CS114 Lecture 8

Review

February 12, 2014

Professor Meteer

Thanks for Jurafsky & Martin & Prof. Pustejovksy for slides

Review

- Linguistics: Morphology, POS
- Ambiguity
- Grammars, FSAs
- Ngrams
  - What are some other applications? Spelling correction, text generation
- Viterbi algorithm and minimum distance
- Other applications of FSAs and HMMs

Terminology

- Morphology
  - Inflectional
  - Derivational
  - Regular vs irregular
- Parts of speech
  - Closed class
  - Open class
- Word forms vs Lemmas
  - In general tokens vs. types

Why care about morphology?

- `Stemming' in information retrieval
  – Might want to search for "going home" and find pages with both "went home" and "will go home"
- Morphology in machine translation
  – Need to know that the Spanish words quiero and quieres are both related to querer 'want'
- Morphology in spell checking
  – Need to know that misclaim and antiundoggingly are not words despite being made up of word parts

What we want

- Something to automatically do the following kinds of mappings:
- Cats cat +N +PL
- Cat cat +N +SG
- Cities city +N +PL
- Merging merge +V +Present-participle
- Caught catch +V +past-participle

Tokenization

- Segmenting words and sentences in running text
- Why not just periods and white-space?
  - Mr. Sherwood said reaction to Sea Containers' proposal has been "very positive." In New York Stock Exchange composite trading yesterday, Sea Containers closed at $62.625, up 62.5 cents.
  - "I said, 'what're you? Crazy?' " said Sadowsky. "I can't afford to do that.''
- **Words like: cents. said, positive." Crazy?**

Other "wordy" problems

- Tokenization
  - which are the words?
- Representing a dictionary using a FSA/FST
- Spell check
  - Edit distance (an example of what?)

Minimum Edit Distance Algorithm

- Create Matrix
- Initialize 1 – length in LH column and bottom row
- For each cell
  - Take the minimum of:
    - Deletion: +1 from cell below
    - Insertion: +1 from left cell
    - Substitution: Diagonal +0 if same +2 if different
  - Keep track of where you came from

Example

- Minimum of:
  - 1+1 (left right)
  - 1+1 (bottom up)
  - 0+0 (diagonal)
- Minimum of:
  - 0+1 (left right)
  - 2+1 (bottom up)
  - 1+2 (diagonal)

T	5				
H	4				
G	3				
I	2				
R	1				
#	0	1	2	3	4
	#	R	I	T	E

Answer to Right-Rite

T	5				
H	4				
G	3				
I	2				
R	1	2, 0, 2			
#	0	1	2	3	4
	#	R	I	T	E

In each box X, Y, Z values are

- X: From left: Insert-add one from left box
- Y: Diagonal, Compare-0 if same, 2 if different
- Z: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source
NOTE: All boxes will have arrows.
I didn't show them all.
Only one path back to root.

Answer to Right-Rite

T	5				
H	4				
G	3				
I	2	3, 3, 1	2, 0, 2		
R	1	2, 0, 2	1, 3, 3		
#	0	1	2	3	4
	#	R	I	T	E

In each box X, Y, Z values are

X: From left: Insert-add one from left box

Y: Diagonal, Compare-0 if same, 2 if different

Z: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source

NOTE: All boxes will have arrows. I didn't show them all.

Only one path back to root.

Answer to Right-Rite

T	5				
H	4	5, 5, 3	4, 4, 2		
G	3	4, 4, 2	3, 3, 1	2, 2, 2	
I	2	3, 3, 1	2, 0, 2	1, 3, 3	
R	1	2, 0, 2	1, 3, 3	2, 4, 4	
#	0	1	2	3	4
	#	R	I	T	E

In each box X, Y, Z values are

- X: From left: Insert-add one from left box
- Y: Diagonal, Compare-0 if same, 2 if different
- Z: From below: Delete-add one from lower box

Minimum is highlighted in red with arrow to source

NOTE: All boxes will have arrows. I didn't show them all.

Only one path back to root.

Answer to Right-Rite

T	5	6, 6, 4	5, 5, 5	6, 2, 4	3, 5, 5
H	4	5, 5, 3	4, 4, 2	3, 3, 3	4, 4, 4
G	3	4, 4, 2	3, 3, 1	2, 2, 2	3, 3, 3
I	2	3, 3, 1	2, 0, 2	1, 3, 3	2, 4, 4
R	1	2, 0, 2	1, 3, 3	2, 4, 4	3, 5, 5
#	0	1	2	3	4
	#	R	I	T	E

In each box X, Y, Z values are
- X: From left: Insert-add one from left box
- Y: Diagonal, Compare-0 if same, 2 if different
- Z: From below: Delete-add one from lower box

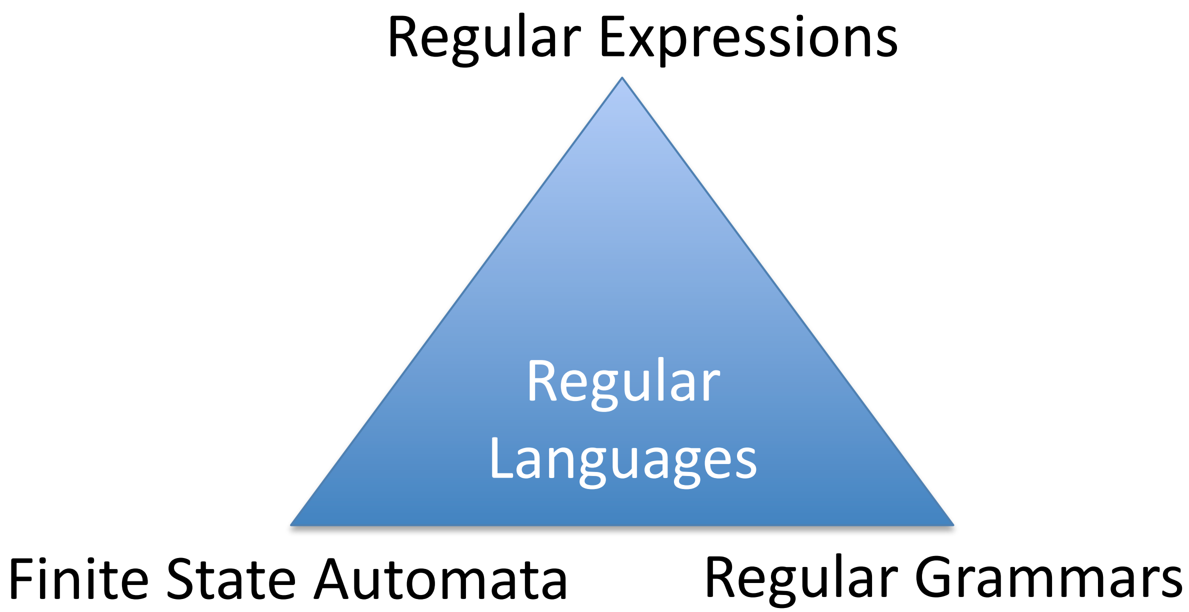
Minimum is highlighted in red with arrow to source
NOTE: All boxes will have arrows.
I didn't show them all.
Only one path back to root.

Summary

- Minimum Edit Distance
- A “dynamic programming” algorithm
- We will see a probabilistic version of this called “Viterbi”

Three Views

- Three equivalent formal ways to look at what we're up to



Three things you can do with any of these (FSA, FSG, RE)

- Generation
  - Produce all the strings in the language
- Recognition:
  - Is a string in a language
- Transducer
  - Use the FSA to transform one string into another

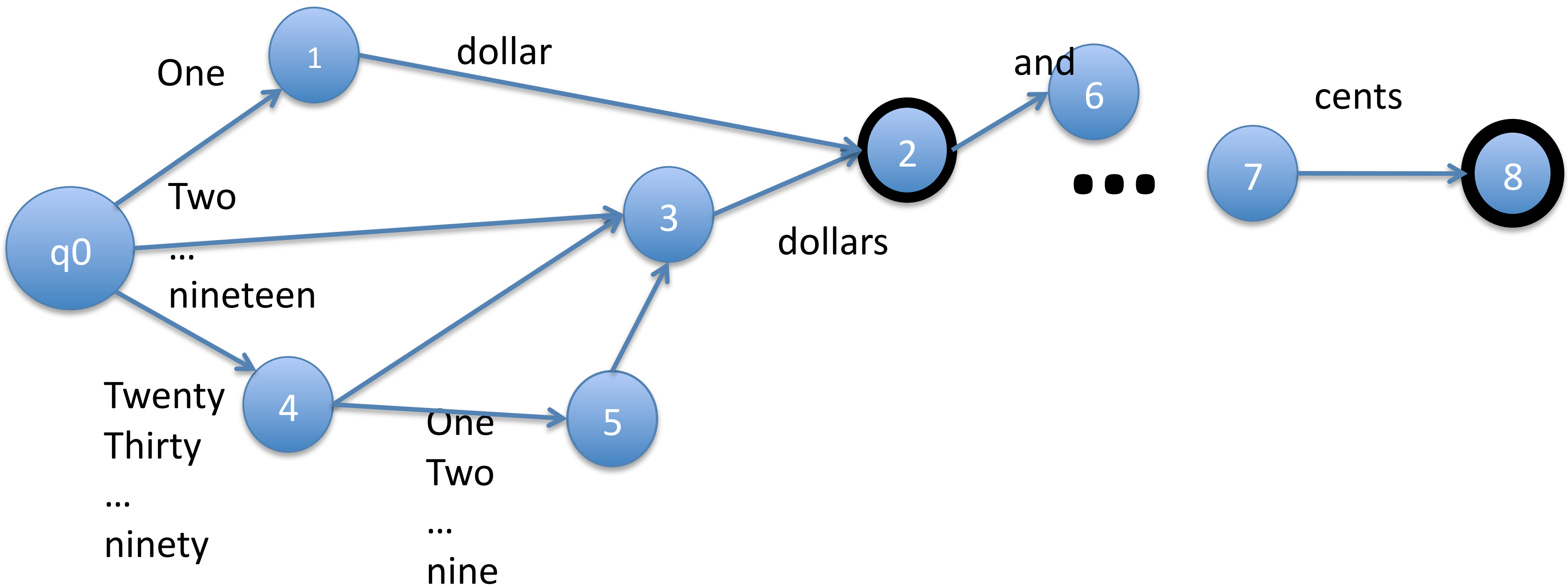
Finite State Automata

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.
- FSAs and their probabilistic relatives are at the core of much of what we'll be doing all semester.
- They also capture significant aspects of what linguists say we need for morphology and parts of syntax.

FSA’s time of day

- Think about the data
  - One o’clock
  - Five twenty three
  - Quarter to nine
  - Six oh four
  - Half past twelve

FSA for dollar amounts under $100.00



And as a grammar (note to make more readable, we go beyond finite state):

MONEY → DOLLARS CENTS?

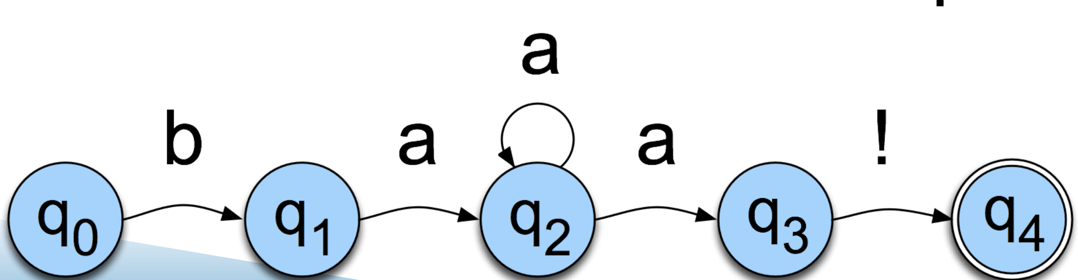
DOLLARS → one dollar | ONETOTEEN | TENS ONES?) dollars

"Operationalizing" FSAs with Transition Tables

- The guts of FSAs can ultimately be represented as tables

	b	a	!	e
0	1			
1		2		
2		2,3		
3			4	
4				

If you're in state 1 and you're looking at an a, go to state 2



Key Points

- Deterministic means that at each point in processing there is always one unique thing to do (no choices).
- D-recognize is a simple table-driven interpreter
- The algorithm is universal for all unambiguous regular languages.
  - To change the machine, you simply change the table.

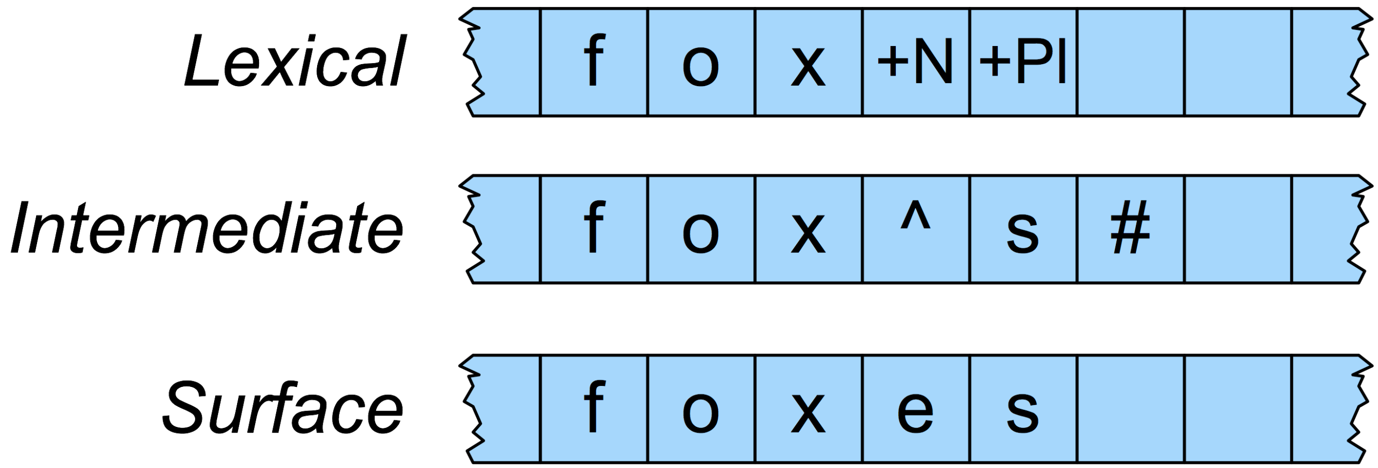
Key Points

- Crudely therefore… matching strings with regular expressions (a la Perl, grep, etc.) is a matter of
  - translating the regular expression into a machine (a table) and
  - passing the table and the string to an interpreter

Recognition as Search

- You can view this algorithm as a trivial kind of *state-space search*.
- States are pairings of tape positions and state numbers.
- Operators are compiled into the table
- Goal state is a pairing with the end of tape position and a final accept state
- It is trivial because?

Multi-Level Tape Machines



- We use one machine to transduce between the lexical and the intermediate level, and another to handle the spelling changes to the surface tape

Cascades

- This is an architecture that we'll see again and again
  - Overall processing is divided up into distinct rewrite steps
  - The output of one layer serves as the input to the next
  - The intermediate tapes may or may not wind up being useful in their own right

Languages and Grammars

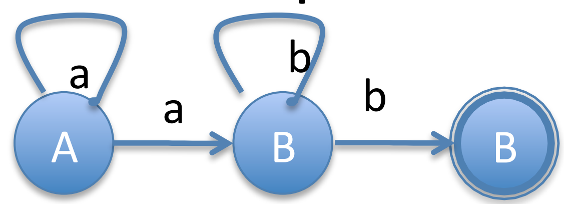
- We can model a language with a grammar
  - Production rules: LHS→ RHS
  - NonTerminals indicate a production rule can be applied
  - Terminals make up the "strings" (sentences) of the language
- The grammar defines all the possible strings of terminals in the language
  - A "language" is generally an infinite number of finite strings
  - Any string can be "accepted"/parsed by the grammar
  - The grammar can generate all the strings

The "Power" of Grammars

- The "Chomsky Hierarchy" shows that grammars written with different constraints on the rules generate different "languages"
- Finite state
  - A → Ab |a
- Context free
  - A → AB | a
- Context Sensitive
  - bAc → bac | cab

An Example

- Consider language $a^n b^n$
- Finite state grammar for a's and b's can't keep track
  - A → aA | aB
  - B → bB | b
  - L1: $a^m b^n$



- Context free grammar
  - S → aSb | ab
  - L2: $a^n b^n$
- Is L2 regular? No. Proof using the Pumping lemma left to the reader

The "C" in CL
(does it really just mean "count"?)

- Conditional probability
- Ngram Modeling
  - *N*-grams are token sequences of length *N*
  - Given knowledge of counts of N-grams such as these, we can guess likely next words in a sequence
- What to count?
  - Types vs. token
  - Wordforms vs. lemmas
  - Punctuation?

Language Modeling

- Back to word prediction
- We can model the word prediction task as the ability to assess the conditional probability of a word given the previous words in the sequence
  - $P(w_n | w_1, w_2 \ldots w_{n-1})$
- We'll call a statistical model that can assess this a *Language Model*

Other ngram concepts

- Perplexity
  - Computes the likelihood of a test set against a model, but nothing about the model per se
- Evaluation
  - Intrinsic vs. extrinsic
  - Separation of training and test
- What to do about gaps in the corpora
  - Unseen combinations
  - OOV

Backoff Vs. Interpolation

- **Backoff**: use trigram if you have it, otherwise bigram, otherwise unigram
- **Interpolation**: mix all three

GT Fish Example

- OR use the 1s for 0s (3/18 spread over2 species)
- AND Look at the things that happened 2s to share with 1s
  - C(whitefish) = 2 happened once
  - Discount 1s by 2/3
- LOTS OF ALTERNATIVES! Just estimates

	unseen (bass or catfish)	trout
c	0	1
MLE p	$p = \frac{0}{18} = 0$	$\frac{1}{18}$
c^*		$c^*(\text{trout}) = 2 \times \frac{N_2}{N_1} = 2 \times \frac{1}{3} = .67$
GT p^*_{GT}	$p^*_{\text{GT}}(\text{unseen}) = \frac{N_1}{N} = \frac{3}{18} = .17$	$p^*_{\text{GT}}(\text{trout}) = \frac{.67}{18} = \frac{1}{27} = .037$

Good-Turing

- Notation: N_x is the frequency-of-frequency-x
  - So $N_{10}=1$
    - Number of fish species seen 10 times is 1 (carp)
  - $N_1=3$
    - Number of fish species seen 1 is 3 (trout, salmon, eel)
- To estimate total number of unseen species
  - Use number of species (words) we've seen once
  - $c_0^* = c_1$ $p_0 = N_1/N$

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

- All other estimates are adjusted (down) to give probabilities for unseen

Good-Turing Intuition

- Notation: N_x is the frequency-of-frequency-x
  - So N_{10}=1, N_1=3, etc
- To estimate total number of unseen species
  - Use number of species (words) we've seen once
  - $c_0^* = c_1$ $p_0 = N_1/N$ $p_0=N_1/N=3/18$

$$P^*_{GT}(\text{things with frequency zero in training}) = \frac{N_1}{N}$$

- All other estimates are adjusted (down) to give probabilities for unseen

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

P(eel) = c*(1) = (1+1) 1/ 3 = 2/3

Slide from Josh Goodman

Could just spread 1s over 0s

Carp	10	10
Perch	3	3
WF	2	2
Trout	1	1
Salmon	1	1
Eel	1	1
Catfish	0	1
Bass	0	1
TOTAL	18	

- Prob of things that occurred once
- 1\18 + 1\18 + 1\18 = 3\18
- Add one to zero counts
- Spread probability over 1s and 0s
- (3/18) / 5 = .066

Part of Speech Tagging

- Parts of speech
  - What's POS tagging good for anyhow
- Important Ideas
  - Training sets and test sets
  - Unknown words
    - How can features help?
- HMM tagging

Hidden Markov Model Tagging

- Using an HMM to do POS tagging is a special case of *Bayesian inference*
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the "noisy channel" model that's the basis for ASR, OCR and MT

POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Two Kinds of Probabilities

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be:
  - Compute P(NN|DT) by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT,NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Kinds of Probabilities

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model (HMM)

HMMs for semantics

- Idea: use an HMM for semantics, just as we did for ASR (and part-of-speech tagging, etc)
- Hidden units:
  - Semantic slot names
    - Origin
    - Destination
    - Departure time
- Observations:
  - Word sequences

Semantics for a sentence

LIST FLIGHTS ORIGIN

Show me flights from Boston

DESTINATION DEPARTDATE

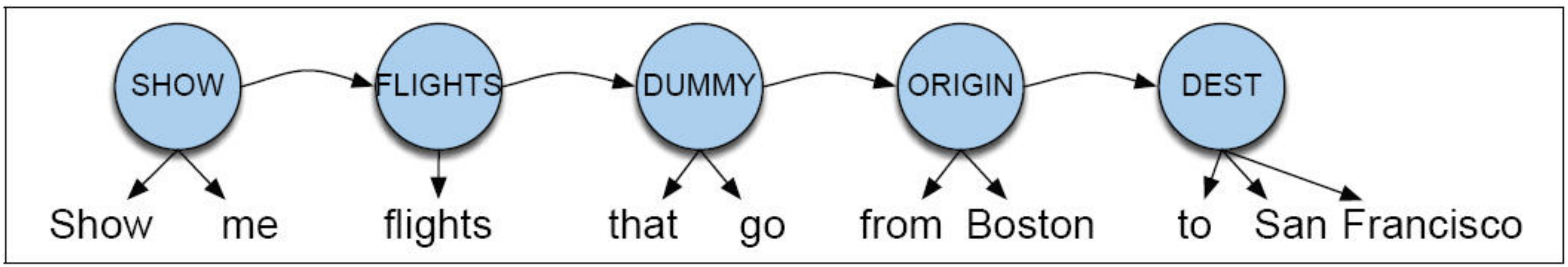
to San Francisco on Tuesday

DEPARTTIME

morning

HMM model of semantics – Pieraccini et al (1991)

- Input is the set of words
- Output is the set of semantic states



Decoding

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- We could just enumerate all paths given the input and use the model to assign probabilities to each.
  - Not a good idea.
  - Luckily dynamic programming (last seen in Ch. 3 with minimum edit distance) helps us here

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, (not all paths).