Recap from last week

- Syntax specifies a relationship between symbols (words)
  - This can be captured in a grammar
- Natural Language is syntactically ambiguous
  - "The president spoke to the nation about the problem of drug use in the schools from one coast to the other."
  - "They criticized the mess on the beach"
- Parsers assign structure to sentences
- but they can’t decide between competing parses
- POS Taggers can make initial first guesses to save wasted time.

Tagging

- Tagging involves making a shallow (and fast) first guess
- Tagging is limited to "local" information -
  - Word Senses
  - Word Stemmers (and Lemmas)
  - Sentence Splitting
  - Parts of Speech
- We can think of POS Tagging as part of pre-processing like
  - Acronyms
  - Capitalisation
  - Name Identification etc.
  - Spelling correction?
Parts of Speech

- Every word belongs to a single syntactic category
  - Nouns, verbs, aux, det etc.

- The next POS can be predicted by looking at the last two or three POS tags
  
  The horse won the race
  Det Noun Verb Det ????

  The goat gobbered
  Det Noun ???????

  Notice we could also guess POS by looking at morphology

Tag Sets

- Tag sets vary in size
  - 8 Tags for Ancient Greek (c. 100 b.c.)
  - 87 Tags for the Brown Corpus (1970's American English)
  - 45 Tags for the Penn Treebank
  - 61 for CS Claws (Used by the BNC and an UK standard tagset for NLP)

- Disagreements (in English) on
  - Verbs – how many different types
  - Compound Nouns

- Tag Sets vary according to the Language
  - Arabic Taggers often feature 130 plus tags
  - (but this depends on morphology)

- Accuracy depends on the size of the tag set

Stochastic Tagging

- Use a HMM and to learn probabilities for sequences of POS tags
  - Hang tag a corpus of words with their POS tags
  - Generate n-grams for all word sequences (uni-, bi- & trigrams)
  - Count frequency of POS tag sequences
  - Use this to generate HMM (using Forward-Backward algorithm)

- Given an ambiguous word, choose tag which maximises
  \[ P(\text{word}|\text{tag}) \times P(\text{tag}|\text{previous N tags}) \]

Brill Taggers

- Developed by Eric Brill in the 1990s
- Non-Stochastic – uses Transformation Based Learning (TBL)

- The Set Up
  - Generate all possible rules from a small set of templates
  - If X is preceded by Y then change X to Z
  - Take a hand tagged corpus – use 80% as a training corpus/gold standard (GS)

- Training
  1) Tag every word with its most likely POS tag
  2) Select the template rule which improves the training corpus the most
  3) Apply new rule to entire corpus
  4) Goto Step 2 unless no further improvement from last iteration compared to GS
POS Tagging Results

- POS Taggers are pretty easy to implement (given enough data)
  - Qtag ~70-80% (my estimate)
  - Stochastic and Brill Taggers have the same level of performance
- Professional level performance is harder to achieve
  - ~90-95% accuracy (depending on tag set)
  - May use additional sources of information (or hacks)
  - Several good taggers are commercially available (e.g. BNC tagger)
  - Caveat Emptor – they still make mistakes. Check a corpus before purchase
    - (e.g. BNC compound nouns are usually wrongly tagged)

How hard is POS Tagging?

- Brown Corpus
  - ~10% of words are ambiguous
  - Unambiguous words (1 tag) 35,340
  - 2-7 tag words (4,100)
  - 3 tags (264)
  - 4 tags (61)
  - 7 tags (1 – still)
  - But this doesn’t factor in word frequency
- Professional level performance is harder to achieve
  - ~90-95% accuracy (depending on tag set)
  - May use additional sources of information (or hacks)
  - Several good taggers are commercially available (e.g. BNC tagger)
  - Caveat Emptor – they still make mistakes. Check a corpus before purchase
    - (e.g. BNC compound nouns are usually wrongly tagged)

Human preferences in Syntax Parsing

- Late closure
  - New words or phrases are attached to the current clause
    - John said he would leave yesterday
- Minimal Attachment
  - Prefer to attach to an existing node rather than propose a new one
    - Predicts
      - I baked a cake for the kids
      - We painted the walls with cracks (as a garden path)
    - Counterexample “We painted the walls for Christmas”
- Right Association
  - Prefer to attach to the right most existing node in the tree
    - Predicts
      - Sarah noted that Emily had written in her diary
      - I believe he loves me with all his heart
      - No obvious counterexamples

Probabilistic Context Free Grammars

- Basic Idea
  - Augment every context free rule with a probability
- P(A -> B [p])
  - P(A -> B)
    - OR
    - Conditional Probability P(A -> B | A)
- Example
  - NP -> Det Nom [.20]
  - NP -> Proper-Noun [.35]
  - NP -> Nom [0.05]
  - NP -> Pronoun [.40]
Sentence Probabilities

- given a PCFG we can estimate
- the probability of a particular parse tree as:

\[ P(T, S) = \prod_{n \in T} p(r(n)) \]

where \( p(r(n)) \) is the probability of every rule used to generate every node in the parse tree.

- \( PT \) is the joint probability of the parse and the sentence
- and also the probability of the parse

\[ P(T, S) = P(T)P(S|T) \]

- but the parse tree contains all the words in the sentence (by definition)

\[ P(S|T) = 1 \]

\[ P(T, S) = P(T)P(S|T) \]

\[ P(T) \]

A trivial grammar

- \( S \rightarrow NP\ VP \ [0.8] \)
- \( S \rightarrow Aux\ NP\ VP \ [0.2] \)
- \( NP \rightarrow \text{Det}\ Noun \ [0.3] \)
- \( NP \rightarrow \text{Pnoun} \ [0.3] \)
- \( NP \rightarrow \text{Det}\ Adj\ Noun \ [0.1] \)
- \( NP \rightarrow \text{NP}\ PP \ [0.3] \)
- \( VP \rightarrow \text{V}\ NP \ [0.6] \)
- \( VP \rightarrow \text{V}\ NP\ PP \ [0.4] \)
- \( PP \rightarrow \text{P}\ NP \ [1] \)

- \( N \rightarrow \{dogs\} \ [0.25] \)
- \( N \rightarrow \{beach\} \ [0.25] \)
- \( N \rightarrow \{fish\} \ [0.25] \)
- \( N \rightarrow \{fork\} \ [0.1] \)
- \( N \rightarrow \{telescope\} \ [0.1] \)
- \( \text{Pnoun} \rightarrow \{they\} \ [0.6] \)
- \( \text{Pnoun} \rightarrow \{she\} \ [0.4] \)
- \( V \rightarrow \{ate\} \ [0.5] \)
- \( V \rightarrow \{kept\} \ [0.5] \)
- \( \text{P} \rightarrow \{with\} \ [0.75] \)
- \( \text{P} \rightarrow \{on\} \ [0.25] \)
- \( \text{Det} \rightarrow \{the\} \ [0.25] \)
- \( \text{Det} \rightarrow \{a\} \ [0.6] \)

An ambiguous sentence

They kept the dogs on the beach

They kept the dogs on the beach

Working through the probabilities

- They kept (the dogs on the beach)

\[ P(T1) = 0.6 \times 0.5 \times 0.4 \times 0.25 \times 0.25 \times 0.6 \times 0.25 \times 0.3 \times 0.3 \times 0.3 \times 0.3 \times 1 \times 0.8 \]

\[ = 7.2 \times 10^{-6} \]

- They (kept the dogs on the beach)

\[ P(T2) = 0.6 \times 0.5 \times 0.4 \times 0.25 \times 0.25 \times 0.4 \times 0.25 \times 0.3 \times 0.3 \times 0.3 \times 0.3 \times 1 \times 0.8 \]

\[ = 1.62 \times 10^{-5} \]

- \( P(T2) > P(T1) \) and therefore is the most likely interpretation.
General Disambiguation

- Disambiguation
- take the best parse tree $T$ for a sentence $S$ out of the set of parse trees $t(S)$
  
  $$T(S) = \arg \max_{T \in t(S)} P(T|S)$$
  
  But recall that $P(T,S) = P(T)$
  
  $$T(S) = \arg \max_{T} P(T)$$
  
  i.e. always pick the parse with the highest probability

Language Models

- PCFGs can also be used to assign a probability to the string of words constituting a sentence
- This can be used in Language Modeling
  - speech recognition
  - spell correction
  - text prediction.
- The probability of an unambiguous sentence is $P(T,S) = P(T)$
- The probability of an ambiguous sentence is the sum of the probabilities
  
  $$P(S) = \sum_{T \in t(S)} P(T,S)$$
  
  $$= \sum_{T \in t(S)} P(T)$$

CYK Parsing

- CKY parsing was introduced last week
- CKY parsing involves building up structure bottom up using a chart or table.
  The chart is used to retain sub-structure and to avoid the need to constantly rebuild structure from failed parses
- CKY parsing is very efficient $O(n^3)$
- Requires a grammar in Chomsky Normal Form

- $A \to B \ C$
- or
- $A \to a$

Data Structures

- Input
  - PCFG $(N, \Sigma, P, S, D)$
    - $N$ = a set of non-terminals with indices 1 ... $N$
    - $\Sigma$ = a set of terminals (words)
    - $P$ = a set of productions $A \to B \mid A \to a$
    - A start symbol $S$
    - $D$ = a function assigning probabilities to each rule in $P$
  - A set of words $w_1 \ldots w_n$
- Data structure is a dynamic programming array $\pi[i, j, a]$ containing maximum probability for a constituent spanning $i$ & $j$
- Output
  - Maximum probability parse will be $\pi[1, n, 1]$
Pseudocode: set up & base case

function CYK (words, grammar) returns best parse
Create and clear p(num_words, num_words, num_nonterminals)

# base case
for i to num_words
  for A = 1 to num_nonterminals
    if A -> wi is in Grammar then
      \( \pi[i, i, a] = P(A \rightarrow w_i) \)

i.e. tag every word with each possible A -> a rule & its probability

Recursive case

# recursive case for j = 2 to num_words
for i = 1 to num_words-j + 1
  for k = 1 to j-1
    for A = 1 to num_nonterminals
      for B = 1 to num_nonterminals
        for C = 1 to num_nonterminals
          prob = \( \pi[i, k, B] \times p[i+k, j-k, C] \times P(A \rightarrow BC) \)
          if (prob > \( \pi[i, j, A] \)) then
            \( \pi[i, j, A] = \text{prob} \)
            \( B[i, j, A] = [k, A, B] \)

B is the array of back pointers used to recover the best parse

* (Taken directly from Jurafsky & Martin page 455)

Learning PCFG Probabilities

- Where do the probabilities come from?
  - Preparsed Corpora (Treebanks)
    - \( P(A \rightarrow B|A) = \frac{\text{Count}(A \rightarrow B)}{\text{Count}(A \rightarrow B) + \sum \text{Count}(A \rightarrow Y)} \)

Penn Treebank (1993) Parse Trees from the Brown Corpus, WSJ, Switchboard & Atis (http://www.cis.upenn.edu/~treebank/)

Alternative is to parse a corpus with a pre-existing grammar and count each partial parse using the Inside Outside algorithm (See Manning and Schutze (1999))

Limitations with PCFGs

- PCFGs assume naïve independence
  - NP1 (VP NP2)
  - NP1 & NP2 are statistically independent

- Position-dependence
  - In Switchboard
    - 91% of subjects are pronouns
    - "She's able to take her baby to work"
  - 66% of objects are non-prominal lexical NPs
    - "All the people signed confessions"

- Lexical Dependence
  - "Moscow sent more than 100,000 soldiers into Afghanistan"
  - Penn Treebank
    - VP "Sent" always attaches to prepositional phrase "into"
Lexically Driven PCFGs

- **Solution**
  - Every NT node has a “head”
  - Typically the main noun, verb or preposition
  - “Workers dumped sacks into a bin”

- **Key idea**
  - Associate a probability with “head” which modifies the node probability
  - We can capture dependencies by conditioning the head node based on its mother node.
  - \[ P(T,S) = \prod_{n \in T} p(r(n)|n,h(n)) \times p(h(n)|n,h(m(n))) \]
    - \( h(n) \) = head of node 
    - \( h(m(n)) \) = head of mother node from Charniak 1997
    - (see J & M p458-463 for more discussion)

Evaluation

- **Parseval (1991 - ongoing)**
- **Some Metrics**
  - **Labelled Recall** = \# of correct constituents in candidate parse of \( S \)
    - \# of correct constituents in treebank parse of \( S \)
  - **Labelled Precision** = \# of correct constituents in candidate parse of \( S \)
    - \# of total constituents in candidate parse of \( S \)
  - Cross brackets = \# of cross brackets ((A B) C) versus (A (B C))
  - Charniak (1997) just under 90% precision & recall & about 1% cross bracketing

Further Reading

- Chapter 12 of Jurafsky and Martin Speech and Language Processing
  - Pretty good overview and discusses Dependency Grammars which I didn’t cover.

  - Good (slightly dated) and reasonably gentle maths

- Chris Manning & Henrich Schutze Foundations of Statistical Natural Language Processing (1999)
  - Now a standard Text for Statistical NLP – maths heavy so be warned!

- Brigitte Krenn Don’t panic! The Linguists guide to Statistics (1997)
  - Free! http://www.ofai.at/~brigitte.krenn/lit.html