Partial Parsing

Slides by James Martin, adapted by Diana Inkpen
for CSI 5386 @ uOttawa

Full Syntactic Parsing

- Probably necessary for deep semantic analysis of texts.
- Probably not practical for many 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)
  - Or that need to deal with large volumes of new material over short periods of time

Two Alternatives

- Partial parsing
  - Approximate phrase-structure parsing with finite-state and statistical approaches
- Dependency parsing
  - Change the underlying grammar formalism
- Both of these approaches give up something (syntactic structure) in return for more robust and efficient parsing

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 syntactic units are.
  - Often referred to as chunks.
- For example, if you’re interested in locating all the people, places and organizations in an English text it can be useful to know where all the NPs are
  - Because that’s where you’ll find the people, places and things

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
- 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

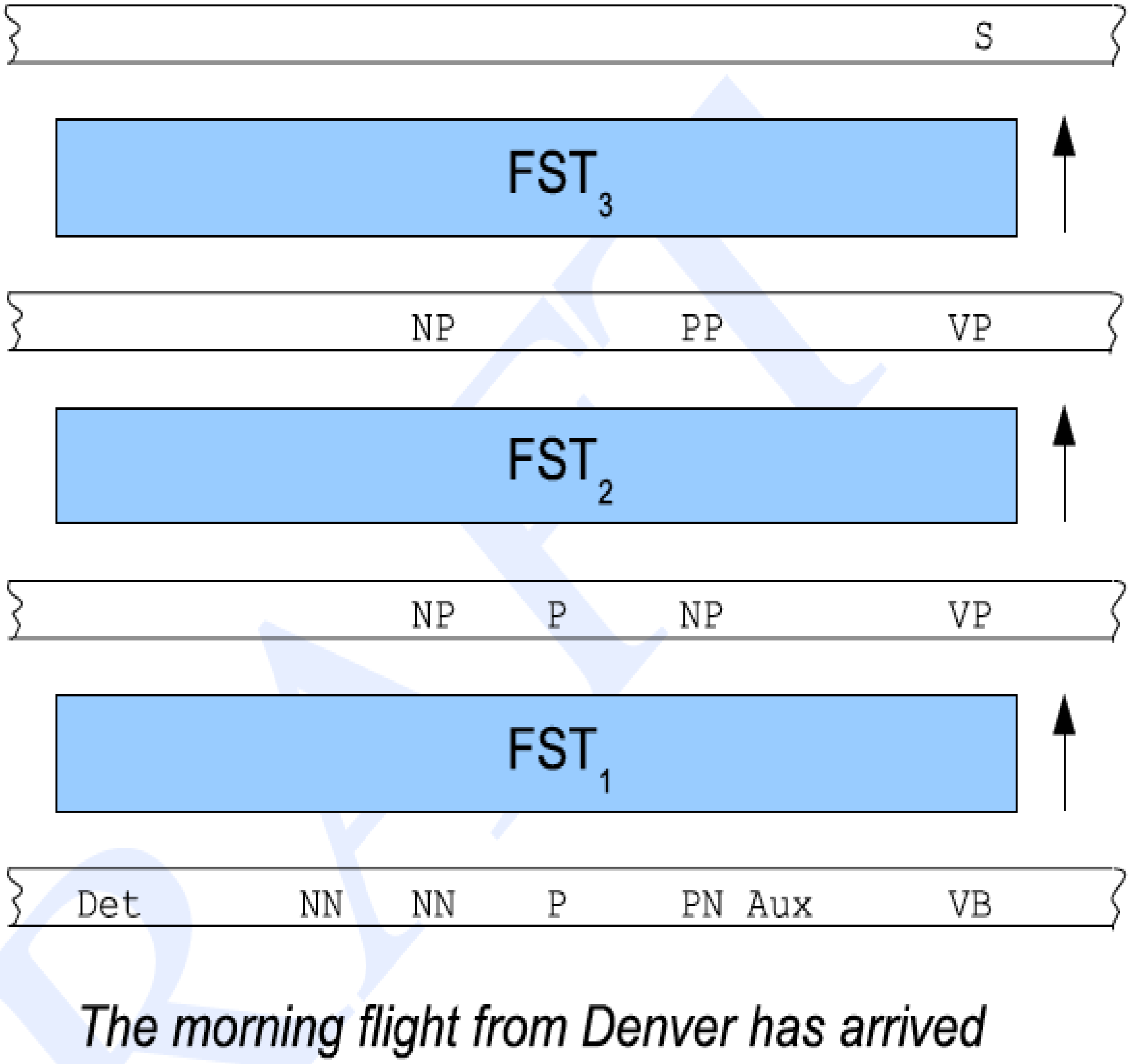
- Phase 1: Part of speech tags
- Phase 2: Base syntactic phrases
- Phase 3: Larger verb and noun groups
- Phase 4: Sentential level rules

Partial Parsing

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

- 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



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.

The Other Way

- An alternative approach is to use statistical machine learning methods to do partial parsing
  - Analogous to the same situation with part-of-speech tagging
    - Rules vs. HMMs

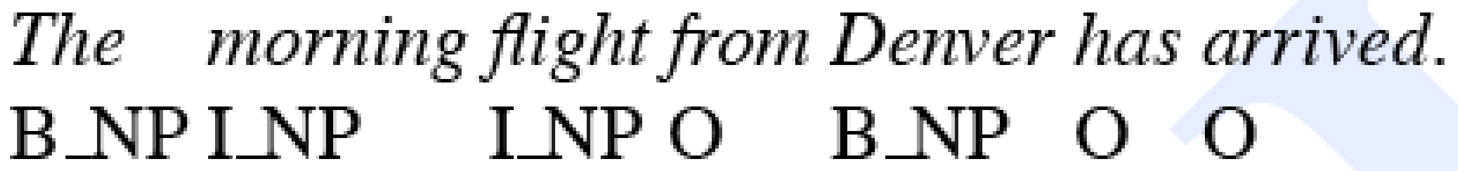
Statistical Sequence Labeling

- 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 a way to view the problem as a classification problem
  - Training data
    - Hand tag a bunch of data (as with POS tagging)
    - Or even better, extract partial parse bracketing information from a treebank.

Encoding

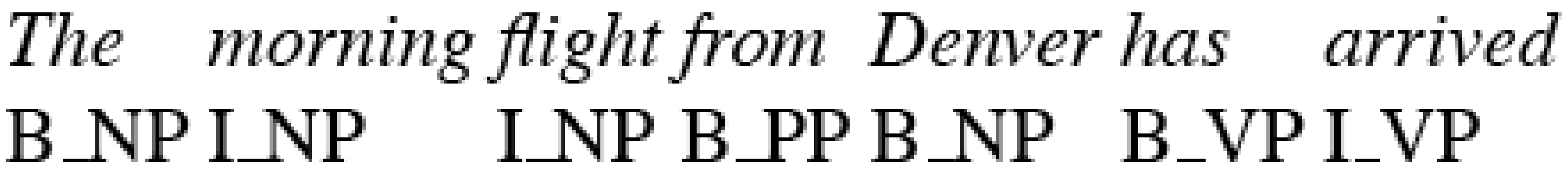
- With the right encoding you can turn any labeled bracketing task into a tagging task. And then proceed exactly as we did with POS Tagging.
- We'll use what's called IOB labeling to do this
  - I -> Inside
  - O -> Outside
  - B -> Begin

IOB encoding



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

- This example shows the encoding for just base-NPs. There are 3 tags in this scheme.



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

- This example shows full coverage. In this scheme there are 2*N+1 kinds of tags. Where N is the number of constituents in your set.

Methods

- Argmax P(Tags|Words)
  - HMMs
  - Discriminative Sequence Classification
    - Using any kind of standard ML-based classifier.

HMM Tagging

- Same as we did with POS tagging
  - Argmax P(T|W) = P(W|T)P(T)
  - The tags are the hidden states
- Works ok, but has one significant shortcoming
  - 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.
- For that we'll turn to classifiers created using classical machine learning techniques

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
  - Logistic regression (maximum entropy)
  - Support Vector Machines
  - …

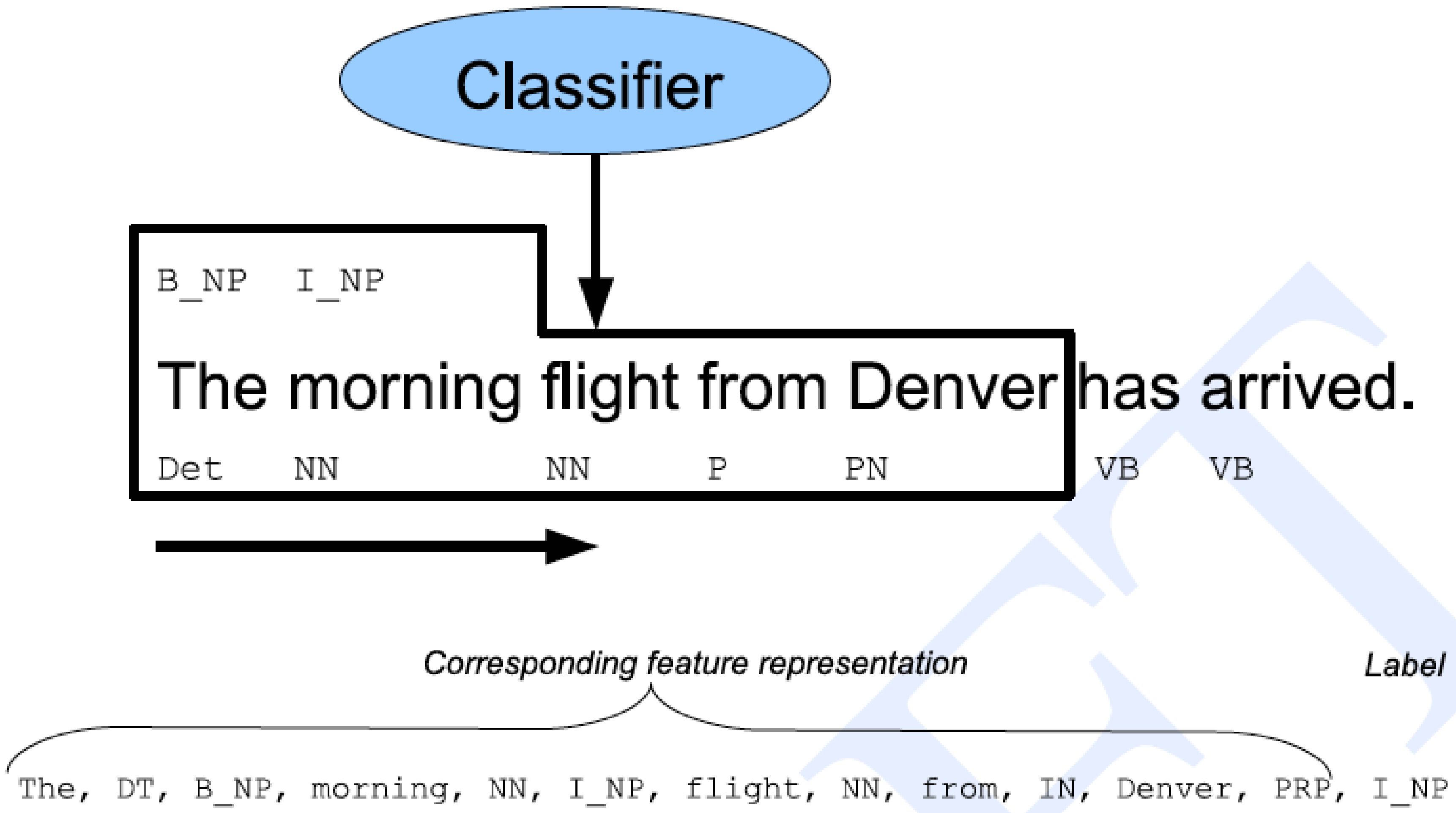
From Classification to Sequence Processing

- 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 immediate neighbors,
    - and features derived from the entire context
  - Sequential tagging means sweeping a classifier across the input assigning tags to words as you proceed.

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
    - Capitalization information
    - ...

Statistical Sequence Labeling


Classifier
B_NP I_NP
The morning flight from Denver has arrived.
Det NN NN P PN VB VB
Corresponding feature representation
Label
The, DT, B_NP, morning, NN, I_NP, flight, NN, from, IN, Denver, PRP, I_NP

Evaluation

- Suppose you employ this IOB 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 simply 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: Simple harmonic mean of those two numbers.

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 goodness.
- Note that HMMs didn’t have this problem since the language model worried about the overall goodness of the tag sequence.
  - But we don’t want to use HMMs since we can’t easily squeeze arbitrary features into the

Answer

- Graft a language model onto the sequential classification scheme.
  - Instead of having the classifier emit one label as an answer for each object, 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 a current popular way of doing this.
    - Although people do the same thing in an ad hoc way with other classifiers
- MEMMs combine two techniques
    - Logistic regression-based classifiers for the individual labeling
    - Markov models for the sequence model.

Models

- HMMs and graphical models are often referred to as generative models since they' re based on using Bayes…
  - So to get P(c|x) we use P(x|c)P(c)
- Alternatively we could use what are called discriminative models; models that get P(c|x) directly without the Bayesian inversion

MaxEnt

- Multinomial logistic regression
- Along with SVMs, MaxEnt is the go-to technique used in NLP these days when a classifier is required.
  - Provides a probability distribution over the classes of interest
  - Admits a wide variety of features
  - Permits the hand-creation of complex features
  - Training time isn’t bad

MaxEnt

$$p(c|x) = \frac{1}{Z}\exp\sum_i w_i f_i$$

MaxEnt

$$p(c|x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} f_i\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^{N} w_{c'i} f_i\right)}$$

Hard Classification

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|x)$$

- If we really want an answer…
- But typically we want a distribution over the answers.

MaxEnt Features

- They' re a little different from the typical supervised ML approach
  - Limited to binary values
    - Think of a feature as being on or off rather than as a feature with a value
  - Feature values are relative to an object/class pair rather than being a function of the object alone.
  - Typically have lots and lots of features (100,000s of features isn' t unusual.)

Features

$$f_3(c,x) = \begin{cases} 1 & \text{if suffix}(word_i) = \text{“ing”} \;\&\; c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if is\_lower\_case}(word_i) \;\&\; c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

Features

$$f_3(c,x) = \begin{cases} 1 & \text{if } \text{suffix}(word_i) = \text{“ing”} \;\; \& \;\; c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c,x) = \begin{cases} 1 & \text{if } \text{is\_lower\_case}(word_i) \;\; \& \;\; c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

- Key point. You can’t squeeze features like these into an HMM.

Mega Features

$$f_{125}(c,x) = \begin{cases} 1 & \text{if } word_{i-1} = \texttt{<s>} \ \& \ \text{isupperfirst}(word_i) \ \& \ c = \text{NNP} \\ 0 & \text{otherwise} \end{cases}$$

- These have to be hand-crafted.
- With the right kind of kernel they can be exploited implicitly with SVMs. At the cost of a increase in training time.

Back to Sequences

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T P(W|T)P(T) \\ &= \operatorname*{argmax}_T \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})\end{aligned}$$

HMMs

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T \prod_i P(tag_i|word_i, tag_{i-1})\end{aligned}$$

- MEMMs

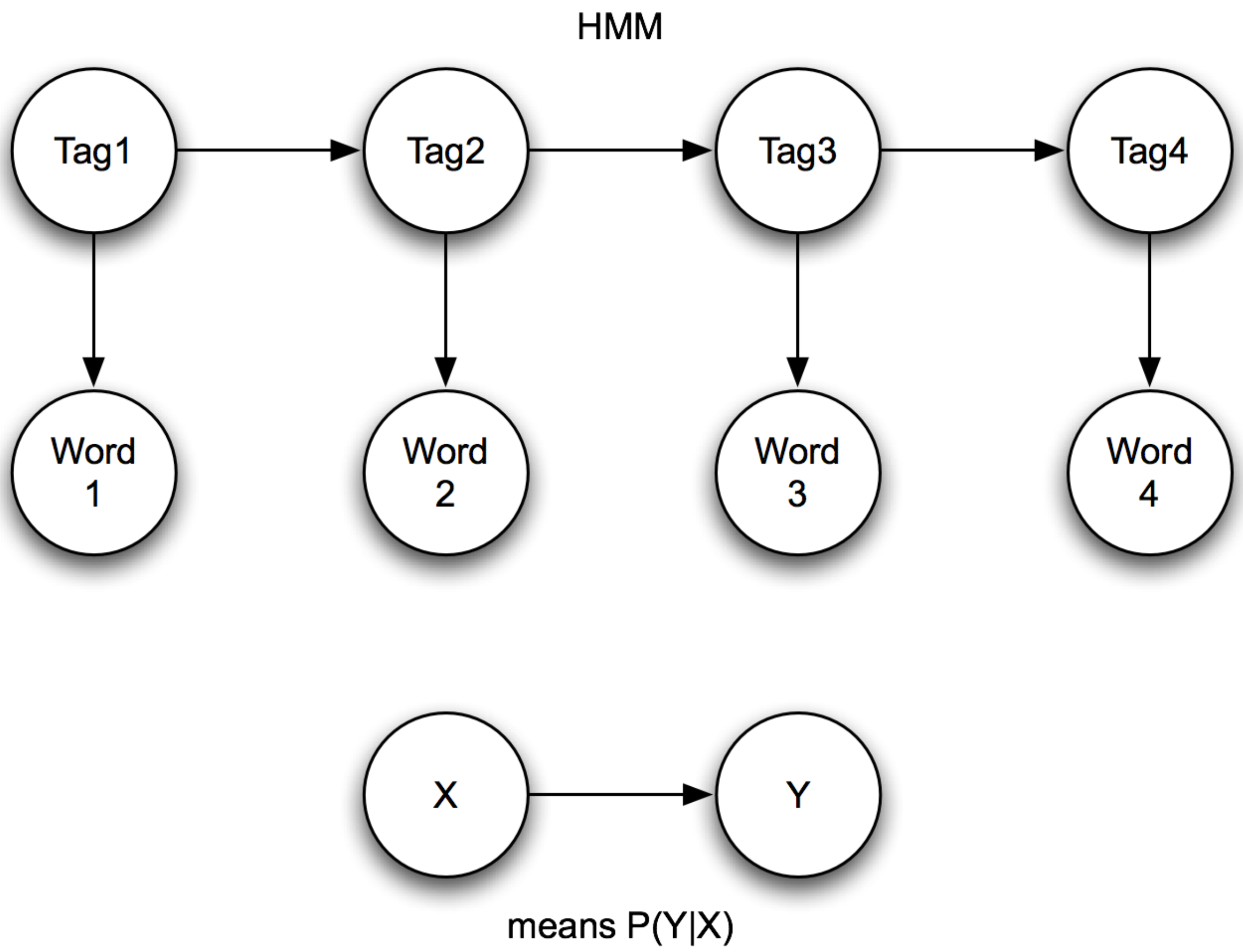
And whatever other features you choose to use!

Back to Viterbi

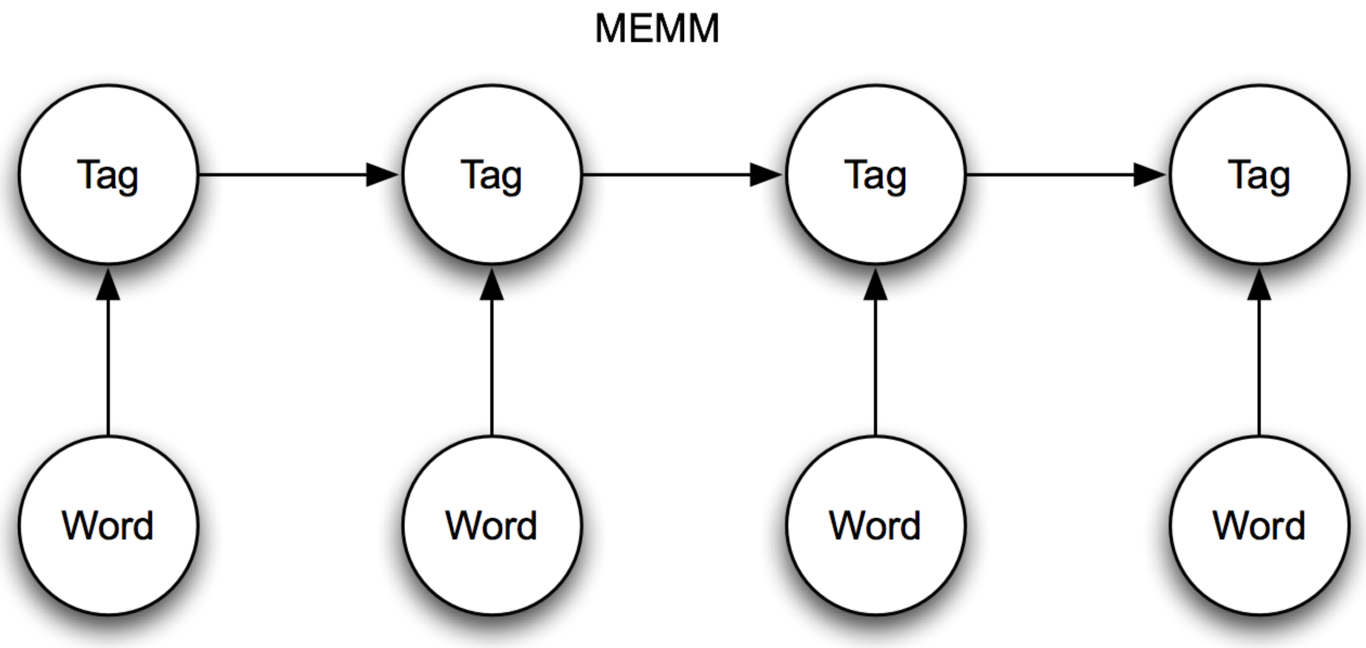
$$v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i)\, P(s_j | s_i, o_t); \quad 1 < j < N, 1 < t < T$$

- The value for a cell is found by examining all the cells in the previous column and multiplying by the posterior for the current column (which incorporates the transition as a factor, along with any other features you like).

HMMs vs. MEMMs

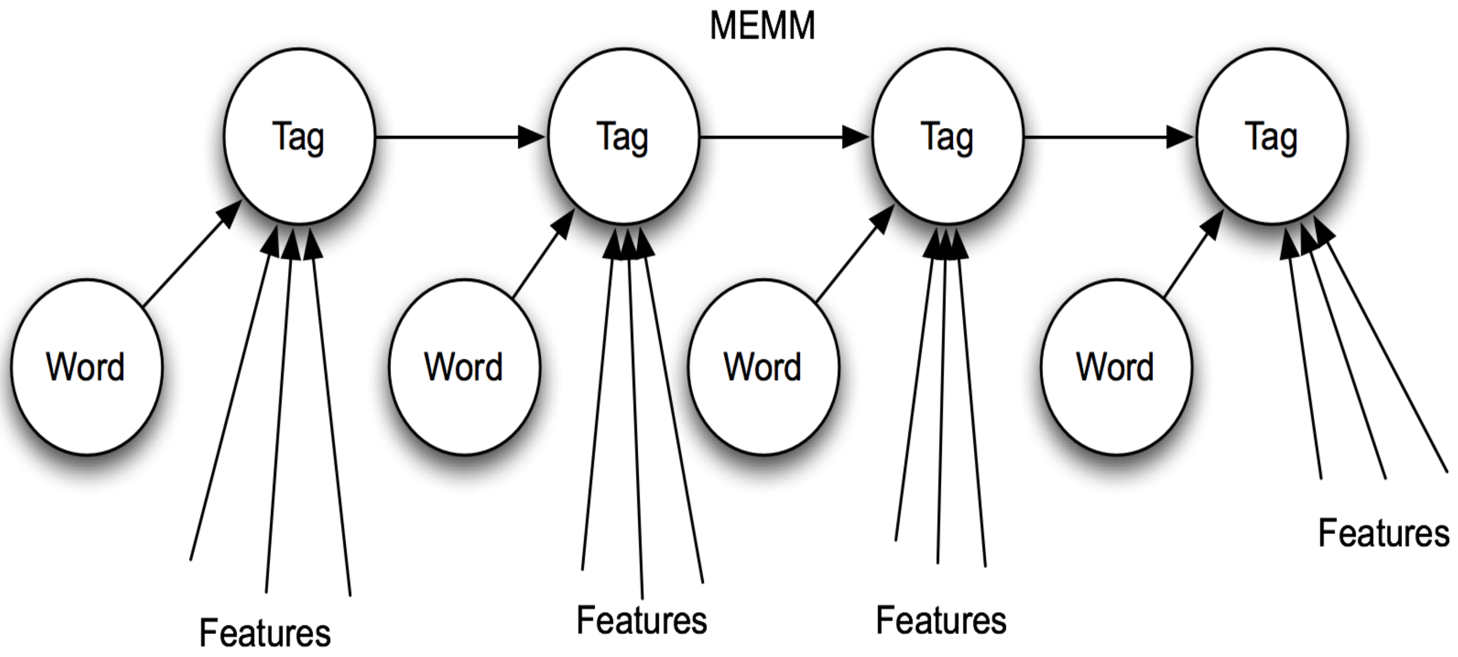


HMMs vs. MEMMs



$$P(T|W) = \prod P(t_i|t_{i-1}, w_i)$$

HMMs vs. MEMMs



$$P(T|W) = \prod P(t_i|t_{i-1}, w_i, f_i)$$

Notes…

- Viewing “structure producing” tasks such as parsing as a sequence of tagging tasks can open up a lot of possibilities
  - IOB style tagging has been applied to a gazzilion tasks
- Generalizing a bit opens up even more possibilities… Abstract the notion of choosing a sequence of tags with making a sequence of decisions