Statistical Machine Translation

LECTURE – 9

HMM AND MEM

22APRIL 2010

Reference:

1) Jurafsky and Martin: Speech and Language Processing, Pearson.

2) Monojit Choudhury, Rahul Saraf, Vijit Jain, Sudeshna Sarkar, Anupam Basu. 2007. *Investigation and Modeling of the Structure of Texting Language, volume 10. International Journal on* Document Analysis and Recognition.

3) **Niladri Chatterjee** and Rohit Mishra. Word-Sense Disambiguation using Maximum Entropy Model. Proc. ICM2C09, IEEE Explorer, ISBN 9789380 043579, pp 154 – 158, 2009.

Brief Outline

-Hidden Markov Model

-- Likelihood Computation

-- Decoding – Viterbi Algorithm

-- Training - Forward & Backward Algorithm

-- Application in Texting to Std. Lang. Translation

-Maximum Entropy Modeling

-- Application in WSD

Introduction

Both HMM and MEM are important classes of statistical models for Machine Learning used in Speech and Language Processing.

A.A.Markov (1913) – can frequency counts be used to compute the probability that next letter is a vowel.? This was related to Pushkin's verse: Eugene Onegin. They have the rhyming: "aBaBccDDeFFeGG" known as Iambic Tetrameter.

-Both are sequence classifier.

-Improved version of FSA (probabilistic).

Hidden Markov Model

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unobserved state.

A stochastic process has the Markov property if the conditional probability distribution of future states of the process depend only upon the present state; that is, given the present, the future does not depend on the past. A process with this property is called Markov process.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.

Wikipedia

Hidden Markov Model

In a *hidden* Markov model, the state is not directly visible, but output dependent on the state is visible.

Each state here has a probability distribution over the possible output tokens.

Therefore the sequence of tokens generated by a HMM gives some information about the sequence of states.

Note that the adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; even if the model parameters are known exactly, the model is still 'hidden'.

Wikipedia

HMM

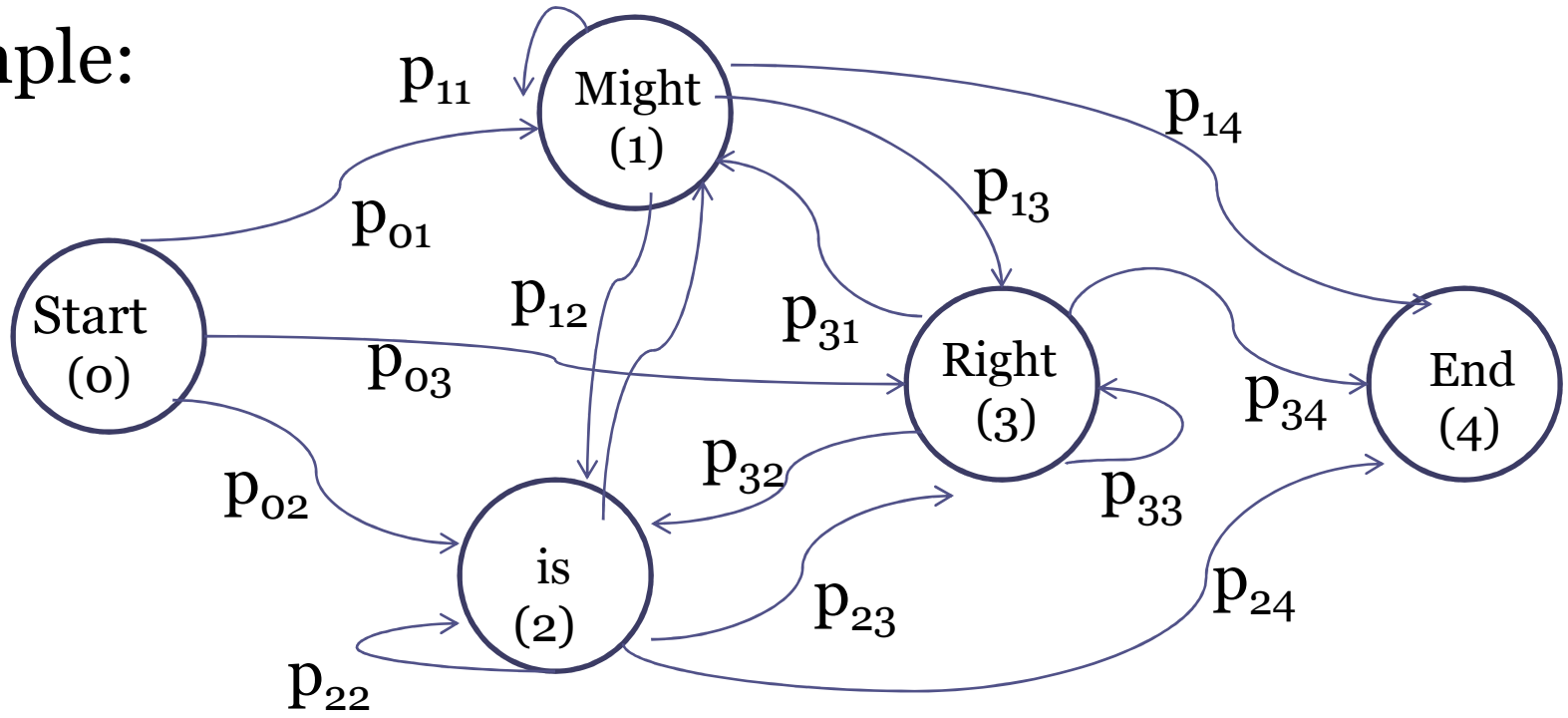
HMM is based on Markov Chain.

A Markov Chain is a special case of weighted Automaton in which the input Sequence uniquely Determines the set of states the automaton will go through.

Note: A weighted Automaton is a FSA in which each arc is associated with a probability.

Markov Chain

Example:



Markov chain for bigram language model

Markov Chain

Markov Assumption:

1st Order $p(q_i \mid q_1 .. q_{i-1}) = p(q_i \mid q_{i-1})$

$$\sum_j p_{ij} = 1 \; \forall i$$

Similarly higher orders ..

Markov Chain

Alternative way:

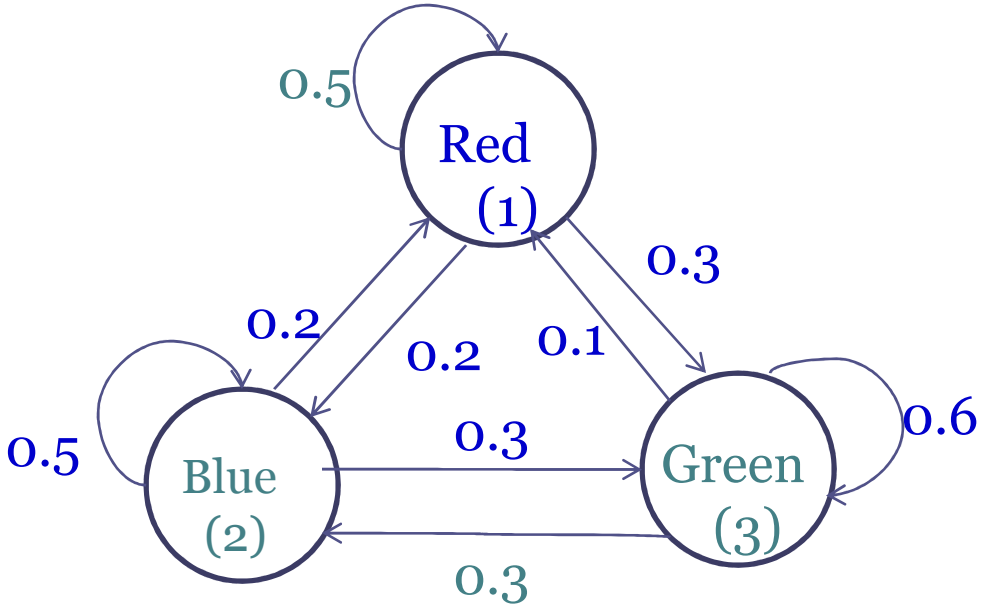
There is no start & end state.

An initial probability distribution π is given:

i.e. π_i = prob. the process starts at state i.

$$\sum_i \pi_i = 1$$

E.g.



If π = 0.5 0.3 0.2

What is the prob. of (say)
Red Red Red

Hidden Markov Model

Hidden Markov Model

Markov chain is useful to compute the probability of a sequence of events that we observe.

How to compute for events not directly observable?

Typically used in POS tagging/ speech recognition
We will show its use in understanding Texting Languages.

A hidden Markov model allows us to compute thru observed as well as hidden event.

Hidden Markov Model

An HMM consists of the following components:

$Q = (q_1, \ldots q_n\}$ n states

$A = ((a_{ij}))$ transition matrix . $\sum_j \pi_{ij} = 1 \quad \forall$ i

$O = \{o_1 , .. o_T\}$ a set of observations

$B = b_i(o_t)$ emission probability that observation o_t emanated from state i.

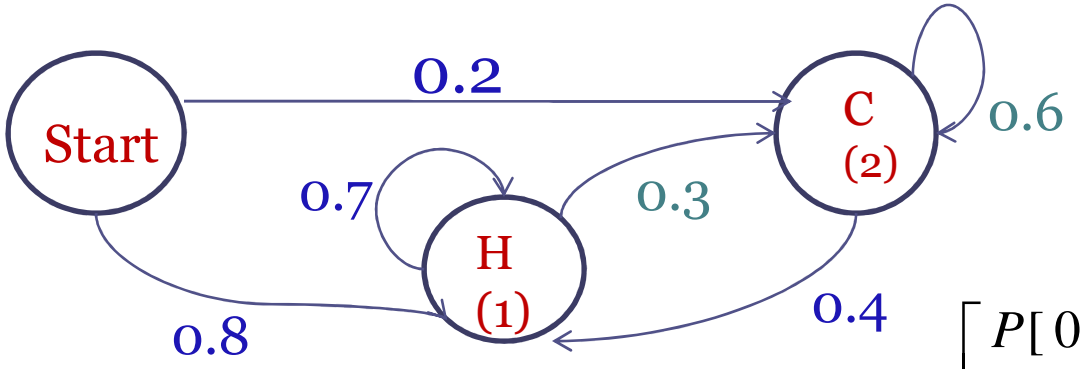
Q_o = start state with emission probabilities $a_{o1} , .. \ a_{on}$

Q_F = end state with emission probabilities $a_{1F} , .. \ a_{nF}$ (Optional)

Hidden Markov Model

Example: Suppose we have no record of temperature of a city. But the diary of a boy records how many ice-creams he took everyday day. Can we guess for each day it was hot or cold?

Use an HMM with 2 hidden states - H and C.



$$O = \{0, 1, 2, 3\}$$

B1

$$\begin{bmatrix} P[0|H]=0.1 \\ P[1|H]=0.4 \\ P[2|H]=0.4 \\ P[3|H]=0.1 \end{bmatrix}$$

B2

$$\begin{bmatrix} P[0|C]=0.5 \\ P[1|C]=0.3 \\ P[2|C]=0.2 \\ P[3|C]=0.0 \end{bmatrix}$$

Hidden Markov Model

First order HMM -- two simplifying assumptions:

1. Markov 1st order assumption:

$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

2. Output independence:

$$P(o_i \mid q_1 \ldots q_i , \; o_1 \ldots o_i) = P(o_i \mid q_i)$$

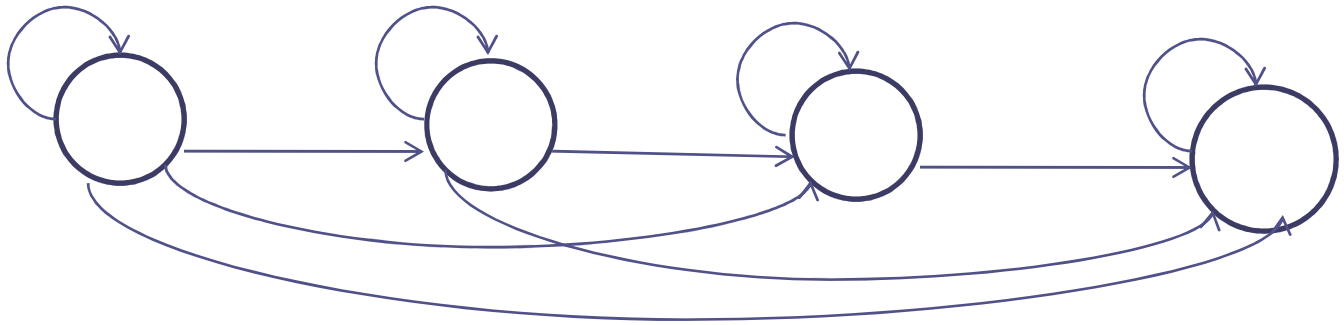
Hidden Markov Model

HMM is said to be fully connected (or ergodic) if there is non-zero probability of transition between any pair of states.

There can be HMMs with some transitions having zero probability e.g Left to Right (also called Bakis HMM).

Example:

Typically used for Monotone signals



Typical Applications

Likelihood:

Given an HMM (A,B) and a sequence of Observation O, determine the likelihood P(O | (A,B)).

Decoding:

Given an HMM (A,B) and a sequence of Observation O, discover the best hidden state sequence Q.

Learning:

Given a sequence of Observation O and set of states Q learn the parameters A and B

Typical Applications

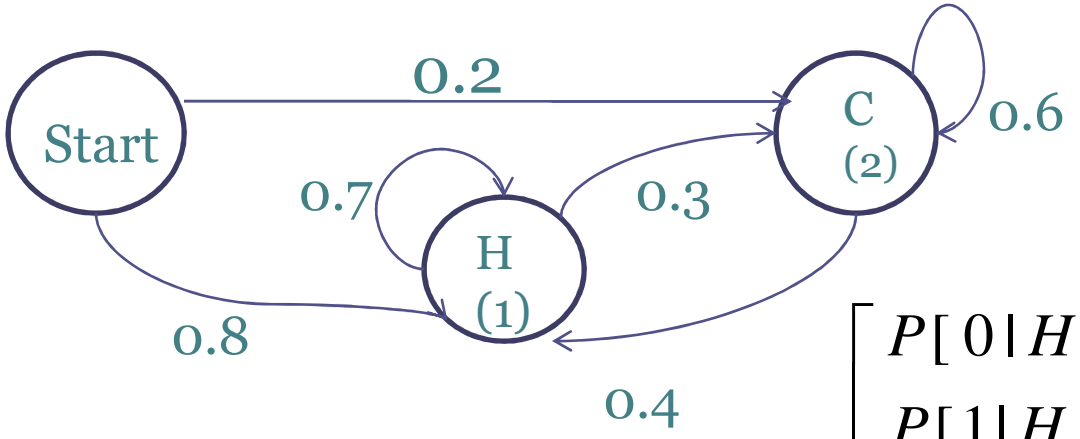
Has been used thoroughly in Speech.

Of late is being used in decoding SMS language – which can also be thought of Language translation.

Likelihood Computation

Consider the ice-cream problem.

We want to compute the probability of 1 1 2.



$$O = \{0, 1, 2, 3\}$$

	B1	B2
	$P[0 \mid H] = 0.1$	$P[0 \mid C] = 0.5$
	$P[1 \mid H] = 0.4$	$P[1 \mid C] = 0.3$
	$P[2 \mid H] = 0.4$	$P[2 \mid C] = 0.2$
	$P[3 \mid H] = 0.1$	$P[3 \mid C] = 0.0$

Likelihood Computation

Note that thee are 8 ways of getting 1 1 2.

HHH, HHC, HCH, HCC, CHH, CHC, CCH, CCC

E.g. $P(1\,1\,2 \mid H\,C\,H) = P(H \mid S) * P(1 \mid H) * P(C \mid H) * P(1 \mid C) * P(H \mid C) * P(2 \mid H)$

$= 0.4 * 0.4 * 0.3 * 0.3 * 0.4 * 0.2$

Note: $$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O \mid Q) P(Q)$$

With N hidden states and T observations - N^T hidden sequences
Which means Exponential complexity

Likelihood Computation

A Dynamic Programming based algorithm give $O(N^2T)$ complexity.

This is called Forward Algorithm:

It computes algorithm trellis $\alpha_t(j)$

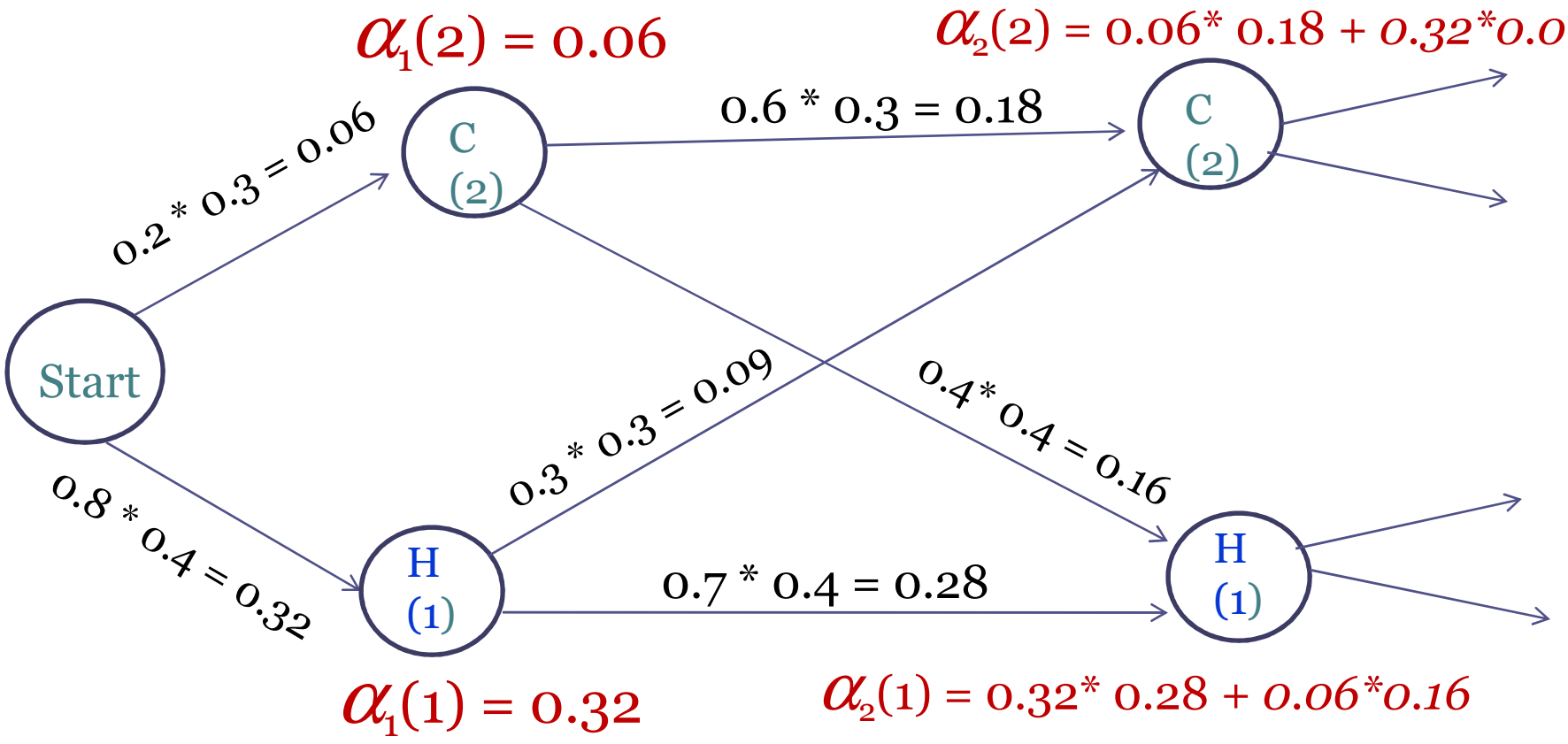
– *probability of being in state j after seeing first t observations.*

i.e. $P(o_1 o_2 \ldots o_t, q_t = j \mid (A, B))$

$$= \sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

Likelihood Computation

Example – ice-cream problem O = 1 1 2



Try out Assignment

1. Write an algorithm for Forward computation for a given HMM.

2. Check that it is $O(N^2T)$

Decoding

Decoding

The task of decoder is to find the most likely sequence of states $q_1\ q_2\ \ldots q_t$ given a sequence of output: $o_1\ o_2\ \ldots\ o_t$

Suppose w.r.t ice-cream problem it is HHH that is most likely once 1 1 2 is observed.

How to find it?

Note: The exponential nature of growth does not allow us to find it for all possible combination of states using Forward algorithm and pick the maximum one!!!

Viterbi Algorithm - the most commonly used.

Very similar to the Forward algorithm.

Decoding

Here we have trellis: $v_t(j)$

$v_t(j)$ = the probability that the HMM in state j at the t^{th} observation after passing through the most possible set of states $q_0, q_1 \ldots q_{t-1}$, i.e.

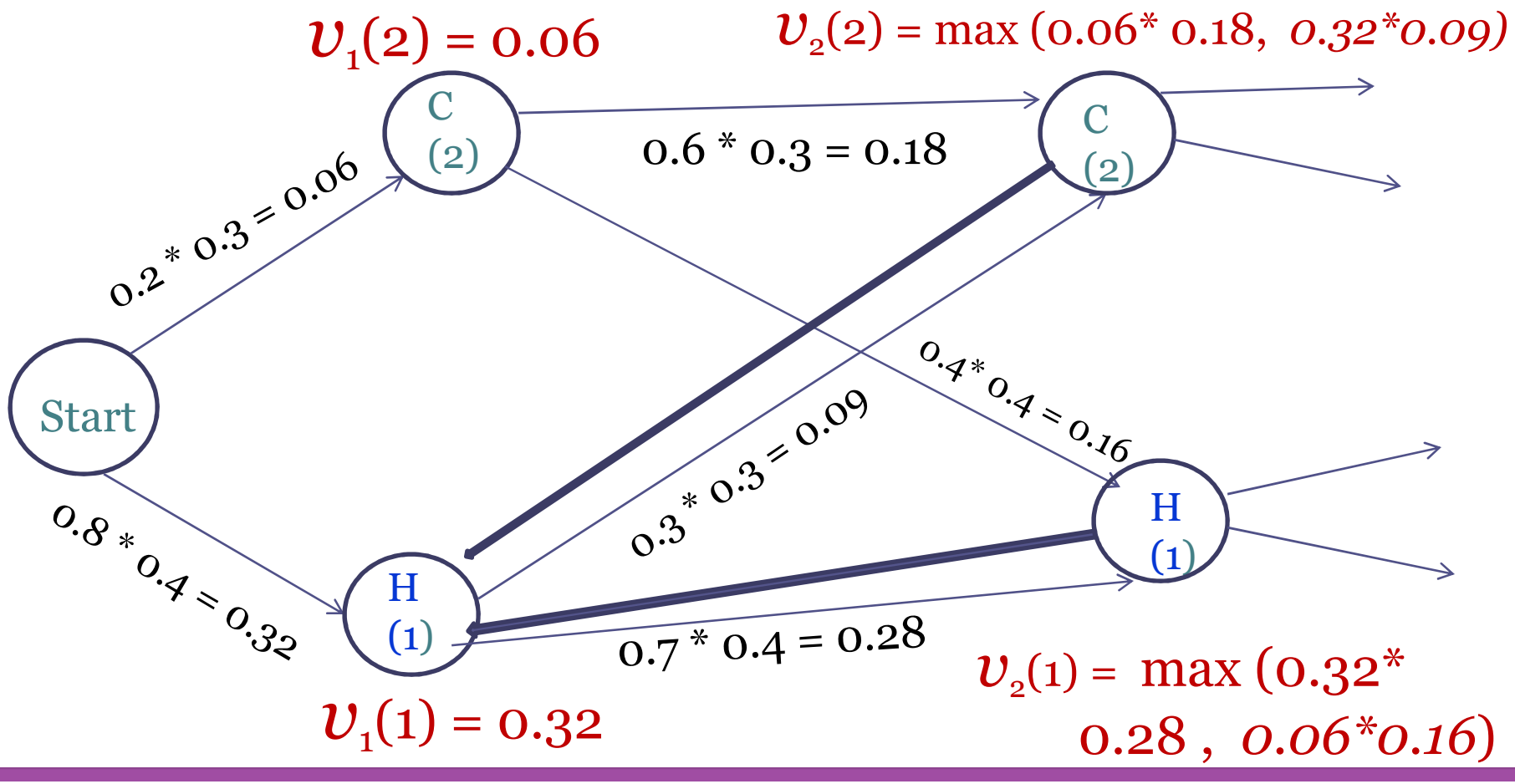
$$v_t(j) = \max_{q0,\, q1 \ldots qt-1} P(q_0, q_1 \ldots q_{t-1}\, o_1\, o_2 \ldots o_t,\, q_t = j \mid (A, B))$$

Thus computation is very similar:

- Sum is replaced by max.
- Needs to maintain a pointer indicating the maximum path.

Viterbi Algorithm

Example – ice-cream problem O = 1 1 2



Viterbi Algorithm

Three steps

1) Initialization:

$$v_1(j) = a_{0j} b_j(o_1), \quad 1 \le j \le N$$

$$bp_1(j) = 0, \quad 1 \le j \le N$$

2)Recursion:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t), \quad 1 \le j \le N$$

$$bp_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t), \quad 1 \le j \le N$$

3) Termination:

$$P^* = v_t(q_F) = \max_{i=1}^{N} v_T(i)\, a_{iF}$$

$$bp_T(q_F) = \operatorname*{argmax}_{i=1}^{N} v_T(i)\, a_{iF}$$

Training an HMM

Training an HMM

The task : to learn the parameters of an HMM – i.e.

Given an observation sequence O and the set of possible states in the HMM to learn the probabilities A and B.

The most popular algorithm:

Forward –Backward Algorithm (Baum-Welch 1972)

-- estimates both A and B.

The task is more difficult for HMM compared to Markov chain – Why?

Training an HMM

For a Markov chain:

- the states are visible
- and there is no emission probabilities.

$\Rightarrow$ no b_{ij} is to be estimated

(output depends on state)

a_{ij} s can be estimated from counts

$$a_{ij} = \frac{count\ (q_i \rightarrow q_j)}{\sum_{k=1}^{N} count\ (q_i \rightarrow q_k)}$$

} Such counting is NOT possible for HMM – as the States are unknown

Training an HMM

Baum-Welch Algorithm solves this as follows:

- Estimate the counts iteratively
- Once a probability is estimated (Forward) it is divided among different paths that contribute.

It is done using the concept of Backward Probability.

$\beta_t(i)$ - the probability of seeing the observation o_{t+1} to o_T given that at time t the process was at state i.

$$\beta_t(i) = P(o_{t+1}\, o_{t+2} \ldots o_T \mid q_t = i)$$

Backward Probability

Initialization: $\beta_T(i) = a_{iF} \quad \forall i = 1,\ldots N$

Recursion: $\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j) \quad 1 \le i \le N,\ T-1 \ge t \ge 1$

Termination: $\beta_0(0) = \sum_{j=1}^{N} a_{0j}\, b_j(o_1)\, \beta_1(j)$

The Backward probability and Forward probability Together help us in estimating the a_{ij}s and b_{jt}s

Training an HMM

Initial estimates of a_{ij} s are done as follows:

$$a_{ij} = \frac{E(no.of\ \ transition\ \ \ from\ \ state\ \ i\ to\ \ state\ \ \ j)}{E(no.of\ \ transition\ \ \ from\ \ state\ \ i)}$$

However, there is no direct way of computing!!

Define $\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \mathcal{M})$

$$\cong \frac{\alpha_t(i)\, a_{ij}\, b_j(o_t)\, \beta_t(j)}{\alpha_T(N)}$$

Then we have $$a_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{j=1}^{N}\sum_{t=1}^{T} \xi_t(i,j)}$$

Training an HMM

In a similar way $b_j(v_k)$ s are estimated

$$b_j(v_k) = \frac{E(no.\ of\ \ times\ in\ \ state\ \ j\ \ and\ \ observe\ symbol\ v_k)}{E(no.\ \ of\ times\ in\ \ state\ \ j)}$$

Then we have

$$b_j(v_k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \quad (s.t. \quad o_t = v_k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Where $\gamma_t(j) = P(q_t = j \mid O, \mathcal{M})$

Training an HMM

Expectation Maximization is then used with the following initialization:

Initialize: A and B (say uniform)

E-step: $\gamma_t(j) = \alpha_t(j)\beta_t(j) / P(O \mid \mathcal{M})$ $\quad \forall$ t, j

$$\xi_t(i, j) = \cong \frac{\alpha_t(i)\, a_{ij}\, b_j(o_t)\, \beta_t(j)}{\alpha_T(N)}$$

$\forall$ i, t, j

Until converge.

Epilogue

Forward-Backward algorithm can be used as an Unsupervised learning model for A and B parameters.

However, initial parameters often play a major Role.

Typically the model is set by hand, and training Set of observations is used for learning.

HMM Application

Translating from texting Language

Introduction

Texting Language (*NetSpeak*;Txtese)

- is a term for the *abbreviations* and *slang* commonlyused due to the necessary brevity of mobile phone text messaging, e-mail and chats.

- words used may often not be found in standard dictionaries

- does not obey or follow Standard English grammar

Examples:

"*2mrw*" stands for tomorrow

"*i <3 u*" stands for 'I love you'

Motivation

- Use of *small keyboards* (like those of mobile phones) motivates the use of compression techniques to reduce message length

- Such techniques help users to *write faster*

- On the other hand, it takes almost *double time to read* such a language as compared to standard language (study by Nenagh Kemp , University of Tasmania)

- A tool for conversion of SMS language to Standard Language can have applications in others fields like *emails, spam filters, blog analysis, search engines* etc.

Common Compression Techniques

- Spelling and Punctuation
  - Phonetic substitutions for letter sequence or words
    (e.g.: *2* – to; *l8r*-later; *v*-we)
  - Deletion of space between words
    (e.g.: *Iamgoing home* – I am going home.)
  - Deletion of punctuation marks
    (e.g. *u looked a different Sarah lunchtime more excited more sure*
    – You looked a different Sarah at lunchtime; more excited, more sure.)

Common Compression Techniques

- Spelling and Punctuation
  - Use of all lower or all upper case
    (e.g. *r u meetg 2day* – Are we meeting today?)
  - Typos (e.g. *lavex* - latex)
  - Deletion of vowels (e.g. *m lvng* – I am leaving)
  - Word truncation (e.g. – *comin* – coming)
  - Repeated consonant deletion (e.g. *2morow* – tomorrow)

Common Compression Techniques

- Grammatical features
  - Deletion of pronouns
    (e.g. *ws gld to fnd d bus pass* –
    I was glad to find the bus pass)
  - Deletion of auxiliaries in verb phrases
    (e.g. *I want go home* – I want to go home)
  - Deletion of prepositions / articles
    (e.g. *cat sleeping sofa* – A cat is sleeping on the sofa)

Common Techniques

- Abbreviations and other features
  - Conventional and unconventional abbreviations
    (e.g. *btw* – by the way;
    ttyl – talk to you later,
    lol – laugh out loudly)
  - Use of dialectal forms and slangs
    (e.g. *aint* – am/is not; *gonna* – going to)
  - Use of emoticons (e.g. use of *smileys* ☺, ☹, :-P)
  - Use of extra letters/words for emphasis
    (e.g. *I am so so so bored* – I am so bored,
    I love u soooooooo muuuuuuch – I love you so much)

Example of Spelling Variation

The variations of the word “tomorrow” and their occurrence frequencies observed in the SMS corpus

tomoz (25)	tomrw (5)	tom (2)	moro (1)
tomorrow (24)	tomora (4)	tomra (2)	tmorro (1)
tomoro (12)	tomo (3)	tomor (2)	morrow (1)
2moro (9)	tomorow (3)	2mro (2)	tomm (1)

Aim

- Formally investigating the nature of SMS texts to build a word level model for the *texting language*.
- Using the model to develop a word level decoder for the *texting language*.

Proposed solution

- Model the problem as a noisy channel process.
- Assume that Standard word -> TL form.
- Use Hidden Markov models – capture stochastic properties.
- HMMs - combine graphemic and phonetic information of word.

Steps to be followed

1. Building a TL-SL word-aligner.
   (Here TL = Texting Lang. SL = Standard Lang.)
2. Building of word models based on HMM.
3. Estimation of parameters for the word-level HMMs.
4. Building a word level decoder to translate TL to Standard language.

Note:

Human-aided word-aligner - facilitates alignment of TL tokens with SL tokens.

Alignments - used for capture statistical data.

The Word Model

- Purpose – capture compressions employed by user.
- Word -> fragments -> equivalent letter sequence.
- Replacement independent of structure and fate of other sequences.
- Each fragment -> *graphemic* or *phonetic* equivalent.
- Above process modeled using Hidden Markov Models.

Constructing the HMM model of an SL word

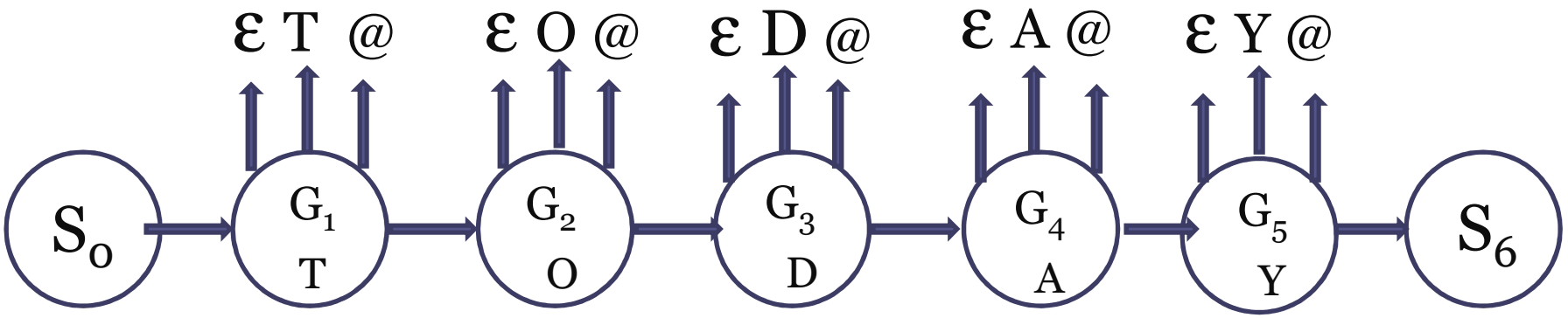
- Building the basic HMM
- Constructing the graphemic path
- Constructing the phonetic path
- Identifying the depicting cross-linkages
- Adding Extended State

HMM model for the word

"TODAY" Graphemic path

Typically for each letter there is a node

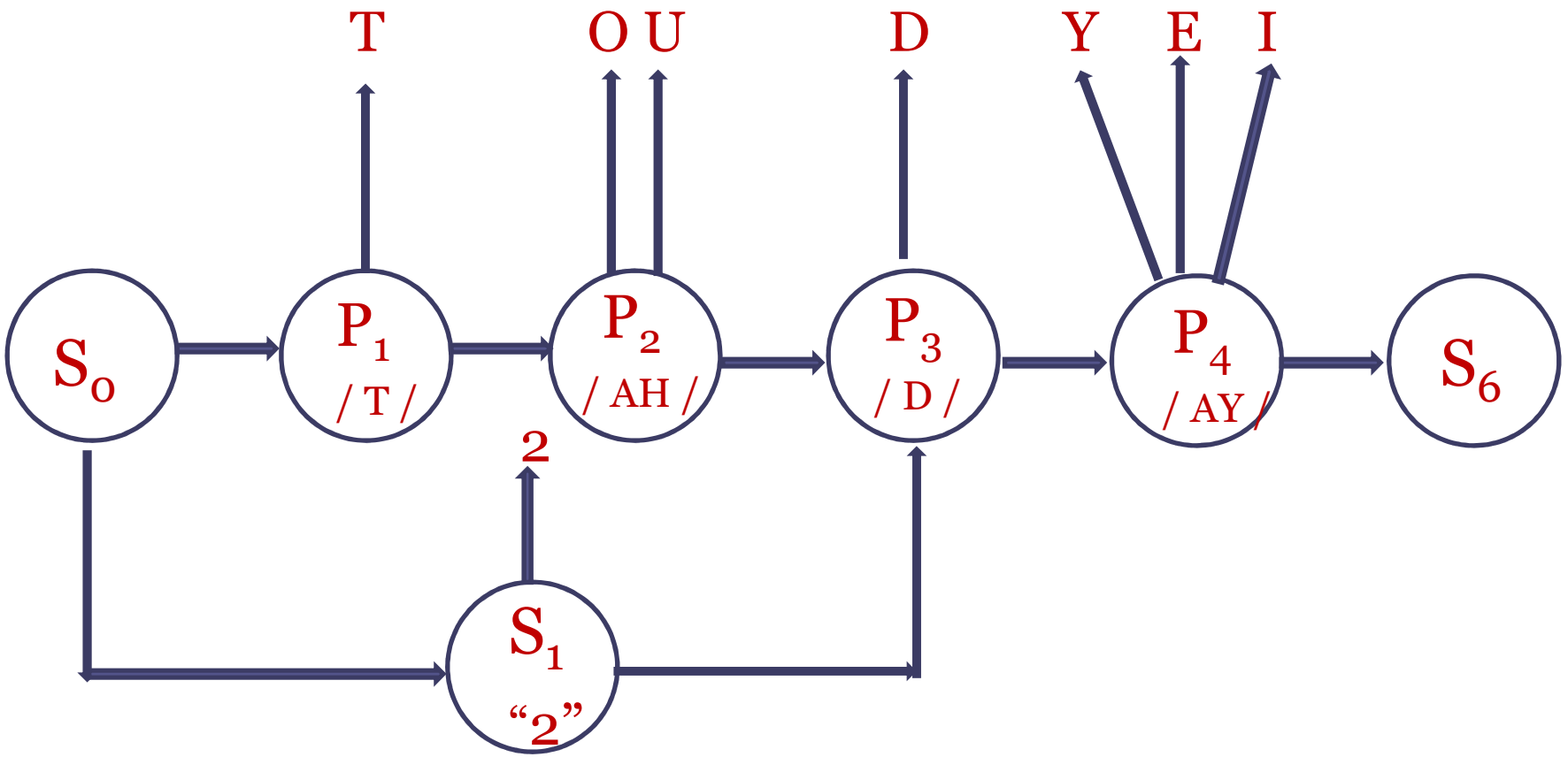
ε → Null observation @ → All other characters



HMM model for the word

"TODAY" Phonemic path

Each node corresponds to some phoneme



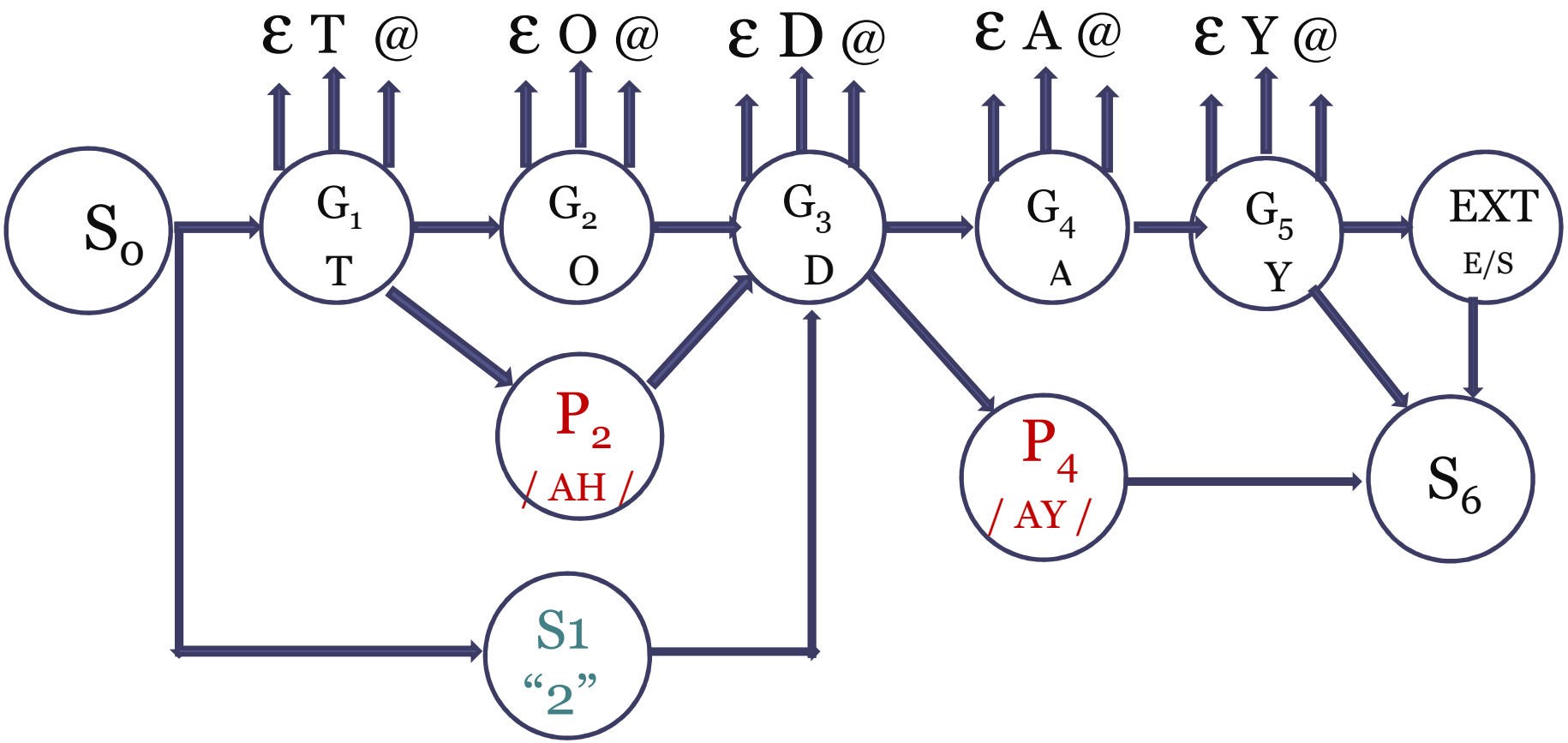
HMM model for the word

The final HMM is constructed on the basis of the from the two after:

1) Cross-linking
2) State minimization
3) Adding an Extra node, for (say rite <=> right)

HMM model for the word

The final HMM



HMM Model for Words

First the graphemic path was constructed.

The phonemic path was constructed using Phonetic dictionary.

The emissions were established manually from phoneme to character mapping

Similarly the grapheme to phoneme mappings.

The cross linkages were done.

Finally state minimization was done.

Such HMMs were constructed for 234 words by Monojit Choudhury et. Al..

Parameter estimation

- The most important aspect of the model design.
- The Expectation Maximization (EM) algorithm is typically used.
- Needs to be done for both Known and Unknown words.
- Parameter estimation steps for known words:

  i. Assigning initial parameter values.

  ii. Identifying the path for a given TL word.

  iii. Re-estimation of the parameters.

Handling of Unknown Words

For unknown words how to build a model is a good question.

Maybe one can use the probabilities for known words to estimate those of unknown words.

(e.g. TODAY for EVERYDAY)

Confusion Network type concepts maybe helpful.

Testing the word model – Sample output

TL Token (Gold Standard)	Decoder output	Rank
2DAY (TODAY)	TODAY(0.9), TRIED(0.069), THEY(0.017), TOOLS(0.01),THANKS(0.0095)	1
FNE (PHONE)	PHONE(0.86),PHONED(0.76),FINE(0.34),FINISHED(0.166), FINANCIAL(0.163)	1
ORITE (ALRIGHT)	ALRIGHT(0.33),OTHERWISE(0.08),ACCOUNT(0.07),AROUND(0.058)	1
CIN (SEEING)	SEEING(0.4),THINKS(0.36),THINK(0.29),THAN(0.285), THINKING(0.268)	1
CUZ (BECAUSE)	BECAUSE(0.21),COULD(0.20),COMING(0.038),COOK(0.0349), COME(0.0343)	1
CHK (CHECK)	TOUCH(0.48),CHECK(0.42),COULD(0.2),THIS(0.12),CHAR(0.04)	2
NYT (NIGHT)	NIGHT(0.81),NEXT(0.064),NEEDED(0.058),NET(0.048), NEITHER(0.047)	1

Epilogue

So far we have talked about *word* modeling through HMM.

For translation of *sentence* one has to work further.

Word Models need to be connected with some Language model.

Beam search based technique is also another option.

Another interesting area to pursue!!

Maximum Entropy Modeling

Maximum Entropy Modeling

Often we face situation when we need to
Choose one of several options:

- Which sense of a word.
- Which ordering of phrases.

Can we use logistic regression here?

Example

Consider the English word "bark".
Its Italian translations are:

noun abbaio, l'abbaiare, scorza,corteccia
Verb abbaiare, latrare, scorticare

the dog is barking >> il cane abbaia
barks of trees >> cortecce degli alberi
Barks of dogs >> latrati di cani

How to decide the proper word?
We need to put it in proper class.

Maximum Entropy Modeling

In logistic regression we have used:

$$p(c|x)=\sum_i w_i f_i$$

Under log-linear scheme we get the equation:

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

Where Z is the normalizing factor:

$$Z=\sum_k p(k|x)=\sum_k \exp\sum_i w_{ki} f_i$$

Maximum Entropy Modeling

Often in NLP the features f_i s are indicator function:

i.e $f_i = f_i(c, x)$ i.e. whether the *i*th feature belongs to Class c or not.

For example: to disambiguate palm, we can think of

Whether the word "hand" is there in the sentence or not.
Whether the word "tree" is there in the sentence or not.

Etc.

Maximum Entropy Modeling

Accordingly we can define our features:

$$f_1(c,x) = \begin{cases} 1 & \text{If the word "hand" occurs in the sentence} \\ 0 & \text{Otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{If the word "tree" occurs in the sentence} \\ 0 & \text{Otherwise} \end{cases}$$

$$f_3(c,x) = \begin{cases} 1 & \text{If the word has the suffix "}ing\text{"} \\ 0 & \text{Otherwise} \end{cases}$$

We can determine the weight of each features
From training data input and Logistic Regression.

Maximum Entropy Modeling

The intuition here is:

- To build a distribution by continuously adding features.
- Each feature is an indicator function – aimed at picking up a subset of the training example.
- For each feature we add a constraint on the total distribution, so that it matches with the empirical distribution found in the training data.
- We then choose a Maximum Entropy Model that otherwise accords with these constraints.

Hence it is called ME modeling.

PRINCIPLE OF MAXIMUM ENTROPY

The best probability model for the data is the one which maximizes entropy, over the set of probability distributions that are consistent with the evidence.

I have used Generalized Iterative Scaling Algorithm for the training.

We describe its application in Word Sense Disambiguation

WSD using MEM

Introduction

In this model

- The sense of the target word is considered as a random variable.

- Various outcomes are precisely the various senses in which the word can be used.

The model measures the bias of each surrounding word Towards each sense of the target word.

This is done using Contextual Predicates

Contextual Predicate

- The contextual predicate (cp) returns true or false.
- Indicates the presence or absence of useful information in the context.

$$cp : B \rightarrow \{true, \ false\}$$

where B is the set of all contexts.

- Once the presence of a keyword is noticed, the model takes into account which sense is being estimated.

$$f_{cp,a'}(a,b) = \begin{cases} 1 & \text{if } (a=a') \ \& \ cp(b)=true \\ 0 & \text{otherwise} \end{cases}$$

Feature Value Calculation

Suppose the model extracts k features from the training text.

A feature means a keyword-sense pair.

Each feature f_j (j = 1,2…k) has a weight α_j associated with it.

The features are then combined in the following way:

$$p(a \mid b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}$$

where $$Z(b) = \sum_{a} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}$$

Feature Value Calculation

The values of the α_j s that best fit the training data is Obtained from the MLE estimation:

$$Q = \left\{ p \mid p(a \mid b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \right\}$$

Q is the set of models of log-linear form,

$$L(p) = \sum_{a,b} \tilde{p}(a,b) \log p(a \mid b)$$

$\tilde{p}(a,b)$ is the probability of seeing (a,b) in the training set T

$$p^* = \arg\max_{q \in Q} L(q)$$

L(p) is the conditional log-likelihood of the training set T

p* is the optimal probability distribution according to the maximum likelihood criterion.

Theory suggests that the maximum likelihood parameter estimation for models of this form is equivalent to maximum entropy parameter estimation over the set of consistent models.

That is, $$p^* = \arg\max_{q \in Q} L(q) = \arg\max_{p \in P} H(p)$$

To obtain p* we have employed the Generalized Iterative Scaling (GIS) Algorithm, which improves the weights of the features iteratively, so that p converges to p*.

References in the next slides gives more details.

Further Reading

Adwait Ratnaparkhi "Maximum Entropy Models For Natural Language Ambiguity Resolution", A Dissertation in Computer and Information Science, Presented to the Faculties of the University of Pennsylvania, 1998.

Berger, A., Della Pietra, S. A., and Della Pietra, V. J. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 22(1):39, 1996.

J.N.Darroch and D.Radcliff , "Generalized Iterative Scaling for Log-linear Models", Flinders University of South Australia and C.S.I.R.O. Adelaide, The Annals of Mathematical Statistics, Vol.43,No.5,1470-1480, 1972.

Thank You