

CS276B

Text Information Retrieval, Mining, and Exploitation

Lecture 7

Information Extraction II

Feb 4, 2003

(includes slides borrowed from David Blei, Andrew McCallum, Nick Kushmerick, BBN, and Ray Mooney)

1

Information Extraction

References

Leslie Pack Kaelbling, Michael L. Littman
and Andrew W. Moore. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, pages 237-285, May 1996.

Headers

Journal of Artificial Intelligence Research 4 (1996) 237-285
Submitted 4/96; published 5/96

Reinforcement Learning: A Survey

Leslie P. Kaelbling LPK@CS.BROWN.EDU
Michael L. Littman MLITTMAN@CS.BROWN.EDU
Computer Science Department, Box 1810, Brown University
Providence, RI 02912-1810 USA

Andrew W. Moore AWMOORE@CMU.EDU
Smith Hall 822, Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213 USA

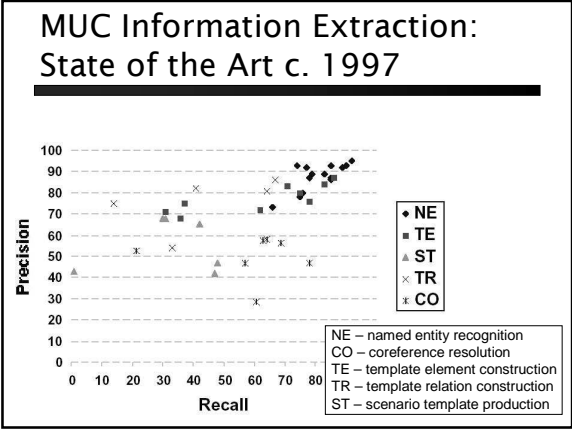
Abstract
This paper surveys the field of reinforcement learning from a computer science perspective. It is written to be accessible to researchers familiar with machine learning. Both the historical basis of the field and a broad selection of current work are summarized. Reinforcement learning is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment. The work described here has a number of applications, but it often concentrates on the details and to the use of the word "reinforcement." The paper discusses central issues of reinforcement learning, including finding off-optimal and sub-optimal, understanding the foundations of the field via Markov decision theory, learning from delayed reinforcement, constructing empirical models to accelerate learning, making use of generalization and knowledge, and coping with hidden state. It concludes with a survey of some implemented systems and an assessment of the practical utility of current methods for reinforcement learning.

1 Introduction

Evaluating IE Accuracy [A common way]

- Always evaluate performance on independent, manually-annotated test data not used during system development!
- Measure for each test document:
 - Total number of correct extractions in the solution template: N
 - Total number of slot/value pairs extracted by the system: E
 - Number of extracted slot/value pairs that are correct (i.e. in the solution template): C
- Compute average value of metrics adapted from IR:
 - Recall = C/N
 - Precision = C/E ← Note subtle difference
 - F-Measure = Harmonic mean of recall and precision

Variants: partial match, etc.



Good Basic IE References

- Douglas E. Appelt and David Israel. 1999. Introduction to Information Extraction Technology. IJCAI 1999 Tutorial. <http://www.ai.sri.com/~appelt/ie-tutorial/>.
- Kushmerick, Weld, Doorenbos: **Wrapper Induction for Information Extraction**, IJCAI 1997. <http://www.cs.ucd.ie/staff/nick/>.
- Stephen Soderland: Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning* 34(1-3): 233-272 (1999)

5

Summary and prelude

- We looked at the "fragment extraction" task. Future?
 - Top-down semantic constraints (as well as syntax)?
 - Unified framework for extraction from regular & natural text? (BWI is one tiny step; Webfoot [Soderland, 97] is another.)
- Beyond fragment extraction:
 - Anaphora resolution, discourse processing, ...
 - Fragment extraction is good enough for many Web information services!
- Applications: What exactly is IE good for?
 - Is there a use for today's "60%" results?
 - Palmtop devices? - IE is valuable if screen is small
- Today
 - Learning methods for information extraction

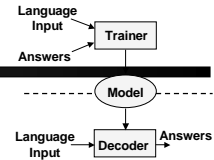
6

Learning for IE

- Writing accurate patterns for each slot for each domain (e.g. each web site) requires laborious software engineering.
- Alternative is to use machine learning:
 - Build a training set of documents paired with human-produced filled extraction templates.
 - Learn extraction patterns for each slot using an appropriate machine learning algorithm.

7

Automatic Pattern-Learning Systems



- Pros:
 - Portable across domains
 - Tend to have broad coverage
 - Robust in the face of degraded input.
 - Automatically finds appropriate statistical patterns
 - System knowledge not needed by those who supply the domain knowledge.
- Cons:
 - Annotated training data, and lots of it, is needed.
 - Isn't necessarily better or cheaper than hand-built sol'n
- Examples: Riloff et al., AutoSlog (UMass); Soderland WHISK (UMass); Mooney et al. Rapier (UTexas):
 - learn lexico-syntactic patterns from templates

8

Rapier [Califf & Mooney, AAAI-99]

- Rapier learns three regex-style patterns for each slot:
 - ▲ Pre-filler pattern
 - ▲ Filler pattern
 - ▲ Post-filler pattern
- One of several recent trainable IE systems that incorporate linguistic constraints. (See also: SIFT [Miller et al, MUC-7]; SRV [Freitag, AAAI-98]; Whisk [Soderland, MLJ-99].)

“...paid \$11M for the company...”
 “...sold to the bank for an undisclosed amount...”
 “...paid Honeywell an undisclosed price...”

Pre-filler:	Filler:	Post-filler:
1) tag: {nn, nnp}	1) word: undisclosed	1) sem: price
2) list: length 2	tag: jj	

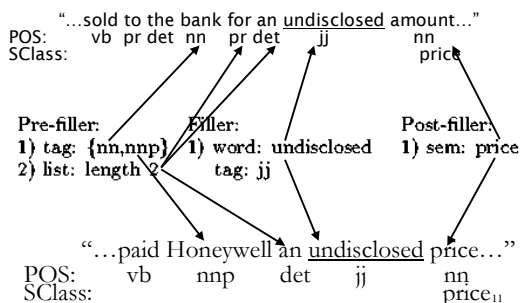
RAPIER rules for extracting “transaction price” 9

Part-of-speech tags & Semantic classes

- Part of speech: syntactic role of a specific word
 - noun (nn), proper noun (nnp), adjective (jj), adverb (rb), determiner (dt), verb (vb), “.” (“.”), ...
 - NLP: Well-known algorithms for automatically assigning POS tags to English, French, Japanese, ... (>95% accuracy)
- Semantic Classes: Synonyms or other related words
 - “Price” class: price, cost, amount, ...
 - “Month” class: January, February, March, ..., December
 - “US State” class: Alaska, Alabama, ..., Washington, Wyoming
 - WordNet: large on-line thesaurus containing (among other things) semantic classes

10

Rapier rule matching example



Rapier Rules: Details

- Rapier rule :=
 - pre-filler pattern
 - filler pattern
 - post-filler pattern
- pattern := subpattern +
- subpattern := constraint +
- constraint :=
 - Word - exact word that must be present
 - Tag - matched word must have given POS tag
 - Class - semantic class of matched word
 - Can specify disjunction with “{...}”
 - List length N - between 0 and N words satisfying other constraints

Pre-filler:	Filler:	Post-filler:
1) tag: {nn, nnp}	1) word: undisclosed	1) sem: price
2) list: length 2	tag: jj	

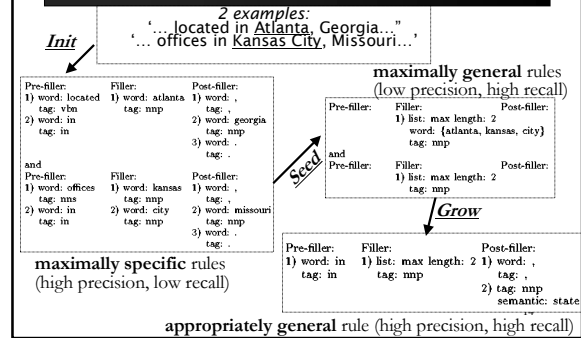
12

Rapier's Learning Algorithm

- Input:** set of training examples (list of documents annotated with "extract this substring")
- Output:** set of rules
- Init:** Rules = a rule that exactly matches each training example
- Repeat several times:
 - Seed:** Select M examples randomly and generate the K most-accurate maximally-general filler-only rules (prefiller = postfiller = "true").
 - Grow:** Repeat For N = 1, 2, 3, ...
 Try to improve K best rules by adding N context words of prefiller or postfiller context
 - Keep:** Rules = Rules \cup the best of the K rules - subsumed rules

13

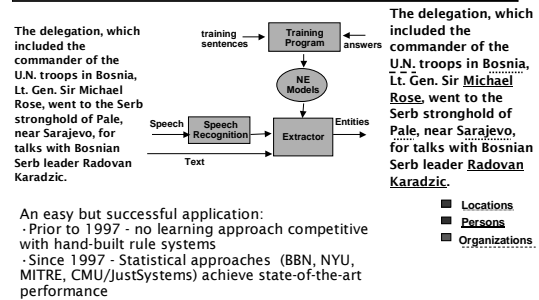
Learning example (one iteration)



Statistical generative models

- Previous discussion examined systems that use explicit extraction patterns/rules
- Hidden Markov Models are a powerful alternative based on statistical token sequence generation models rather than explicit extraction patterns.
- Pros:**
 - Well-understood underlying statistical model makes it easy to use wide range of tools from statistical decision theory
 - Portable, broad coverage, robust, good recall
- Cons:**
 - Range of features and patterns usable may be limited
 - Not necessarily as good for complex multi-slot patterns

Name Extraction via HMMs

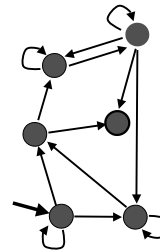


Applying HMMs to IE

- Document** \Rightarrow generated by a stochastic process modelled by an HMM
- Token** \Rightarrow word
- State** \Rightarrow "reason/explanation" for a given token
 - 'Background' state emits tokens like 'the', 'said', ...
 - 'Money' state emits tokens like 'million', 'euro', ...
 - 'Organization' state emits tokens like 'university', 'company', ...
- Extraction:** via the Viterbi algorithm, a dynamic programming technique for efficiently computing the most likely sequence of states that generated a document.

17

HMM formalism



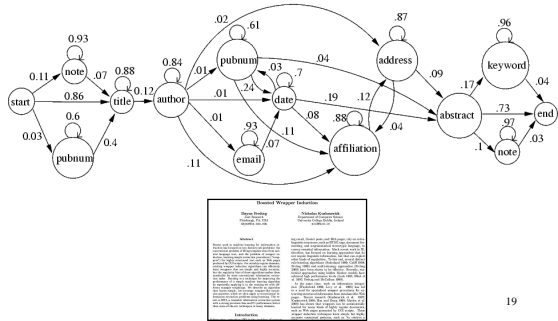
HMM = probabilistic FSA

HMM = states s_1, s_2, \dots
 (special start state s_1
 special end state s_n)
 token alphabet a_1, a_2, \dots
 state transition probs $P(s_j | s_i)$
 token emission probs $P(a_i | s_j)$

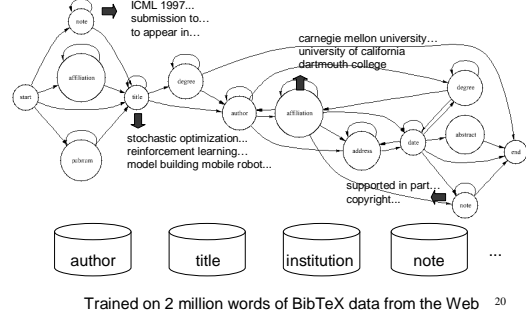
Widely used in many language processing tasks, e.g., speech recognition [Lee, 1989], POS tagging [Kupiec, 1992], topic detection [Yamron *et al.*, 1998]

18

HMM for research papers: transitions [Seymore et al., 99]



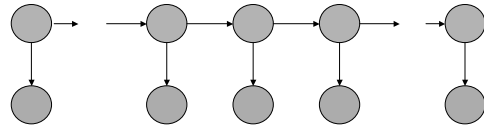
HMM for research papers: emissions [Seymore et al., 99]



Learning HMMs

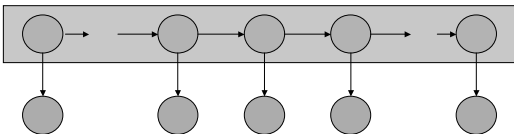
- **Good news:** If training data tokens are tagged with their generating states, then simple frequency ratios are a maximum-likelihood estimate of transition/emission probabilities. Easy. (Use smoothing to avoid zero probs for emissions/transitions absent in the training data.)
 - **Great news:** Baum-Welch algorithm trains an HMM using partially labeled or unlabelled training data.
 - **Bad news:** How many states should the HMM contain? How are transitions constrained?
 - Only semi-good answers to finding answer automatically
 - Insufficiently expressive \Rightarrow Unable to model important distinctions (long distance correlations, other features)
 - Overly expressive \Rightarrow sparse training data, overfitting
- 21

What is an HMM?



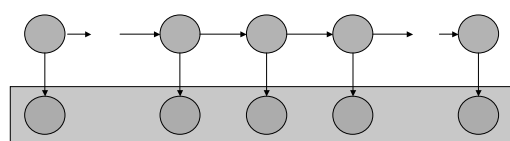
- Graphical Model Representation: Variables by time
 - Circles indicate states
 - Arrows indicate probabilistic dependencies between states
- 22

What is an HMM?



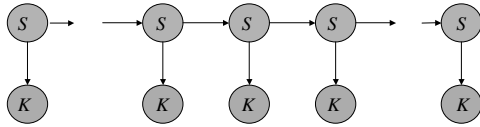
- Green circles are **hidden states**
 - Dependent only on the previous state: Markov process
 - "The past is independent of the future given the present."
- 23

What is an HMM?



- Purple nodes are **observed states**
 - Dependent only on their corresponding hidden state
- 24

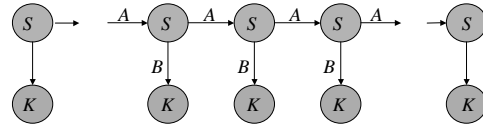
HMM Formalism



- $\{S, K, \Pi, A, B\}$
- $S: \{s_1 \dots s_N\}$ are the values for the hidden states
- $K: \{k_1 \dots k_M\}$ are the values for the observations

25

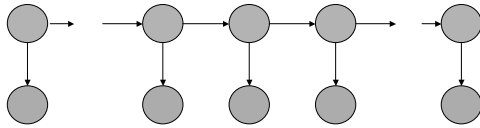
HMM Formalism



- $\{S, K, \Pi, A, B\}$
- $\Pi = \{\pi_i\}$ are the initial state probabilities
- $A = \{a_{ij}\}$ are the state transition probabilities
- $B = \{b_{ik}\}$ are the observation state probabilities

26

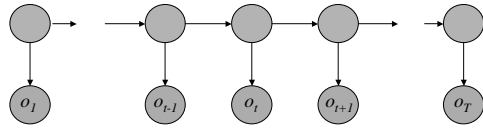
Inference for an HMM



- Compute the probability of a given observation sequence
- Given an observation sequence, compute the most likely hidden state sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

27

Sequence Probability



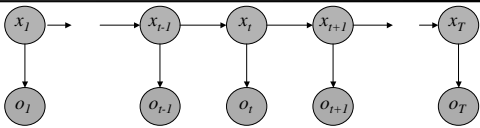
Given an observation sequence and a model, compute the probability of the observation sequence

$$O = (o_1, \dots, o_T), \mu = (A, B, \Pi)$$

Compute $P(O | \mu)$

28

Sequence probability



$$P(O | X, \mu) = b_{x_1 o_1} b_{x_2 o_2} \dots b_{x_T o_T}$$

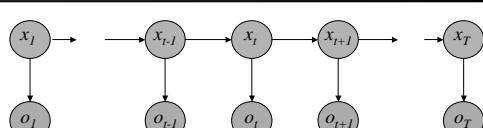
$$P(X | \mu) = \pi_{x_1} a_{x_1 x_2} a_{x_2 x_3} \dots a_{x_{T-1} x_T}$$

$$P(O, X | \mu) = P(O | X, \mu) P(X | \mu)$$

$$P(O | \mu) = \sum_X P(O | X, \mu) P(X | \mu)$$

29

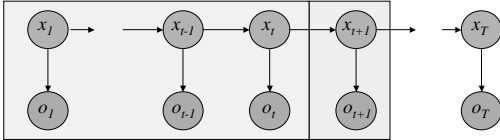
Sequence probability



$$P(O | \mu) = \sum_{\{x_1 \dots x_T\}} \pi_{x_1} b_{x_1 o_1} \prod_{t=1}^{T-1} a_{x_t x_{t+1}} b_{x_{t+1} o_{t+1}}$$

30

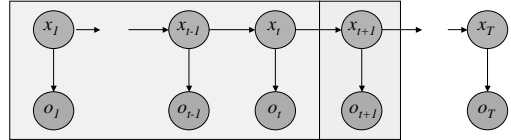
Sequence probability



- Special structure gives us an efficient solution using *dynamic programming*.
- **Intuition:** Probability of the first t observations is the same for all possible $t + 1$ length state sequences.
- **Define:** $\alpha_i(t) = P(o_1 \dots o_t, x_t = i | \mu)$

31

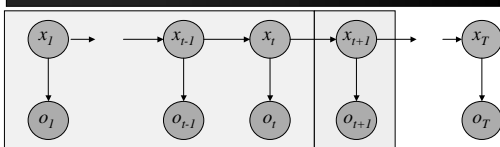
Forward Procedure



$$\begin{aligned} \alpha_j(t+1) &= P(o_1 \dots o_{t+1}, x_{t+1} = j) \\ &= P(o_1 \dots o_{t+1} | x_{t+1} = j) P(x_{t+1} = j) \\ &= P(o_1 \dots o_t | x_{t+1} = j) P(o_{t+1} | x_{t+1} = j) P(x_{t+1} = j) \\ &= P(o_1 \dots o_t, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j) \end{aligned}$$

32

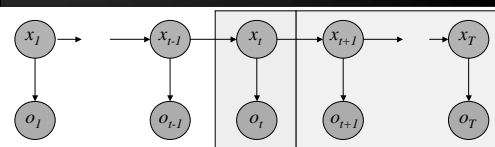
Forward Procedure



$$\begin{aligned} &= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i, x_{t+1} = j) P(o_{t+1} | x_{t+1} = j) \\ &= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_{t+1} = j | x_t = i) P(x_t = i) P(o_{t+1} | x_{t+1} = j) \\ &= \sum_{i=1 \dots N} P(o_1 \dots o_t, x_t = i) P(x_{t+1} = j | x_t = i) P(o_{t+1} | x_{t+1} = j) \\ &= \sum_{i=1 \dots N} \alpha_i(t) a_{ij} b_{j o_{t+1}} \end{aligned}$$

33

Backward Procedure

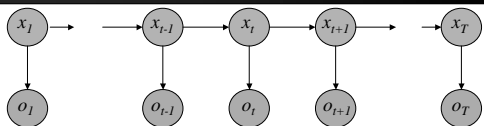


$$\begin{aligned} \beta_i(T+1) &= 1 \\ \beta_i(t) &= P(o_t \dots o_T | x_t = i) \\ \beta_i(t) &= \sum_{j=1 \dots N} a_{ij} b_{j o_{t+1}} \beta_j(t+1) \end{aligned}$$

Probability of the rest of the states given the first state

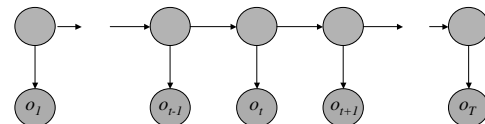
34

Sequence probability



$P(O \mu) = \sum_{i=1}^N \alpha_i(T)$	Forward Procedure
$P(O \mu) = \sum_{i=1}^N \pi_i \beta_i(1)$	Backward Procedure
$P(O \mu) = \sum_{i=1}^N \alpha_i(t) \beta_i(t)$	Combination

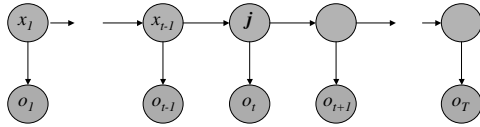
Best State Sequence



- Find the state sequence that best explains the observations
- **Viterbi algorithm** (1967)
- $\arg \max_x P(X | O)$

36

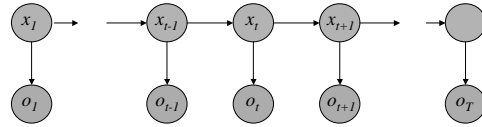
Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \dots x_{t-1}} P(x_1 \dots x_{t-1}, o_1 \dots o_{t-1}, x_t = j, o_t)$$

The state sequence which maximizes the probability of seeing the observations to time t-1, landing in state j, and seeing the observation at time t

Viterbi Algorithm



$$\delta_j(t) = \max_{x_1 \dots x_{t-1}} P(x_1 \dots x_{t-1}, o_1 \dots o_{t-1}, x_t = j, o_t)$$

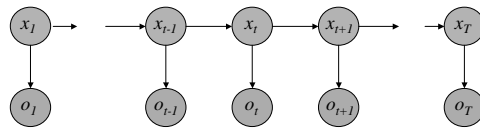
$$\delta_j(t+1) = \max_i \delta_i(t) a_{ij} b_{j o_{t+1}}$$

$$\psi_j(t+1) = \arg \max_i \delta_i(t) a_{ij} b_{j o_{t+1}}$$

Recursive Computation

38

Viterbi Algorithm



$$\hat{X}_T = \arg \max_i \delta_i(T)$$

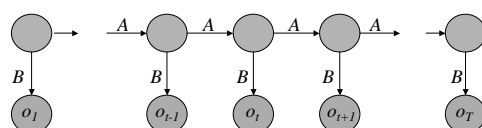
$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

$$P(\hat{X}) = \arg \max_i \delta_i(T)$$

Compute the most likely state sequence by working backwards

39

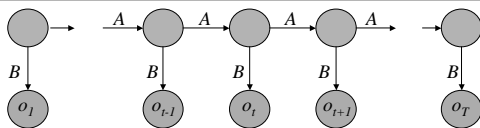
Learning = Parameter Estimation



- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method, so:
- Given a model and observation sequence, update the model parameters to better fit the observations.

40

Parameter Estimation: Baum-Welch or Forward-Backward



$$p_t(i, j) = \frac{\alpha_i(t) a_{ij} b_{j o_{t+1}} \beta_j(t+1)}{\sum_{m=1 \dots N} \alpha_m(t) \beta_m(t)}$$

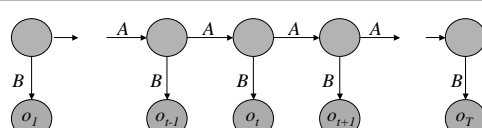
Probability of traversing an arc

$$\gamma_i(t) = \sum_{j=1 \dots N} p_t(i, j)$$

Probability of being in state i

41

Parameter Estimation: Baum-Welch or Forward-Backward



$$\hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{t=1}^T \gamma_i(t)}$$

$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{b}_{ik} = \frac{\sum_{\{t, o_t=k\}} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$$

Now we can compute the new estimates of the model parameters.

Is it that easy?

- As often with text, the biggest problem is the *sparseness* of observations (words)
- Need to use many techniques to do it well
 - Smoothing* (as in NB) to give suitable nonzero probability to unseens
 - Featural decomposition* (capitalized?, number?, etc.) gives a better estimate
 - Shrinkage* allows pooling of estimates over multiple states of same type (e.g., prefix states)
- Well designed (or learned) HMM topology

43

HMM example

"Seminar announcements" task

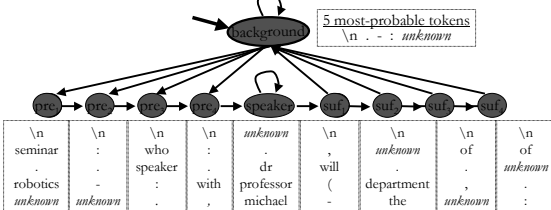
```
<0.15.4.95.15.11.55.rudibear+@CMU.EDU.0>
Type: cmu.andrew.assoc.UEA
Topic: Re: entrepreneurship speaker
Dates: 17-Apr-95
Time: 7:00 PM
PostedBy: Colin S Osburn on 15-Apr-95 at 15:11 from CMU.EDU
Abstract:

hello again
to reiterate
there will be a speaker on the law and startup business
this monday evening the 17th
it will be at 7pm in room 261 of GSIA in the new building, ie
upstairs.
please attend if you have any interest in starting your own
business or
are even curious.
Colin
```

44

HMM example, continued

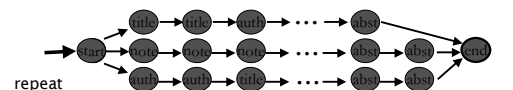
Fixed topology that captures limited context: 4 "prefix" states before and 4 "suffix" after target state.



[Freitag, 99]
45

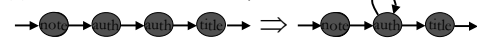
Learning HMM structure [Seymore et al, 1999]

start with maximally-specific HMM (one state per observed word):

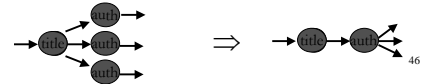


repeat

- merge adjacent identical states
- eliminate redundant fan-out/in

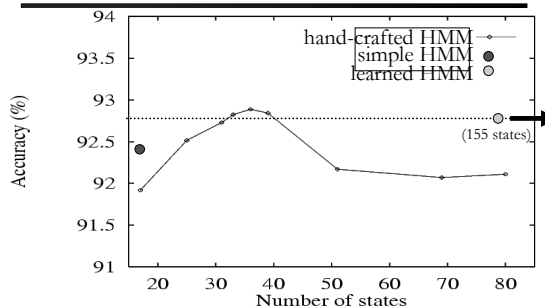


until obtain good tradeoff between HMM accuracy and complexity



46

Evaluation (% tokens tagged correctly)



47

References

- Mary Elaine Califf and Raymond J. Mooney: Relational Learning of Pattern-Match Rules for Information Extraction. In *AAAI 1999*: 328-334.
- Leek, T. R. 1997, Information Extraction using Hidden Markov Models, Master's thesis, UCSD
- Bikel, D. M.; Miller, S; Schwartz, R.; and Weischedel, R. 1997, Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, 194-201. [Also in *MLJ 1999*]
- Kristie Seymore, Andrew McCallum, Ronald Rosenfeld, 1999, Learning Hidden Markov Model Structure for Information Extraction, In *Proceedings of the AAAI-99 Workshop on ML for IE*.
- Dayne Freitag and Andrew McCallum, 2000, [Information Extraction with HMM Structures Learned by Stochastic Optimization](#). *AAAI-2000*.

48