CSCI 5832
Natural Language Processing

Lecture 14

Jim Martin

Today 3/6

- **Review**
- **Partial Parsing**
- **Sequence Labeling/Chunking**

Review

- Last we covered CKY and Earley parsing.
- Both are dynamic programming methods used to build a table that contains all the possible parses for an input given some CFG grammar.

Review

- CKY
  - Bottom up
  - Often used in probabilistic parsing
  - Restricted to Chomsky-Normal form grammars
- Earley
  - Top-Down
  - Accepts arbitrary context-free grammars

Full Syntactic Parsing

- Probably necessary for deep semantic analysis of texts (as we'll see).
- Probably not practical for most applications (given typical resources)
  - O(n^3) for straight parsing
  - O(n^5) for probabilistic versions
  - Too slow for applications that need to process texts in real time (search engines)

Partial Parsing

- For many applications you don't really need a full-blown syntactic parse. You just need a good idea of where the base level syntactic units are.
  - Often referred to as chunks.
- For example, if you're interested in locating all the people, places and organizations in a text it might be useful to know where all the NPs are.

Examples

[$_{NP}$ The morning flight] [$_{PP}$ from] [$_{NP}$ Denver] [$_{VP}$ has arrived.]

[$_{NP}$ a flight] [$_{PP}$ from] [$_{NP}$ Indianapolis][$_{PP}$ to][$_{NP}$ Houston][$_{PP}$ on][$_{NP}$ TWA]

[$_{NP}$ The morning flight] from [$_{NP}$ Denver] has arrived.

- **The first two are examples of full partial parsing or chunking. All of the elements in the text are part of a chunk. And the chunks are non-overlapping.**
- **Note how the second example has no hierarchical structure.**
- **The last example illustrates base-NP chunking. Ignore anything that isn't in the kind of chunk you're looking for.**

Partial Parsing

- **Two approaches**
  - **Rule-based (hierarchical) transduction.**
  - **Statistical sequence labeling**
    - **HMMs**
    - **MEMMs**

Rule-Based Partial Parsing

- Restrict the form of rules to exclude recursion (make the rules flat).
- Group and order the rules so that the RHS of the rules can refer to non-terminals introduced in earlier transducers, but not later ones.
- Combine the rules in a group in the same way we did with the rules for spelling changes.
- Combine the groups into a cascade…
- Then compose, determinize and minimize the whole thing (optional).

Typical Architecture

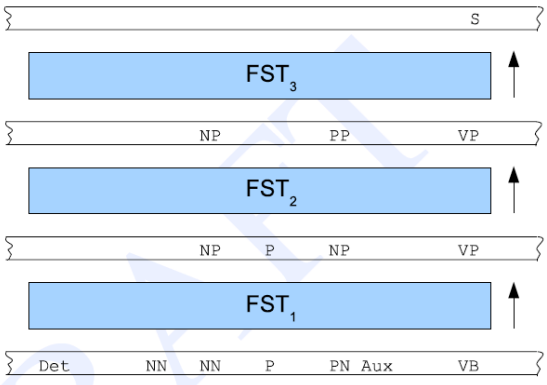
- Phase 1: Part of speech tags
- Phase 2: Base syntactic phrases
- Phase 3: Larger verb and NP groups
- Phase 4: Sentential level rules

Partial Parsing

$$NP \rightarrow (Det)\ Noun^*\ Noun$$
$$NP \rightarrow Proper\text{-}Noun$$
$$VP \rightarrow Verb$$
$$VP \rightarrow Aux\ Verb$$

- **No direct or indirect recursion allowed in these rules.**
- **That is you can't directly or indirectly reference the LHS of the rule on the RHS.**

Cascaded Transducers



The morning flight from Denver has arrived

Partial Parsing

- This cascaded approach can be used to find the sequence of flat chunks you're interested in.
- Or it can be used to approximate the kind of hierarchical trees you get from full parsing with a CFG.

Break

- I'll be back Thursday.
- No office hours today (3/6).

Statistical Sequence Labeling

- As with POS tagging, we can use rules to do partial parsing or we can train systems to do it for us. To do that we need training data and the right kind of encoding.
  - Training data
    - Hand tag a bunch of data (as with POS tagging)
    - Or even better, extract partial parse bracketing information from a treebank.

Encoding

- With the right encoding you can turn the labeled bracketing task into a tagging task. And then proceed exactly as we did with POS Tagging.
- We'll use whats called IOB labeling to do this.
  - I -> Inside
  - O -> Outside
  - B -> Begins

IOB encoding

The morning flight from Denver has arrived.
B_NP I_NP I_NP O B_NP O O

The morning flight from Denver has arrived
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

- The first example shows the encoding for just base-NPs. There are 3 tags in this scheme.
- The second shows full coverage. In this scheme there are 2*N+1 tags. Where N is the number of constituents in your set.

Methods

- HMMs
- Sequence Classification
  - Using any kind of standard ML-based classifier.

Evaluation

- Suppose you employ this scheme. What's the best way to measure performance.
- Probably not the per-tag accuracy we used for POS tagging.
  - Why?

•It's not measuring what we care about
•We need a metric that looks at the chunks not the tags

Example

- Suppose we were looking for PP chunks for some reason.
- If the system simple said O all the time it would do pretty well on a per-label basis since most words reside outside any PP.

Precision/Recall/F

- **Precision:**
  - The fraction of chunks the system returned that were right
    - "Right" means the boundaries and the label are correct given some labeled test set.
- **Recall:**
  - The fraction of the chunks that system got from those that it should have gotten.
- **F: Harmonic mean of those two numbers.**

HMM Tagging

- **Same as with POS tagging**
  - Argmax P(T|W) = P(W|T)P(T)
  - The tags are the hidden states
- **Works ok but it isn't great.**
  - The typical kinds of things that we might think would be useful in this task aren't easily squeezed into the HMM model
- **We'd like to be able to make arbitrary features available for the statistical inference being made.**

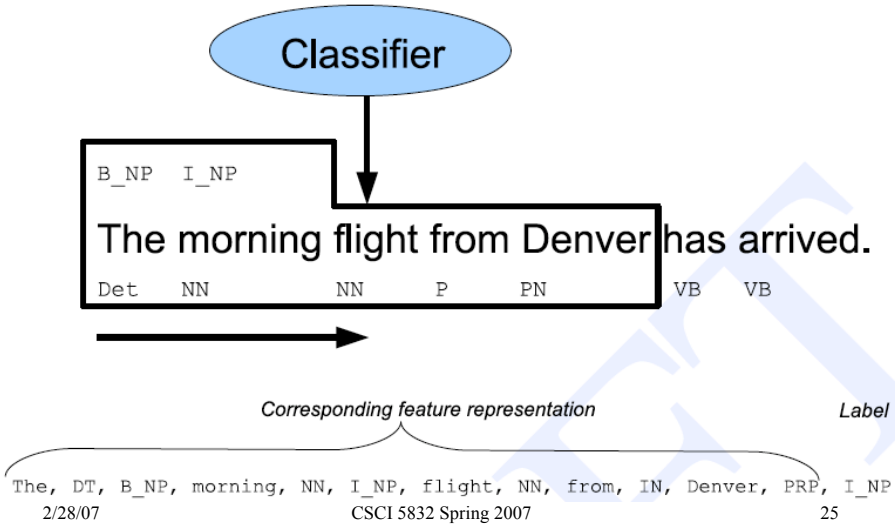
Supervised Classification

- Training a system to take an object represented as a set of features and apply a label to that object.
- Methods typically include
  - Naïve Bayes
  - Decision Trees
  - Maximum Entropy (logistic regression)
  - Support Vector Machines
  - …

Supervised Classification

- Applying this to tagging…
  - The object to be tagged is a word in the sequence
  - The features are features of the word, features of its neighbors, and features derived from the entire sentence.
  - Sequential tagging means sweeping the classifier across the input assigning tags to words as you proceed.

Statistical Sequence Labeling



Typical Features

- **Typical setup involves**
  - **A small sliding window around the object being tagged**
  - **Features extracted from the window**
    - **Current word token**
    - **Previous/next N word tokens**
    - **Current word POS**
    - **Previous/next POS**
    - **Previous N chunk labels**
      - **????**

Performance

- With a decent ML classifier
  - SVMs
  - Maxent
  - Even decision trees
- You can get decent performance with this arrangement.
- Good CONLL 2000 scores had F-measures in the mid-90s.

Problem

- You're making a long series of local judgments. Without attending to the overall goodness of the final sequence of tags. You're just hoping that local conditions will yield global optima.
- Note that HMMs didn't have this problem since the language model worried about the overall goodness of the tag sequence.

Answer

- Graft a language model onto the sequential classification scheme.
  - Instead of having the classifier emit one label as an answer, get it to emit an N-best list for each judgment.
  - Train a language model for the kinds of sequences we're trying to produce.
  - Run viterbi over the N-best lists for the sequence to get the best overall sequence.

MEMMs

- Maximum entropy Markov models are the current standard way of doing this.
  - Although people do the same thing in an ad hoc way with SVMs.
- MEMMs combine two techniques
  - Maximum entropy (logistic) classifiers for the individual labeling
  - Markov models for the sequence model.

Next Time

Next Time

- We're now done with 11 and 12.
- Now go back (before Thursday) and re-read Chapter 6. In particular,
  - Review Viterbi
  - And read sections 6.6, 6.7 and 6.8