastered by the time you were 3 years old without explicit instruction

• Not the kind of stuff you were later taught in “grammar” school
Syntax

- Why should you care?
- Grammars (and parsing) are key components in many applications
  - Grammar checkers
  - Dialogue management
  - Question answering
  - Information extraction
  - Machine translation
Syntax

• Key notions that we’ll cover
  • Constituency
  • Grammatical relations and Dependency
    • Heads
• Key formalism
  • Context-free grammars
• Resources
  • Treebanks
Constituency

• The basic idea here is that groups of words within utterances can be shown to act as single units.
• And in a given language, these units form coherent classes that can be shown to behave in similar ways
  • With respect to their internal structure
  • And with respect to other units in the language
Constituency

• Internal structure
  • We can describe an internal structure to the class (might have to use disjunctions of somewhat unlike sub-classes to do this).

• External behavior
  • For example, we can say that noun phrases can come before verbs
Constituency

- For example, it makes sense to the say that the following are all *noun phrases* in English...

| Harry the Horse | a high-class spot such as Mindy’s |
| the Broadway coppers | the reason he comes into the Hot Box |
| they | three parties from Brooklyn |

verbs.
- This is external evidence
Grammars and Constituency

- Of course, there’s nothing easy or obvious about how we come up with right set of constituents and the rules that govern how they combine...
- That’s why there are so many different theories of grammar and competing analyses of the same data.
- The approach to grammar, and the analyses, adopted here are very generic (and don’t correspond to any modern linguistic theory of grammar).
Context-Free Grammars

- Context-free grammars (CFGs)
  - Also known as
    - Phrase structure grammars
    - Backus-Naur form

- Consist of
  - Rules
  - Terminals
  - Non-terminals
Context-Free Grammars

- **Terminals**
  - We’ll take these to be words (for now)

- **Non-Terminals**
  - The constituents in a language
    - Like noun phrase, verb phrase and sentence

- **Rules**
  - Rules are equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right.
Some NP Rules

• Here are some rules for our noun phrases

\[
NP \rightarrow Det\ Nominal \\
NP \rightarrow ProperNoun \\
Nominal \rightarrow Noun \mid Nominal\ Noun
\]

• Together, these describe two kinds of NPs.
  • One that consists of a determiner followed by a nominal
  • And another that says that proper names are NPs.
  • The third rule illustrates two things
    • An explicit disjunction
      • Two kinds of nominals
    • A recursive definition
      • Same non-terminal on the right and left-side of the rule
<table>
<thead>
<tr>
<th>Grammar Rules</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S</strong> → <strong>NP</strong> <strong>VP</strong></td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td><strong>NP</strong> → <strong>Pronoun</strong></td>
<td>I</td>
</tr>
<tr>
<td><strong>Proper-Noun</strong></td>
<td>Los Angeles</td>
</tr>
<tr>
<td><strong>Det Nominal</strong></td>
<td>a + flight</td>
</tr>
<tr>
<td><strong>Nominal</strong> → <strong>Nominal Noun</strong></td>
<td>morning + flight</td>
</tr>
<tr>
<td><strong>Noun</strong></td>
<td>flights</td>
</tr>
<tr>
<td><strong>VP</strong> → <strong>Verb</strong></td>
<td>do</td>
</tr>
<tr>
<td><strong>Verb NP</strong></td>
<td>want + a flight</td>
</tr>
<tr>
<td><strong>Verb NP PP</strong></td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td><strong>Verb PP</strong></td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td><strong>PP</strong> → <strong>Preposition NP</strong></td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>
Derivations

- A derivation is a sequence of rules applied to a string that accounts for that string.
  - Covers all the elements in the string.
  - Covers only the elements in the string.
Definition

• More formally, a CFG consists of

- $N$ a set of non-terminal symbols (or variables)
- $\Sigma$ a set of terminal symbols (disjoint from $N$)
- $R$ a set of rules or productions, each of the form $A \rightarrow \beta$
  where $A$ is a non-terminal,
  $\beta$ is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
- $S$ a designated start symbol
Parsing

• Parsing is the process of taking a string and a grammar and returning a (multiple?) parse tree(s) for that string
• It is completely analogous to running a finite-state transducer with a tape
  • It’s just more powerful
    • Remember this means that there are languages we can capture with CFGs that we can’t capture with finite-state methods
    • More on this when we get to Ch. 13.
An English Grammar Fragment

- Sentences
- Noun phrases
  - Agreement
- Verb phrases
  - Subcategorization
Sentence Types

- **Declaratives:** *A plane left.*
  \[ S \rightarrow NP \text{ VP} \]

- **Imperatives:** *Leave!*
  \[ S \rightarrow VP \]

- **Yes-No Questions:** *Did the plane leave?*
  \[ S \rightarrow \text{Aux NP VP} \]

- **WH Questions:** *When did the plane leave?*
  \[ S \rightarrow \text{WH-NP Aux NP VP} \]
Noun Phrases

• Let’s consider the following rule in more detail...

\[ NP \rightarrow \text{Det Nominal} \]

• Most of the complexity of English noun phrases is hidden in this rule.

• Consider the derivation for the following example
  • All the morning flights from Denver to Tampa leaving before 10
Noun Phrases

NP
  PreDet
    all
  Det
    the
  Nom
  GerundiveVP
    leaving before 10
      PP
        to Tampa
          PP
            from Denver
              Nom
                Noun
                  flights
              Nom
                morning
NP Structure

• Clearly this NP is really about flights. That’s the central critical noun in this NP. Let’s call that the head.
• We can dissect this kind of NP into the stuff that can come before the head, and the stuff that can come after it.
Determiners

- Noun phrases can start with determiners...
- Determiners can be
  - Simple lexical items: the, this, a, an, etc.
    - A car
  - Or simple possessives
    - John’s car
  - Or complex recursive versions of that
    - John’s sister’s husband’s son’s car
Nominals

- Contains the head and any pre- and post- modifiers of the head.
  - Pre-
    - Quantifiers, cardinals, ordinals...
      - Three cars
    - Adjectives and Aps
      - large cars
    - Ordering constraints
      - Three large cars
      - ?large three cars
Postmodifiers

• Three kinds
  • Prepositional phrases
    • From Seattle
  • Non-finite clauses
    • Arriving before noon
  • Relative clauses
    • That serve breakfast

• Same general (recursive) rule to handle these
  • $\text{Nominal} \rightarrow \text{Nominal PP}$
  • $\text{Nominal} \rightarrow \text{Nominal GerundVP}$
  • $\text{Nominal} \rightarrow \text{Nominal RelClause}$
Agreement

- By *agreement*, we have in mind constraints that hold among various constituents that take part in a rule or set of rules.
- For example, in English, determiners and the head nouns in NPs have to agree in their number.

This flight
Those flights

*This flights
*Those flight
The Point

• CFGs appear to be just about what we need to account for a lot of basic syntactic structure in English.

• But there are problems
  • That can be dealt with adequately, although not elegantly, by staying within the CFG framework.

• There are simpler, more elegant, solutions that take us out of the CFG framework (beyond its formal power)
  • LFG, HPSG, Construction grammar, XTAG, etc.
  • Chapter 15 explores the unification approach in more detail
Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree (presumably the right one).
- These are generally created
  - By first parsing the collection with an automatic parser
  - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.
Penn Treebank

- Penn TreeBank is a widely used treebank.

- Most well known is the Wall Street Journal section of the Penn TreeBank.
Treebank Grammars

- Treebanks implicitly define a grammar for the language covered in the treebank.
- Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar.
- Not complete, but if you have decent size corpus, you’ll have a grammar with decent coverage.
Treebank Grammars

• Such grammars tend to be very flat due to the fact that they tend to avoid recursion.
  • To ease the annotators burden
• For example, the Penn Treebank has 4500 different rules for VPs. Among them...

\[
\begin{align*}
\text{VP} & \rightarrow \text{VBD} \quad \text{PP} \\
\text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \\
\text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \quad \text{PP} \\
\text{VP} & \rightarrow \text{VBD} \quad \text{PP} \quad \text{PP} \quad \text{PP} \quad \text{PP}
\end{align*}
\]
Heads in Trees

- Finding heads in treebank trees is a task that arises frequently in many applications.
  - Particularly important in statistical parsing
- We can visualize this task by annotating the nodes of a parse tree with the heads of each corresponding node.
Lexically Decorated Tree

```
S(dumped)
  /    |
NP(workers) VP(dumped)
  /      |
NNS(workers) VBD(dumped)
    /     |
    workers dumped
```

```
  |
VP(dumped)
  |
NP(sacks)
  |
NNS(sacks)
  |
P(into)
  |
DT(a)
  |
NN(bin)
```
Head Finding

- The standard way to do head finding is to use a simple set of tree traversal rules specific to each non-terminal in the grammar.
Noun Phrases
Treebank Uses

- Treebanks (and headfinding) are particularly critical to the development of statistical parsers
  - Chapter 14
- Also valuable to *Corpus Linguistics*
  - Investigating the empirical details of various constructions in a given language
Summary

• Context-free grammars can be used to model various facts about the syntax of a language.
• When paired with parsers, such grammars constitute a critical component in many applications.
• Constituency is a key phenomena easily captured with CFG rules.
  • But agreement and subcategorization do pose significant problems
• Treebanks pair sentences in corpus with their corresponding trees.
For Now

• Assume…
  • You have all the words already in some buffer
  • The input isn’t POS tagged
  • We won’t worry about morphological analysis
  • All the words are known

• These are all problematic in various ways, and would have to be addressed in real applications.
Top-Down Search

- Since we’re trying to find trees rooted with an $S$ (Sentences), why not start with the rules that give us an $S$.
- Then we can work our way down from there to the words.
Top Down Space
Bottom-Up Parsing

- Of course, we also want trees that cover the input words. So we might also start with trees that link up with the words in the right way.
- Then work your way up from there to larger and larger trees.
Bottom-Up Search

Book that flight
Bottom-Up Search
Bottom-Up Search

```
Nominal
  Verb Det Noun
    Book that flight
```
Bottom-Up Search
Bottom-Up Search

```
VP
  /   
NP  Nominal
   /   
Verb Book
   /   Det that
   /   Noun flight
```
Top-Down and Bottom-Up

• Top-down
  • Only searches for trees that can be answers (i.e. S’s)
  • But also suggests trees that are not consistent with any of the words

• Bottom-up
  • Only forms trees consistent with the words
  • But suggests trees that make no sense globally
Control

• Of course, in both cases we left out how to keep track of the search space and how to make choices
  • Which node to try to expand next
  • Which grammar rule to use to expand a node

• One approach is called backtracking.
  • Make a choice, if it works out then fine
  • If not then back up and make a different choice
Problems

• Even with the best filtering, backtracking methods are doomed because of two inter-related problems
  • Ambiguity
  • Shared subproblems
Ambiguity

Tree 1:
S
  NP
    Pronoun
    I
  VP
    Verb
    shot
   NP
     Det
     an
    Nominal
  PP
    Nominal
    in my pajamas
    Noun
    elephant

Tree 2:
S
  NP
    Pronoun
    I
  VP
    Verb
    shot
   NP
     Det
     an
    Nominal
  PP
    in my pajamas
    Noun
    elephant
Shared Sub-Problems

- No matter what kind of search (top-down or bottom-up or mixed) that we choose.
  - We don’t want to redo work we’ve already done.
  - Unfortunately, naïve backtracking will lead to duplicated work.
Shared Sub-Problems

- Consider
  - A flight from Indianapolis to Houston on TWA
Shared Sub-Problems

• Assume a top-down parse making choices among the various Nominal rules.

• In particular, between these two
  • Nominal -> Noun
  • Nominal -> Nominal PP

• Statically choosing the rules in this order leads to the following bad results...
Shared Sub-Problems
Shared Sub-Problems

```
NP
  Det  Nominal
    a    
     |     
     Nominal  PP
       |       
       Noun  
         |  from Indianapolis...
         flight
```
Shared Sub-Problems
Shared Sub-Problems
Dynamic Programming

- DP search methods fill tables with partial results and thereby
  - Avoid doing avoidable repeated work
  - Solve exponential problems in polynomial time (well, no not really)
  - Efficiently store ambiguous structures with shared sub-parts.
- We’ll cover the CKY algorithm
CKY Parsing

• First we’ll limit our grammar to epsilon-free, binary rules (more later)

• Consider the rule $A \rightarrow BC$
  • If there is an A somewhere in the input then there must be a B followed by a C in the input.
  • If the A spans from $i$ to $j$ in the input then there must be some $k$ st. $i<k<j$
    • Ie. The B splits from the C someplace.
Problem

- What if your grammar isn’t binary?
  - As in the case of the TreeBank grammar?
- Convert it to binary… any arbitrary CFG can be rewritten into Chomsky-Normal Form automatically.
- What does this mean?
  - The resulting grammar accepts (and rejects) the same set of strings as the original grammar.
  - **But** the resulting derivations (trees) are different.
Problem

• More specifically, we want our rules to be of the form

\[ A \rightarrow B \ C \]

Or

\[ A \rightarrow w \]

That is, rules can expand to either 2 non-terminals or to a single terminal.
Binarization Intuition

- Eliminate chains of unit productions.
- Introduce new intermediate non-terminals into the grammar that distribute rules with length > 2 over several rules.

So... \( S \rightarrow A \ B \ C \) turns into
\( S \rightarrow X \ C \) and
\( X \rightarrow A \ B \)

Where \( X \) is a symbol that doesn’t occur anywhere else in the grammar.
## Sample L1 Grammar

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>$Det \rightarrow that</td>
</tr>
<tr>
<td>$S \rightarrow Aux\ NP\ VP$</td>
<td>$Noun \rightarrow book</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>$Verb \rightarrow book</td>
</tr>
<tr>
<td>$NP \rightarrow Pronoun$</td>
<td>$Pronoun \rightarrow I</td>
</tr>
<tr>
<td>$NP \rightarrow Proper-Noun$</td>
<td>$Proper-Noun \rightarrow Houston</td>
</tr>
<tr>
<td>$NP \rightarrow Det\ Nominal$</td>
<td>$Aux \rightarrow does$</td>
</tr>
<tr>
<td>Nominal $\rightarrow Noun$</td>
<td>$Preposition \rightarrow from</td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal\ Noun$</td>
<td></td>
</tr>
<tr>
<td>Nominal $\rightarrow Nominal\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ NP\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow Verb\ PP$</td>
<td></td>
</tr>
<tr>
<td>$VP \rightarrow VP\ PP$</td>
<td></td>
</tr>
<tr>
<td>$PP \rightarrow Preposition\ NP$</td>
<td></td>
</tr>
</tbody>
</table>
## CNF Conversion

<table>
<thead>
<tr>
<th>Grammar</th>
<th>$L_1$ in CNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \to NP\ VP$</td>
<td>$S \to NP\ VP$</td>
</tr>
<tr>
<td>$S \to Aux\ NP\ VP$</td>
<td>$S \to X1\ VP$</td>
</tr>
<tr>
<td></td>
<td>$X1 \to Aux\ NP$</td>
</tr>
<tr>
<td>$S \to VP$</td>
<td>$S \to book\</td>
</tr>
<tr>
<td></td>
<td>$S \to Verb\ NP$</td>
</tr>
<tr>
<td></td>
<td>$S \to X2\ PP$</td>
</tr>
<tr>
<td></td>
<td>$S \to Verb\ PP$</td>
</tr>
<tr>
<td></td>
<td>$S \to VP\ PP$</td>
</tr>
<tr>
<td>$NP \to Pronoun$</td>
<td>$NP \to I\</td>
</tr>
<tr>
<td>$NP \to Proper-Noun$</td>
<td>$NP \to TWA\</td>
</tr>
<tr>
<td>$NP \to Det\ Nominal$</td>
<td>$NP \to Det\ Nominal$</td>
</tr>
<tr>
<td>Nominal $\to Noun$</td>
<td>Nominal $\to book\</td>
</tr>
<tr>
<td>Nominal $\to Nominal\ Noun$</td>
<td>Nominal $\to Nominal\ Noun$</td>
</tr>
<tr>
<td>Nominal $\to Nominal\ PP$</td>
<td>Nominal $\to Nominal\ PP$</td>
</tr>
<tr>
<td>$VP \to Verb$</td>
<td>$VP \to book\</td>
</tr>
<tr>
<td>$VP \to Verb\ NP$</td>
<td>$VP \to Verb\ NP$</td>
</tr>
<tr>
<td>$VP \to Verb\ NP\ PP$</td>
<td>$VP \to X2\ PP$</td>
</tr>
<tr>
<td></td>
<td>$X2 \to Verb\ NP$</td>
</tr>
<tr>
<td>$VP \to Verb\ PP$</td>
<td>$VP \to Verb\ PP$</td>
</tr>
<tr>
<td>$VP \to VP\ PP$</td>
<td>$VP \to VP\ PP$</td>
</tr>
<tr>
<td>$PP \to Preposition\ NP$</td>
<td>$PP \to Preposition\ NP$</td>
</tr>
</tbody>
</table>
CKY

- So let’s build a table so that an $A$ spanning from $i$ to $j$ in the input is placed in cell $[i,j]$ in the table.
- So a non-terminal spanning an entire string will sit in cell $[0, n]$
  - Hopefully an $S$
- If we build the table bottom-up, we’ll know that the parts of the $A$ must go from $i$ to $k$ and from $k$ to $j$, for some $k$. 
Meaning that for a rule like $A \rightarrow B \ C$ we should look for a
$B$ in $[i,k]$ and a $C$ in $[k,j]$.

In other words, if we think there might be an $A$ spanning $i,j$
in the input… AND

$A \rightarrow B \ C$ is a rule in the grammar THEN

There must be a $B$ in $[i,k]$ and a $C$ in $[k,j]$ for some $i<k<j$
CKY

- So to fill the table loop over the cell[i,j] values in some systematic way
  - What constraint should we put on that systematic search?

- For each cell, loop over the appropriate k values to search for things to add.
Example
Example

Filling column 5
Example

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1]</td>
<td>S, VP, Verb, Nominal, Noun</td>
<td>[0,2]</td>
<td>[0,3]</td>
<td>[0,4]</td>
<td>[0,5]</td>
</tr>
<tr>
<td>[1,2]</td>
<td>Det</td>
<td>NP</td>
<td>[1,3]</td>
<td>[1,4]</td>
<td>NP</td>
</tr>
<tr>
<td>[3,4]</td>
<td></td>
<td></td>
<td>[3,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NP, Proper-Noun</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1]</td>
<td>S, VP, Verb, Nominal, Noun</td>
<td>[0,2]</td>
<td>S, VP, X2</td>
<td>[0,3]</td>
<td>[0,4]</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, VP, Verb, Nominal, Noun [0,1]</td>
<td>S, VP, X2 [0,3]</td>
<td></td>
</tr>
<tr>
<td>Det [1,2]</td>
<td>NP [1,3]</td>
<td></td>
</tr>
<tr>
<td>Nominal, Noun [2,3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP, Proper-Noun [4,5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example
To formalize it: CKY Algorithm

```plaintext
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    table[j - 1, j] ← \{ A | A → words[j] ∈ grammar \}
for i ← from j - 2 downto 0 do
    for k ← i + 1 to j - 1 do
        table[i, j] ← table[i, j] ∪ \{ A | A → BC ∈ grammar, B ∈ table[i, k], C ∈ table[k, j] \}
```

Exercises

• Try to parse the following sentence:
• I prefer meal in flight.
Take-home Messages

• Context-free grammars can be used to model various facts about the syntax of a language.
• When paired with parsers, such grammars constitute a critical component in many applications.
• Constituency is a key phenomena easily captured with CFG rules.
  • But agreement and subcategorization do pose significant problems
• CKY is a bottom-up dynamic programming algorithm
• We can convert CFG rules into CNF forms