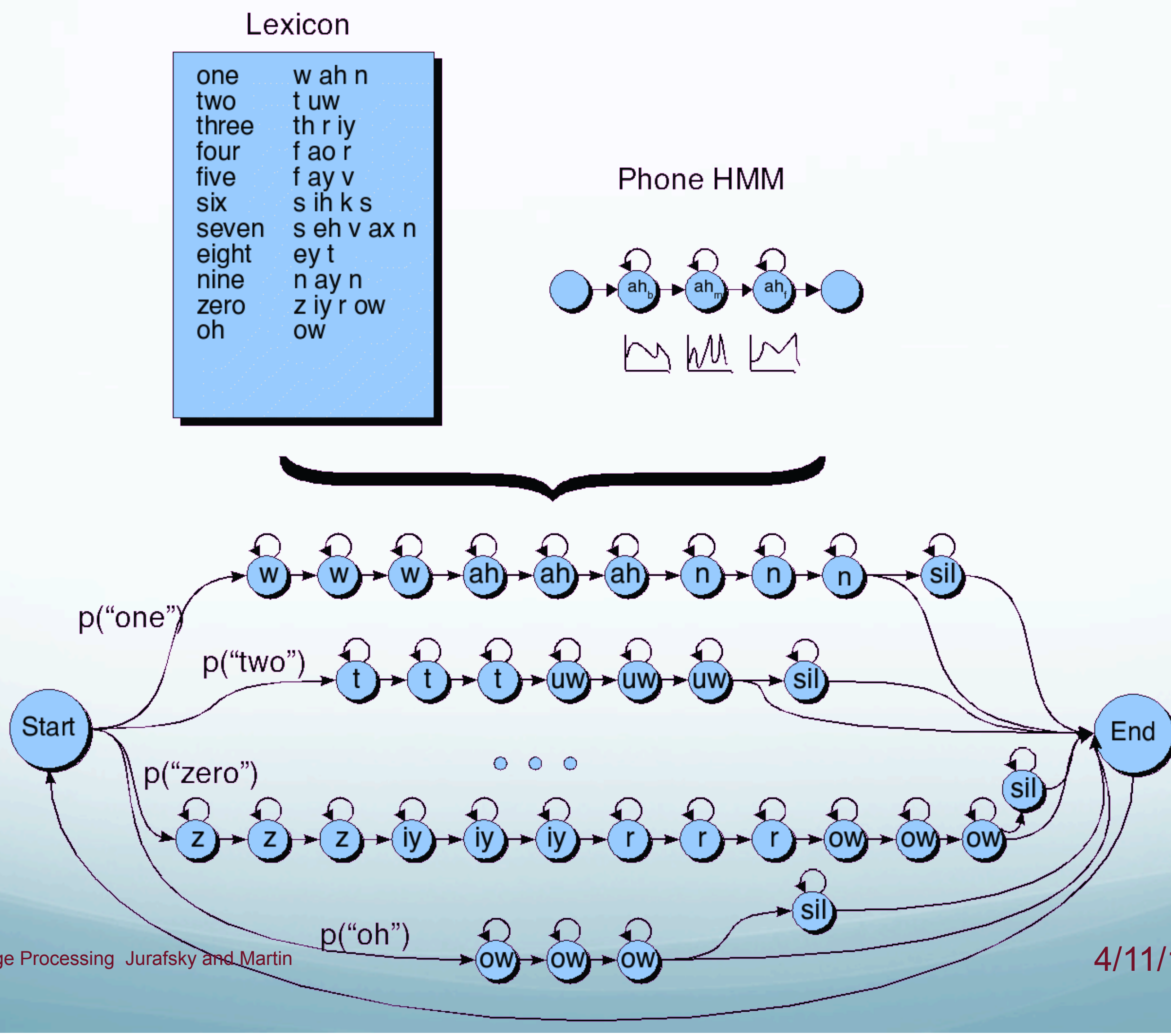
SDS: ASR, NLU, & VXML

Ling575
Spoken Dialog
April 16, 2015

Roadmap

- Dialog System components:
  - ASR: Noisy channel model
    - Representation
    - Decoding
  - NLU:
    - Call routing
    - Grammars for dialog systems
  - Basic interfaces: VoiceXML

HMM for the digit recognition task



Typical MFCC features

- Window size: 25ms
- Window shift: 10ms
- Pre-emphasis coefficient: 0.97
- MFCC:
  - 12 MFCC (mel frequency cepstral coefficients)
  - 1 energy feature
  - 12 delta MFCC features
  - 12 double-delta MFCC features
  - 1 delta energy feature
  - 1 double-delta energy feature
- Total 39-dimensional features

Why is MFCC so popular?

- Efficient to compute
- Incorporates a perceptual Mel frequency scale
- Separates the source and filter
- Fits well with HMM modelling

Decoding

- In principle:

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \overbrace{P(O|W)}^{\text{likelihood}} \; \overbrace{P(W)}^{\text{prior}}$$

- In practice:

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(O|W)P(W)^{LMSF}$$

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(O|W)P(W)^{LMSF}\, WIP^{N}$$

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} \log P(O|W) + LMSF \times \log P(W) + N \times \log WIP$$

Why is ASR decoding hard?

[ay d ih s hh er d s ah m th ih ng ax b aw m uh v ih ng r ih s en l ih]

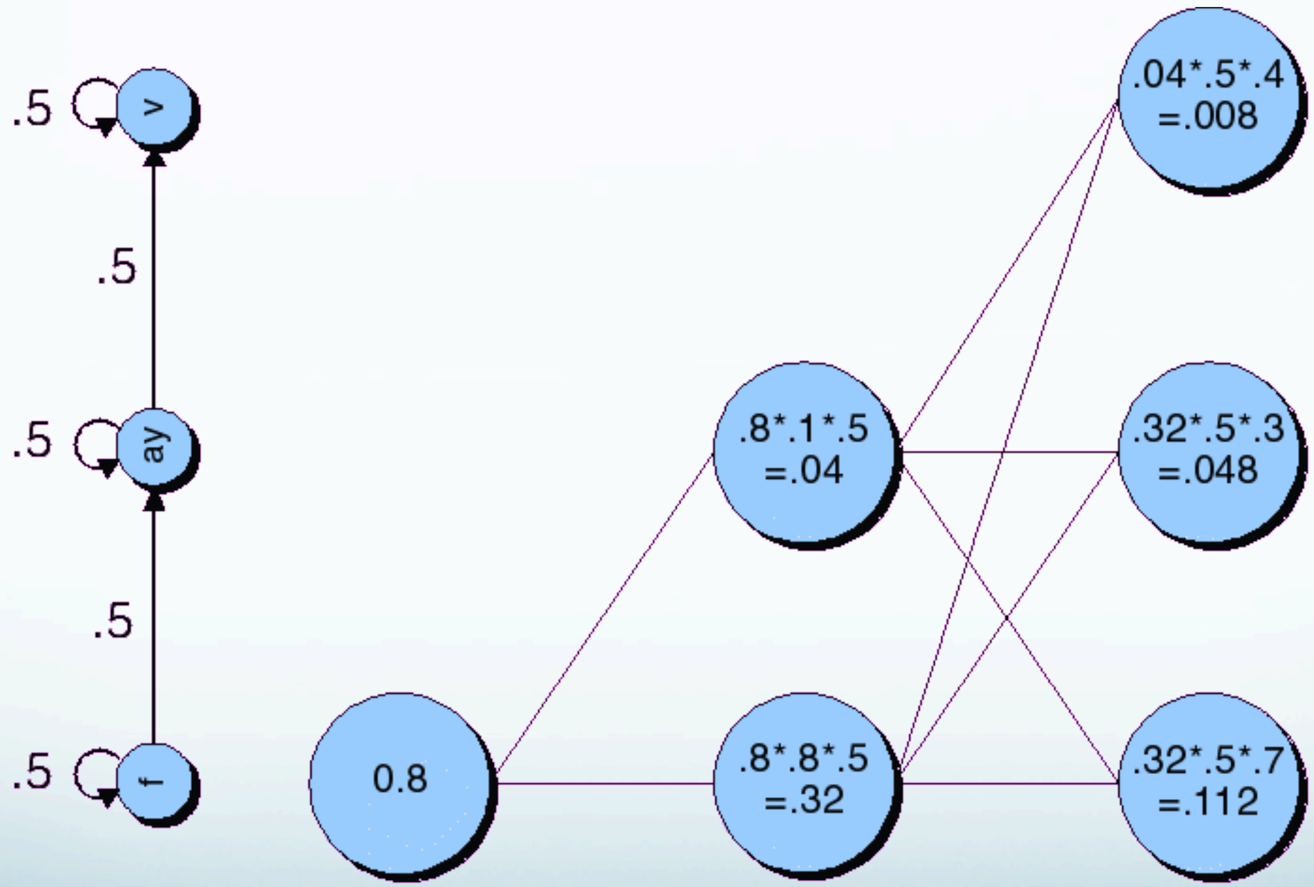
The Evaluation (forward) problem for speech

- The observation sequence O is a series of MFCC vectors
- The hidden states W are the phones and words
- For a given phone/word string W, our job is to evaluate P(O|W)
- Intuition: how likely is the input to have been generated by just that word string W

Evaluation for speech: Summing over all different paths!

- f ay ay ay ay v v v v
- f f ay ay ay ay v v v
- f f f f ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay ay ay ay ay ay ay ay v
- f f ay v v v v v v v

Viterbi trellis for "five"



Viterbi trellis for "five"

V	0	0	0.008	0.0072	0.00672	0.00403	0.00188	0.00161	0.000667	0.000493
AY	0	0.04	0.048	0.0448	0.0269	0.0125	0.00538	0.00167	0.000428	8.78e-05
F	0.8	0.32	0.112	0.0224	0.00448	0.000896	0.000179	4.48e-05	1.12e-05	2.8e-06
Time	1	2	3	4	5	6	7	8	9	10
B	*f* 0.8	*f* 0.8	*f* 0.7	*f* 0.4	*f* 0.4	*f* 0.4	*f* 0.4	*f* 0.5	*f* 0.5	*f* 0.5
	ay 0.1	*ay* 0.1	*ay* 0.3	*ay* 0.8	*ay* 0.8	*ay* 0.8	*ay* 0.8	*ay* 0.6	*ay* 0.5	*ay* 0.4
	v 0.6	*v* 0.6	*v* 0.4	*v* 0.3	*v* 0.3	*v* 0.3	*v* 0.3	*v* 0.6	*v* 0.8	*v* 0.9
	p 0.4	*p* 0.4	*p* 0.2	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.1	*p* 0.3	*p* 0.3
	iy 0.1	*iy* 0.1	*iy* 0.3	*iy* 0.6	*iy* 0.6	*iy* 0.6	*iy* 0.6	*iy* 0.5	*iy* 0.5	*iy* 0.4

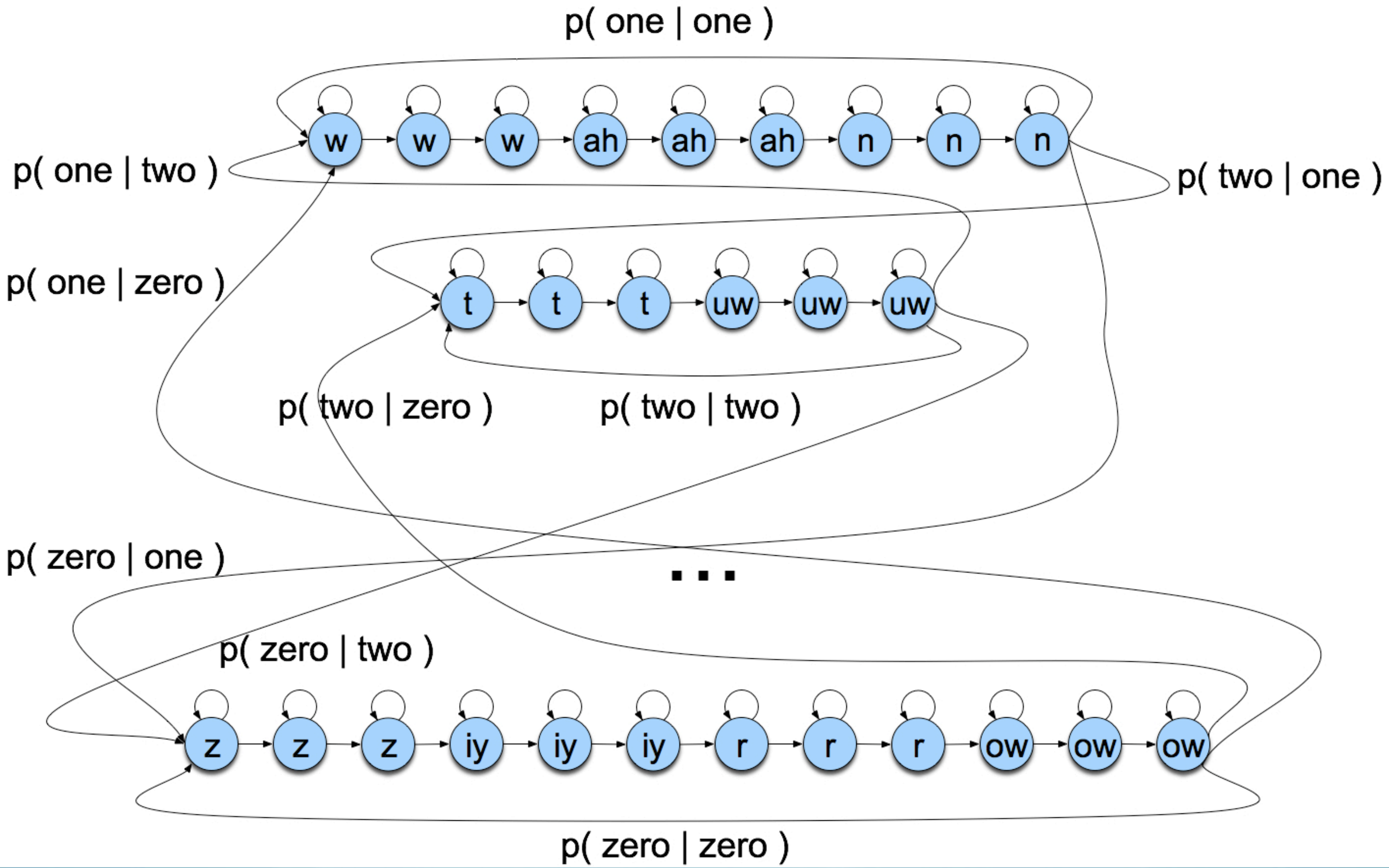
Language Model

- Idea: some utterances more probable

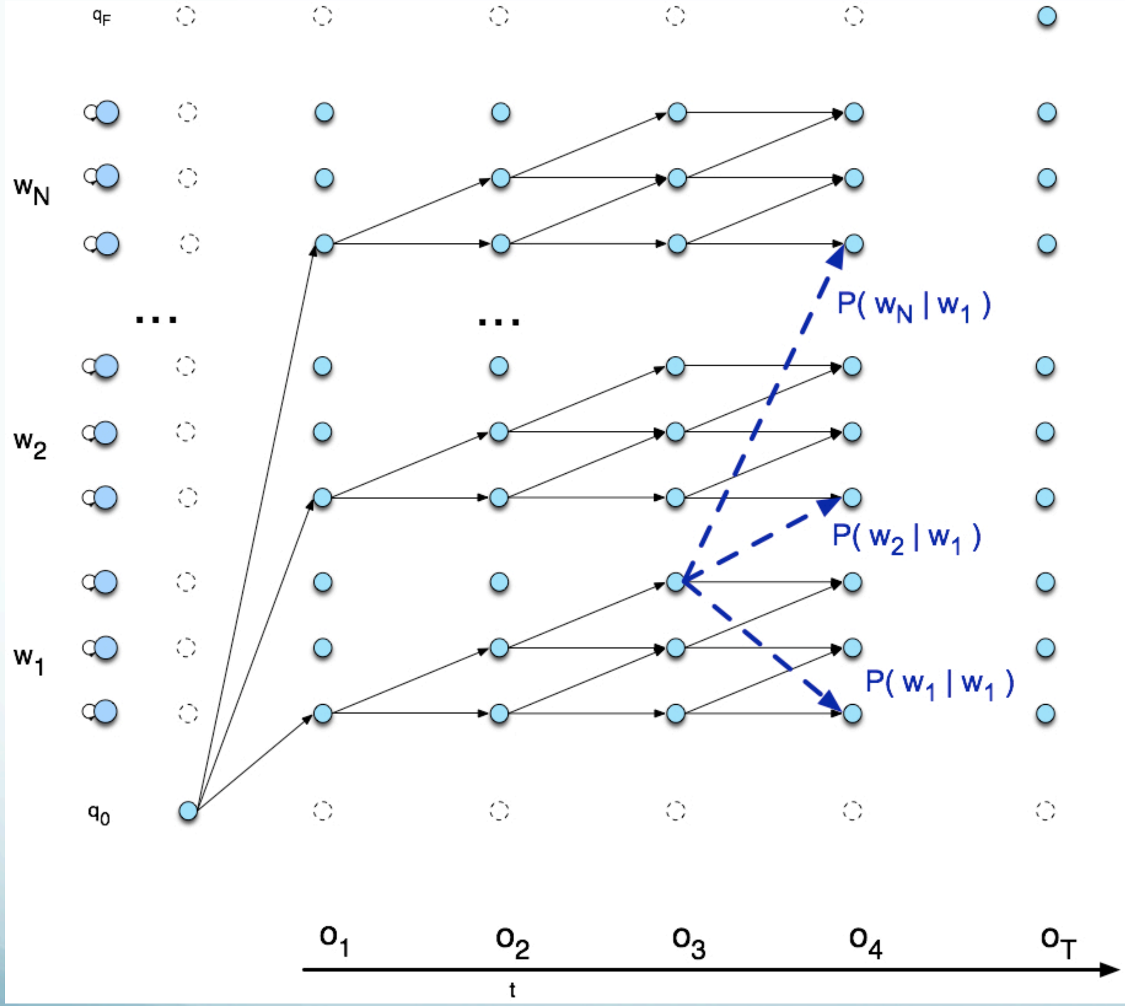
- Standard solution: “n-gram” model
  - Typically tri-gram: $P(w_i | w_{i-1}, w_{i-2})$
    - Collect training data from large side corpus
      - Smooth with bi- & uni-grams to handle sparseness
  - Product over words in utterance:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1}, w_{k-2})$$

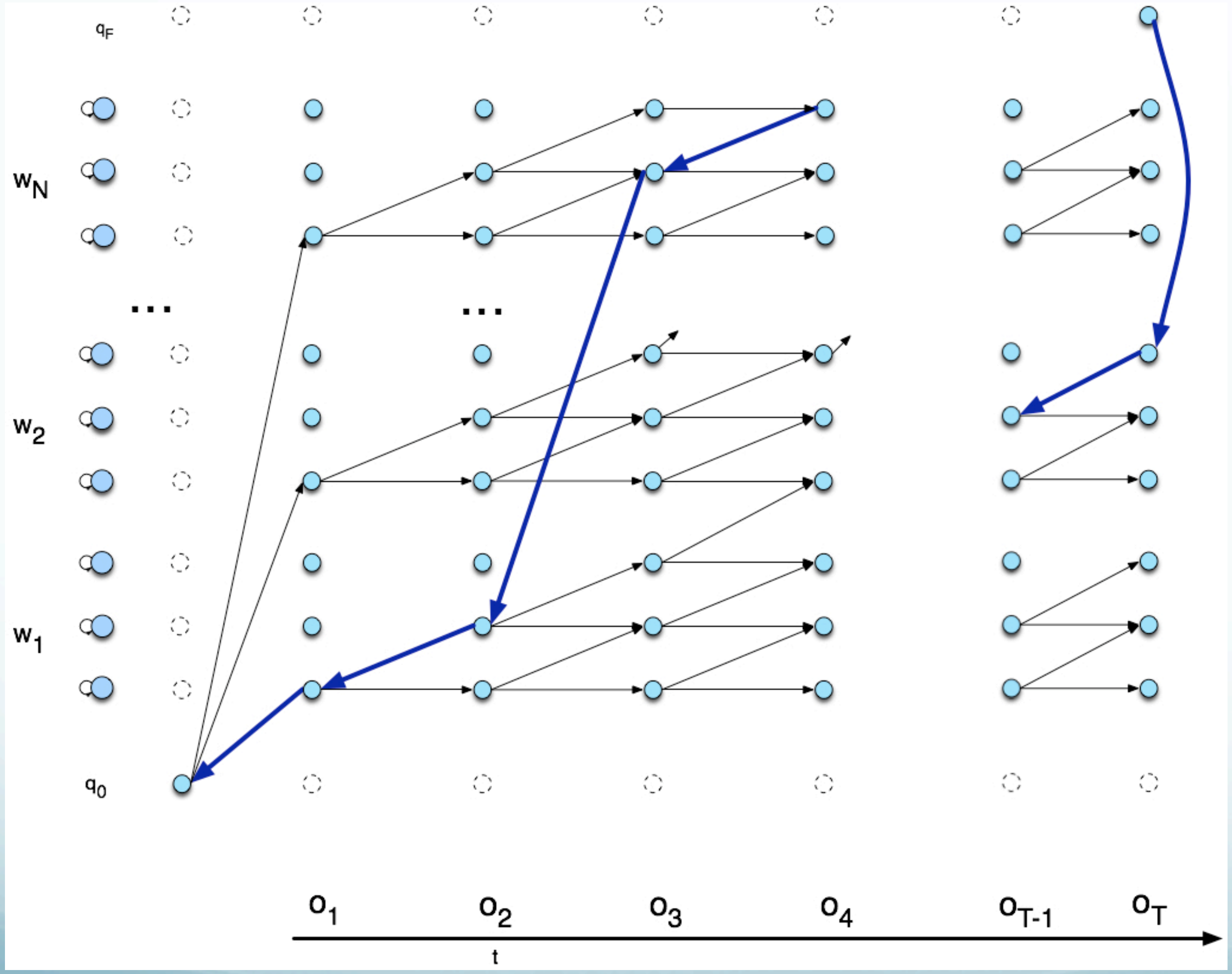
Search space with bigrams



Viterbi trellis

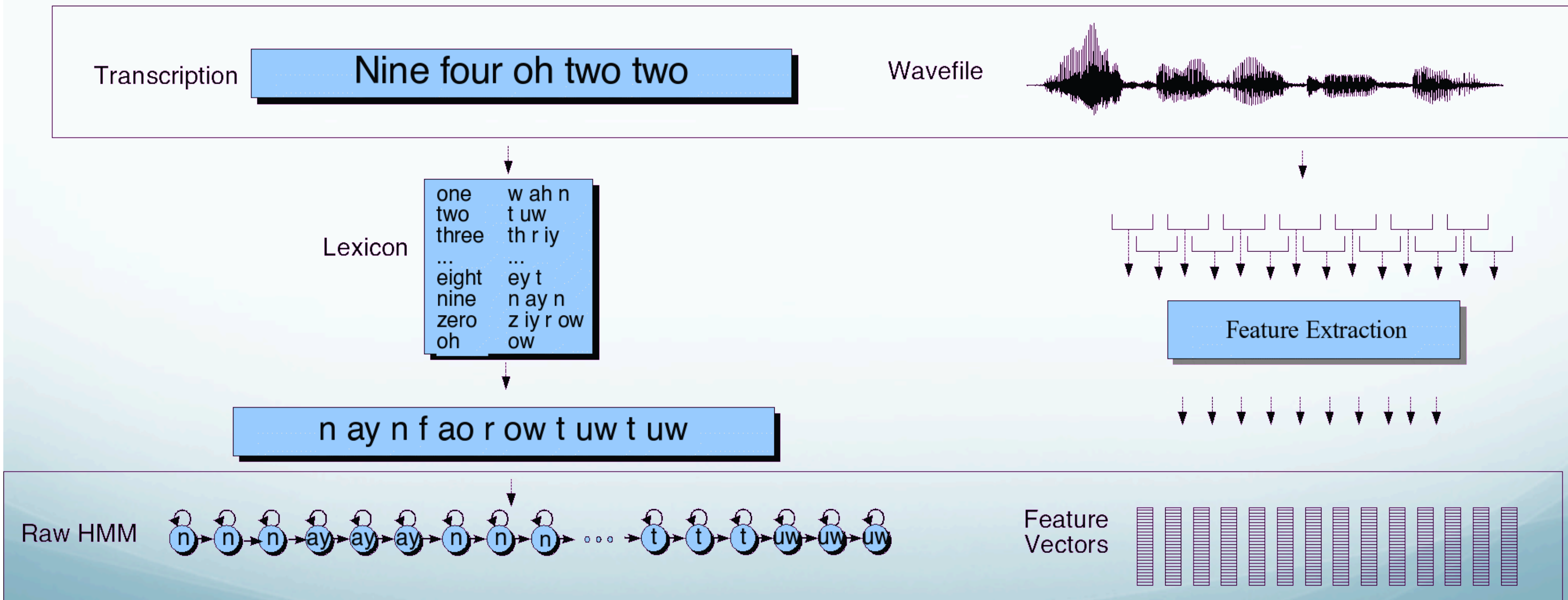


Viterbi backtrace



Training

- Trained using Baum-Welch algorithm



Summary: ASR Architecture

- Five easy pieces: ASR Noisy Channel architecture
  1) Feature Extraction:
     39 "MFCC" features
  2) Acoustic Model:
     Gaussians for computing p(o|q)
  3) Lexicon/Pronunciation Model
     - HMM: what phones can follow each other
  4) Language Model
     - N-grams for computing $p(w_i|w_{i-1})$
  5) Decoder
     - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!

Evaluation

- How to evaluate the word string output by a speech recognizer?

Word Error Rate

- Word Error Rate =

100 (Insertions+Substitutions + Deletions)

Total Word in Correct Transcript

Aligment example:

REF: portable **** PHONE UPSTAIRS last night so

HYP: portable FORM OF STORES last night so

Eval I S S

WER = 100 (1+2+0)/6 = 50%

NIST sctk-1.3 scoring software: Computing WER with sclite

- http://www.nist.gov/speech/tools/
- Sclite aligns a hypothesized text (HYP) (from the recognizer) with a correct or reference text (REF) (human transcribed)

```
id: (2347-b-013)

Scores: (#C #S #D #I) 9 3 1 2

REF:  was an engineer SO I   i was always with **** **** MEN UM   and they

HYP:  was an engineer ** AND i was always with THEM THEY ALL THAT and they

Eval:                 D  S                     I    I    S   S
```

Better metrics than WER?

- WER has been useful
- But should we be more concerned with meaning ("semantic error rate")?
  - Good idea, but hard to agree on
  - Has been applied in dialogue systems, where desired semantic output is more clear

Accents: An experiment

- A word by itself

- The word in context

Challenges for the Future

- Doing more with more
  - More applications:
    - From Siri, in-car navigation, call-routing
    - To full voice search, voice-based personal assistants, ubiquitous computing
  - More speech types:
    - Accented speech
    - Speech in noise
    - Overlapping speech
    - Child speech
    - Speech pathology

NLU for Dialog Systems

Natural Language Understanding

- Generally:
  - Given a string of words representing a natural language utterance, produce a meaning representation

Natural Language Understanding

- Generally:
  - Given a string of words representing a natural language utterance, produce a meaning representation
- For well-formed natural language text (see ling571),
  - Full parsing with a probabilistic context-free grammar
    - Augmented with semantic attachments in FOPC
    - Producing a general lambda calculus representation

Natural Language Understanding

- Generally:
  - Given a string of words representing a natural language utterance, produce a meaning representation
- For well-formed natural language text (see ling571),
  - Full parsing with a probabilistic context-free grammar
    - Augmented with semantic attachments in FOPC
    - Producing a general lambda calculus representation
- What about spoken dialog systems?

NLU for SDS

- Few SDS fully exploit this approach

NLU for SDS

- Few SDS fully exploit this approach
- Why not?

NLU for SDS

- Few SDS fully exploit this approach
- Why not?
  - Examples of travel air speech input (due to A. Black)
    - Eh, I wanna go, wanna go to Boston tomorrow
    - If its not too much trouble I'd be very grateful if one might be able to aid me in arranging my travel arrangements to Boston, Logan airport, at sometime tomorrow morning, thank you.
    - Boston, tomorrow

NLU for SDS

- Analyzing speech vs text

NLU for SDS

- Analyzing speech vs text
  - Utterances:
    - ill-formed, disfluent, fragmentary, desultory, rambling
      - Vs well-formed

NLU for SDS

- Analyzing speech vs text
  - Utterances:
    - ill-formed, disfluent, fragmentary, desultory, rambling
      - Vs well-formed
  - Domain:
    - Restricted, constrains interpretation
      - Vs. unrestricted

NLU for SDS

- Analyzing speech vs text
  - Utterances:
    - ill-formed, disfluent, fragmentary, desultory, rambling
      - Vs well-formed
  - Domain:
    - Restricted, constrains interpretation
      - Vs. unrestricted
  - Interpretation:
    - Need specific pieces of data
      - Vs. full, complete representation

NLU for SDS

- Analyzing speech vs text
  - Utterances:
    - ill-formed, disfluent, fragmentary, desultory, rambling
      - Vs well-formed
  - Domain:
    - Restricted, constrains interpretation
      - Vs. unrestricted
  - Interpretation:
    - Need specific pieces of data
      - Vs. full, complete representation
  - Speech recognition:
    - Error-prone, perfect full analysis difficult to obtain

NLU for Spoken Dialog

- Call routing (aka call classification):
  - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
- Shallow form of NLU

NLU for Spoken Dialog

- Call routing (aka call classification):
  - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
- Shallow form of NLU
- Goal:
  - Given a spoken utterance, assign to class c, in finite set C

NLU for Spoken Dialog

- Call routing (aka call classification):
  - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
  - Shallow form of NLU
  - Goal:
    - Given a spoken utterance, assign to class c, in finite set C
  - Banking Example:
    - Open prompt: **"How may I direct your call?"**

NLU for Spoken Dialog

- Call routing (aka call classification):
  - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
- Shallow form of NLU
- Goal:
  - Given a spoken utterance, assign to class c, in finite set C
- Banking Example:
  - Open prompt: **"How may I direct your call?''**
  - Responses: may I have consumer lending?,

NLU for Spoken Dialog

- Call routing (aka call classification):
    - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
  - Shallow form of NLU
  - Goal:
    - Given a spoken utterance, assign to class c, in finite set C
  - Banking Example:
    - Open prompt: **"How may I direct your call?''**
    - Responses: may I have consumer lending?,
      - I'd like my checking account balance, or

NLU for Spoken Dialog

- Call routing (aka call classification):
  - (Chu-Carroll & Carpenter, 1998, Al-Shawi 2003)
- Shallow form of NLU
- Goal:
  - Given a spoken utterance, assign to class c, in finite set C
- Banking Example:
  - Open prompt: **"How may I direct your call?''**
  - Responses: may I have consumer lending?,
    - I'd like my checking account balance, or
    - "ah I'm calling 'cuz ah a friend gave me this number and ah she told me ah with this number I can buy some cars or whatever but she didn't know how to explain it to me so I just called you you know to get that information."

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model: Vector of word unigram, bigrams, trigrams
    - Filtering:

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model: Vector of word unigram, bigrams, trigrams
    - Filtering: by frequency

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model: Vector of word unigram, bigrams, trigrams
    - Filtering: by frequency
      - Exclude high frequency stopwords, low frequency rare words
    - Weighting

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model: Vector of word unigram, bigrams, trigrams
    - Filtering: by frequency
      - Exclude high frequency stopwords, low frequency rare words
    - Weighting: term frequency * inverse document frequency

Call Routing

- General approach:
  - Build classification model based on labeled training data, e.g. manually routed calls
    - Apply classifier to label new data
- Vector-based call routing:
  - Model: Vector of word unigram, bigrams, trigrams
    - Filtering: by frequency
      - Exclude high frequency stopwords, low frequency rare words
    - Weighting: term frequency * inverse document frequency
    - (Dimensionality reduction by singular value decomposition)
  - Compute cosine similarity for new call & training examples

Meaning Representations for Spoken Dialog

- Typical model: Frame-slot semantics
  - Majority of spoken dialog systems
    - Almost all deployed spoken dialog systems

Meaning Representations for Spoken Dialog

- Typical model: Frame-slot semantics
  - Majority of spoken dialog systems
    - Almost all deployed spoken dialog systems

- Frame:
  - Domain-dependent information structure
  - Set of attribute-value pairs
  - Information relevant to answering questions in domain

Natural Language Understanding

- Most systems use frame-slot semantics
  Show me morning flights from Boston to SFO on Tuesday

  - SHOW:
  - FLIGHTS:
    - ORIGIN:
      - CITY: Boston
      - DATE:
        - DAY-OF-WEEK: Tuesday
      - TIME:
        - PART-OF-DAY: Morning
    - DEST:
      - CITY: San Francisco

Another NLU Example

- Sagae et 2009
- Utterance (speech): we are prepared to give you guys generators for electricity downtown
- ASR (NLU input): we up apparently give you guys generators for a letter city don town
- Frame (NLU output):
  - **<s>.mood declarative**
  - **<s>.sem.agent kirk**
  - **<s>.sem.event deliver**
  - **<s>.sem.modal.possibility can**
  - **<s>.sem.speechact.type offer**
  - **<s>.sem.theme power-generator**
  - **<s>.sem.type event**

Question

- Given an ASR output string, how can we tractably and robustly derive a meaning representation?

Question

- Given an ASR output string, how can we tractably and robustly derive a meaning representation?
- Many approaches:
  - Shallow transformation:
    - Terminal substitution

Question

- Given an ASR output string, how can we tractably and robustly derive a meaning representation?
- Many approaches:
  - Shallow transformation:
    - Terminal substitution
  - Integrated parsing and semantic analysis
    - E.g. semantic grammars

Question

- Given an ASR output string, how can we tractably and robustly derive a meaning representation?
- Many approaches:
  - Shallow transformation:
    - Terminal substitution
  - Integrated parsing and semantic analysis
    - E.g. semantic grammars
  - Classification or sequence labeling approaches
    - HMM-based, MaxEnt-based

Grammars

- Formal specification of strings in a language
- A 4-tuple:
  - A set of terminal symbols: Σ
  - A set of non-terminal symbols: N
  - A set of productions P: of the form A -> α
  - A designated start symbol S
- In regular grammars:
  - A is a non-terminal and α is of the form {N}Σ*
- In context-free grammars:
  - A is a non-terminal and α in (Σ U N)*

Simple Air Travel Grammar

- LIST -> show me | I want | can I see|…
- DEPARTTIME -> (after|around|before) HOUR| morning | afternoon | evening
- HOUR -> one|two|three…|twelve (am|pm)
- FLIGHTS -> (a) flight|flights
- ORIGIN -> from CITY
- DESTINATION -> to CITY
- CITY -> Boston | San Francisco | Denver | Washington

Shallow Semantics

- Terminal substitution
  - Employed by some speech toolkits, e.g. CSLU

Shallow Semantics

- Terminal substitution
  - Employed by some speech toolkits, e.g. CSLU
- Rules convert terminals in grammar to semantics
  - LIST -> show me | I want | can I see|…

Shallow Semantics

- Terminal substitution
  - Employed by some speech toolkits, e.g. CSLU
- Rules convert terminals in grammar to semantics
  - LIST -> show me | I want | can I see|…
    - e.g. show -> LIST
    -

Shallow Semantics

- Terminal substitution
  - Employed by some speech toolkits, e.g. CSLU
- Rules convert terminals in grammar to semantics
  - LIST -> show me | I want | can I see|…
    - e.g. show -> LIST
    - see -> LIST
    - I -> ε
    - can -> ε
    - * Boston -> Boston

Shallow Semantics

- Terminal substitution
  - Employed by some speech toolkits, e.g. CSLU
- Rules convert terminals in grammar to semantics
  - LIST -> show me | I want | can I see|…
    - e.g. show -> LIST
    - see -> LIST
    - I -> ε
    - can -> ε
    - * Boston -> Boston
- Simple, but…
  - VERY limited, assumes direct correspondence

Semantic Grammars

- Domain-specific semantic analysis

Semantic Grammars

- Domain-specific semantic analysis
- Syntactic structure:
  - Context-free grammars (CFGs) (typically)
    - Can be parsed by standard CFG parsing algorithms
      - e.g. Earley parsers or CKY

Semantic Grammars

- Domain-specific semantic analysis
- Syntactic structure:
  - Context-free grammars (CFGs) (typically)
    - Can be parsed by standard CFG parsing algorithms
      - e.g. Earley parsers or CKY
- Semantic structure:
  - Some designated non-terminals correspond to slots
    - Associate terminal values to corresponding slot

Semantic Grammars

- Domain-specific semantic analysis
- Syntactic structure:
  - Context-free grammars (CFGs) (typically)
    - Can be parsed by standard CFG parsing algorithms
      - e.g. Earley parsers or CKY
- Semantic structure:
  - Some designated non-terminals correspond to slots
    - Associate terminal values to corresponding slot
- Frames can be nested
- Widely used: Phoenix NLU (CU, CMU), vxml grammars

Show me morning flights from Boston to SFO on Tuesday

- LIST -> show me | I want | can I see|…
- DEPARTTIME -> (after| around|before) HOUR| morning | afternoon | evening
- HOUR -> one|two|three…| twelve (am|pm)
- FLIGHTS -> (a) flight|flights
- ORIGIN -> from CITY
- DESTINATION -> to CITY
- CITY -> Boston | San Francisco | Denver | Washington

- SHOW:
- FLIGHTS:
  - ORIGIN:
    - CITY: Boston
    - DATE:
      - DAY-OF-WEEK: Tuesday
    - TIME:
      - PART-OF-DAY: Morning
  - DEST:
    - CITY: San Francisco

Semantic Grammars: Issues

- Issues:

Semantic Grammars: Issues

- Issues:
  - Generally manually constructed
    - Can be expensive, hard to update/maintain

Semantic Grammars: Issues

- Issues:
  - Generally manually constructed
    - Can be expensive, hard to update/maintain
  - Managing ambiguity:
    - Can associate probabilities with parse & analysis
    - Build rules manually, then train probabilities w/data

Semantic Grammars: Issues

- Issues:
  - Generally manually constructed
    - Can be expensive, hard to update/maintain
  - Managing ambiguity:
    - Can associate probabilities with parse & analysis
    - Build rules manually, then train probabilities w/data
  - Domain- and application-specific
    - Hard to port

Learning Probabilistic Slot Filling

- Goal: Use machine learning to map from recognizer strings to semantic slots and fillers

Learning Probabilistic Slot Filling

- Goal: Use machine learning to map from recognizer strings to semantic slots and fillers
- Motivation:
  - Improve robustness – fail-soft
  - Improve ambiguity handling – probabilities
  - Improve adaptation – train for new domains, apps

Learning Probabilistic Slot Filling

- Goal: Use machine learning to map from recognizer strings to semantic slots and fillers
- Motivation:
  - Improve robustness – fail-soft
  - Improve ambiguity handling – probabilities
  - Improve adaptation – train for new domains, apps
- Many alternative classifier models
  - HMM-based, MaxEnt-based

HMM-Based Slot Filling

- Find best concept sequence C given words W

HMM-Based Slot Filling

- Find best concept sequence C given words W
- $C^* = \text{argmax } P(C|W)$
-

HMM-Based Slot Filling

- Find best concept sequence C given words W
- $C^* = \text{argmax } P(C|W)$
- $= \text{argmax } P(W|C)P(C)/P(W)$
-

HMM-Based Slot Filling

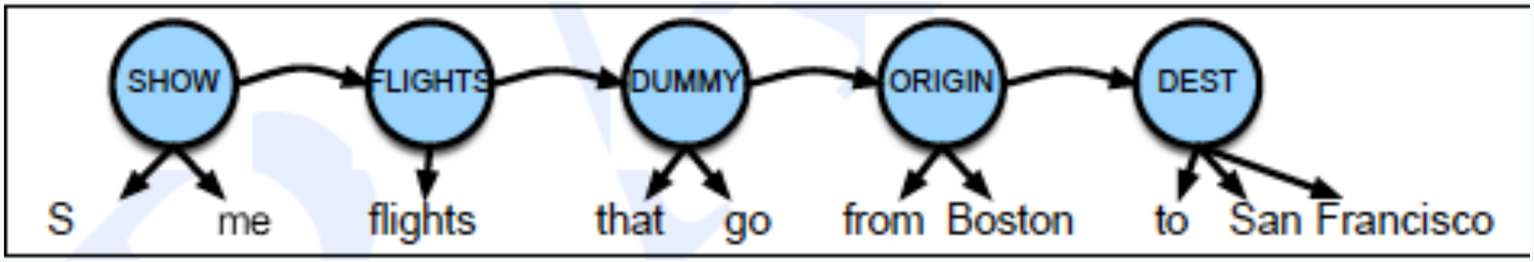
- Find best concept sequence C given words W
- $C^* = \text{argmax}\ P(C|W)$
- $= \text{argmax}\ P(W|C)P(C)/P(W)$
- $= \text{argmax}\ P(W|C)P(C)$

HMM-Based Slot Filling

- Find best concept sequence C given words W
- C*= argmax P(C|W)
-     = argmax P(W|C)P(C)/P(W)
-     = argmax P(W|C)P(C)
- Assume limited M-concept history, N-gram words
  - = $$\prod_{i=2}^{N} P(w_i \mid w_{i-1} \ldots w_{i-N+1}, c_i) \prod_{i=2}^{N} P(c_i \mid c_{i-1} \ldots c_{i-M+1})$$

Probabilistic Slot Filling

- Example HMM



VoiceXML

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps
    - Structures voice interaction

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps
    - Structures voice interaction
    - Can incorporate Javascript/PHP/etc for functionality

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative
  - Can implement frame-based mixed initiative

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative
  - Can implement frame-based mixed initiative
  - Support for sub-dialog call-outs

Speech I/O

- ASR:
  - Supports speech recognition defined by
    - Grammars
    - Trigrams
    - Domain managers: credit card nos etc

Speech I/O

- ASR:
  - Supports speech recognition defined by
    - Grammars
    - Trigrams
    - Domain managers: credit card nos etc
- TTS:
  - <ssml> markup language
  - Allows choice of: language, voice, pronunciation
  - Allows tuning of: timing, breaks

Simple VoiceXML Example

- Minimal form:

```
<form>
  <field name="transporttype">
     <prompt>
        Please choose airline, hotel, or rental car.
     </prompt>
     <grammar type="application/x=nuance-gsl">
       [airline hotel "rental car"]
     </grammar>
  </field>
  <block>
    <prompt>
     You have chosen <value expr="transporttype">.
    </prompt>
  </block>
</form>
```

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name=“transporttype”>

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name="transporttype">
    - Prompt for user input
      - <prompt> Please choose airline, hotel, or rental car.</prompt>
        - Can include URL for recorded prompt, backs off

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name="transporttype">
    - Prompt for user input
      - <prompt> Please choose airline, hotel, or rental car.</prompt>
        - Can include URL for recorded prompt, backs off
    - Specify grammar to recognize/interpret user input
      - <grammar>[airline hotel "rental car"]</grammar>

Other Field Elements

- Context-dependent help:
  - <help>Please select activity.</help>

Other Field Elements

- Context-dependent help:
  - <help>Please select activity.</help>

- Action to be performed on input:
  - <filled>
    - <prompt>You have chosen <value exp="transporttype">.
    - </prompt></filled>

Control Flow

- Default behavior:
  - Step through elements of form in document order

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next=“weather.xml”>
  - Other position in form: <goto next=“#departdate”>
-

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next="weather.xml">
  - Other position in form: <goto next="#departdate">
- Conditionals:
  - <if cond="varname=='air'">....</if>

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next="weather.xml">
  - Other position in form: <goto next="#departdate">
- Conditionals:
  - <if cond="varname=='air'">....</if>
- Guards:
  - Default: Skip field if slot value already entered

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout
- <nomatch>:
  - Speech, but nothing valid recognized

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout
- <nomatch>:
  - Speech, but nothing valid recognized
- <help>:
  - General system help prompt

Complex Interaction

- Preamble, grammar:

```
<noinput>    I'm sorry, I didn't hear you.  <reprompt/>  </noinput>

<nomatch> I'm sorry, I didn't understand that.  <reprompt/> </nomatch>

<form>
   <grammar type="application/x=nuance-gsl">
    <![ CDATA[
    Flight (   ?[
                 (i [wanna (want to)] [fly go])
                 (i'd like to [fly go])
                 ([(i wanna)(i'd like a)] flight)
            ]
            [
              ( [from leaving departing] City:x) {<origin $x>}
              ( [(?going to)(arriving in)] City:x) {<destination $x>}
              ( [from leaving departing] City:x
                [(?going to)(arriving in)] City:y) {<origin $x> <destination $y>}
            ]
            ?please
     )
     City [ [(san francisco) (s f o)] {return( "san francisco, california")}
            [(denver) (d e n)] {return( "denver, colorado")}
            [(seattle) (s t x)] {return( "seattle, washington")}
     ]
     ]]> </grammar>

  <initial name="init">
     <prompt> Welcome to the consultant. What are your travel plans?  </prompt>
  </initial>
```

Mixed Initiative

- With guard defaults

```
  <field name="origin">
     <prompt> Which city do you want to leave from?  </prompt>
     <filled>
        <prompt> OK, from <value expr="origin"> </prompt>
     </filled>
  </field>
  <field name="destination">
     <prompt> And which city do you want to go to?  </prompt>
     <filled>
        <prompt> OK, to <value expr="destination"> </prompt>
     </filled>
  </field>
  <block>
     <prompt> OK, I have you are departing from  <value expr="origin">
              to  <value expr="destination">.  </prompt>
    send the info to book a flight...
  </block>
</form>
```

Complex Interaction

- Preamble, external grammar:

```
<?xml version="1.0"?>
<vxml version = "2.0">

<form id="F1">

  <field name="F_1">
      <grammar src="NameGram.xml"
type="application/grammar-xml" />
      <prompt>
      Please tell me your full name so I can verify you
      </prompt>
  </field>

  <filled mode="all" namelist="F_1">
      <prompt>
        Your name is <value expr="F_1"/>
        <break strength="medium"/>
      </prompt>
    </filled>
</form>
</vxml>
```

Multi-slot Grammar

```
<?xml version= "1.0"?>
 <grammar xml:lang="en-US" root = "TOPLEVEL">
  <rule id="TOPLEVEL" scope="public">
   <item>

  <!-- FIRST NAME RETURN -->

    <item repeat="0-1">
     <ruleref uri="#FIRSTNAME"/>
     <tag>out.firstNameSlot=rules.FIRSTNAME.firstNameSubslot;</tag>
    </item>
  <!-- MIDDLE NAME RETURN -->

    <item repeat="0-1">
     <ruleref uri="#MIDDLENAME"/>
     <tag>out.middleNameSlot=rules.MIDDLENAME.middleNameSubslot;</tag>
    </item>
  <!-- LAST NAME RETURN -->

     <ruleref uri="#LASTNAME"/>
     <tag>out.lastNameSlot=rules.LASTNAME.lastNameSubslot;</tag>
    </item>

 <!-- TOP LEVEL RETURN-->
   <tag> out.F_1= out.firstNameSlot + out.middleNameSlot + out.lastNameSlot; </tag>
  </rule>
```

Multi-slot Grammar II

```
<rule id="FIRSTNAME" scope="public">
  <one-of>
    <item> matt<tag>out.firstNameSubslot="matthew";</tag></item>
    <item> dee <tag> out.firstNameSubslot="dee ";</tag></item>
    <item> jon <tag> out.firstNameSubslot="jon ";</tag></item>
    <item> george <tag>out.firstNameSubslot="george ";</tag></item>
    <item> billy <tag> out.firstNameSubslot="billy ";</tag></item>
  </one-of>
  </rule>

  <rule id="MIDDLENAME" scope="public">
  <one-of>
    <item> bon <tag>out.middleNameSubslot="bon ";</tag></item>
    <item> double ya <tag> out.middleNameSubslot="w ";</tag></item>
    <item> dee <tag> out.middleNameSubslot="dee ";</tag></item>
  </one-of>
  </rule>

  <rule id="LASTNAME" scope="public">
  <one-of>
    <item> henry <tag> out.lastNameSubslot="henry "; </tag></item>
    <item> ramone <tag> out.lastNameSubslot="dee ";  </tag></item>
    <item> jovi <tag> out.lastNameSubslot="jovi ";  </tag></item>
    <item> bush <tag> out.lastNameSubslot=""bush ";  </tag></item>
    <item> williams <tag> out.lastNameSubslot="williams "; </tag></item>
  </one-of>
  </rule>

</grammar>
```

Augmenting VoiceXML

- Don't write XML directly
  - Use php or other system to generate VoiceXML
    - Used in 'Let's Go Dude' bus info system

Augmenting VoiceXML

- Don’t write XML directly
  - Use php or other system to generate VoiceXML
    - Used in ‘Let’s Go Dude’ bus info system
- Pass input to other web services
  - i.e. to RESTful services

Augmenting VoiceXML

- Don’t write XML directly
  - Use php or other system to generate VoiceXML
    - Used in ‘Let’s Go Dude’ bus info system
- Pass input to other web services
  - i.e. to RESTful services
- Access web-based audio for prompts