Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Automatic Speech Recognition

Tobias Scheffer

Motivation

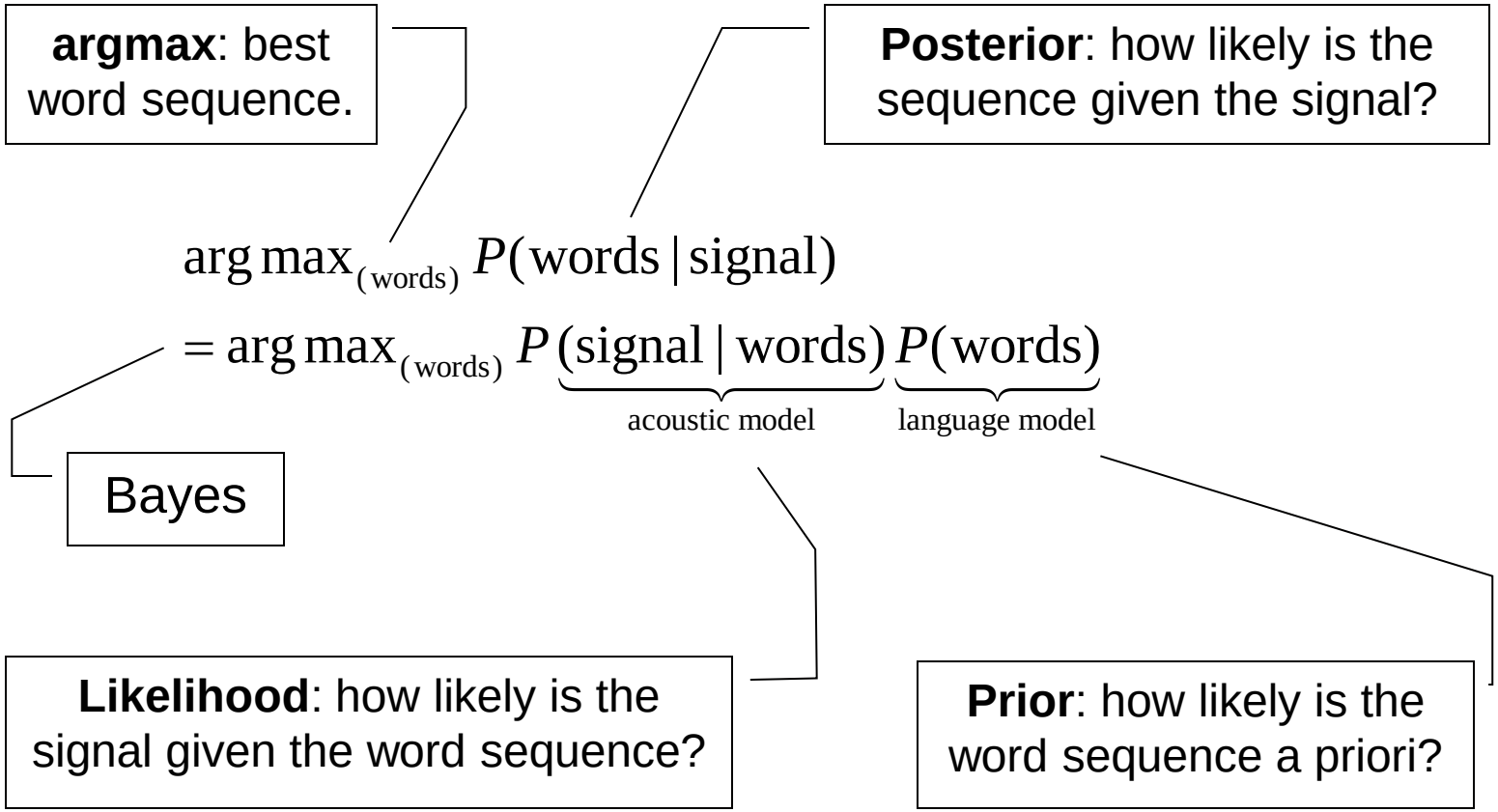
- Spoken language is an extremely natural form of human communication.
- Does not require a keyboard.
- Does not require eyes or hands to be available.
- Can be implemented on phone, car, embedded device.

Overview

- Speech recognition usind hidden Markov models.
- Speech recognition using deep recurrent neural networks.
- Voice portals.

Statistical Speech Recognition

- Components: acoustic model + language model



Statistical Speech Recognition

$$\arg\max_{(\text{words})} P(\text{words} \mid \text{signal})$$

$$= \arg\max_{(\text{words})} P\underbrace{(\text{signal} \mid \text{words})}_{\text{acoustic model}} \underbrace{P(\text{words})}_{\text{language model}}$$

Language model has to be trained on domain-specific documents.

Statistical Speech Recognition

$$\arg\max_{(\text{words})} P(\text{words} \mid \text{signal})$$

$$= \arg\max_{(\text{words})} P\underbrace{(\text{signal} \mid \text{words})}_{\text{acoustic model}}\underbrace{P(\text{words})}_{\text{language model}}$$

Acoustic model:

- Trained on annotated voice signal.
- Phonetic models for individual phones.
- Phones combined into models of words.

Statistical Speech Recognition

Decoding (applying speech recognition):

- Search in the space of all word sequences.
- Search algorithms, Viterbi beam search.

$$\arg\max_{(\text{words})} P(\text{words} \mid \text{signal})$$

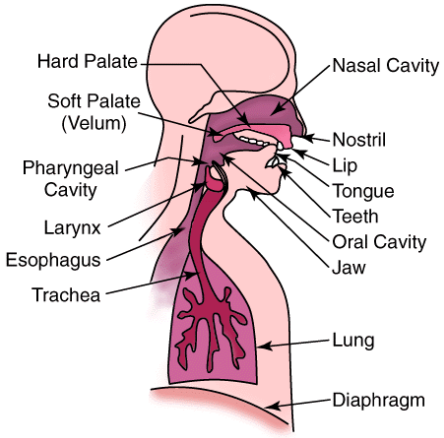
$$= \arg\max_{(\text{words})} P\underbrace{(\text{signal} \mid \text{words})}_{\text{acoustic model}} \underbrace{P(\text{words})}_{\text{language model}}$$

Acoustic Model: Granularity

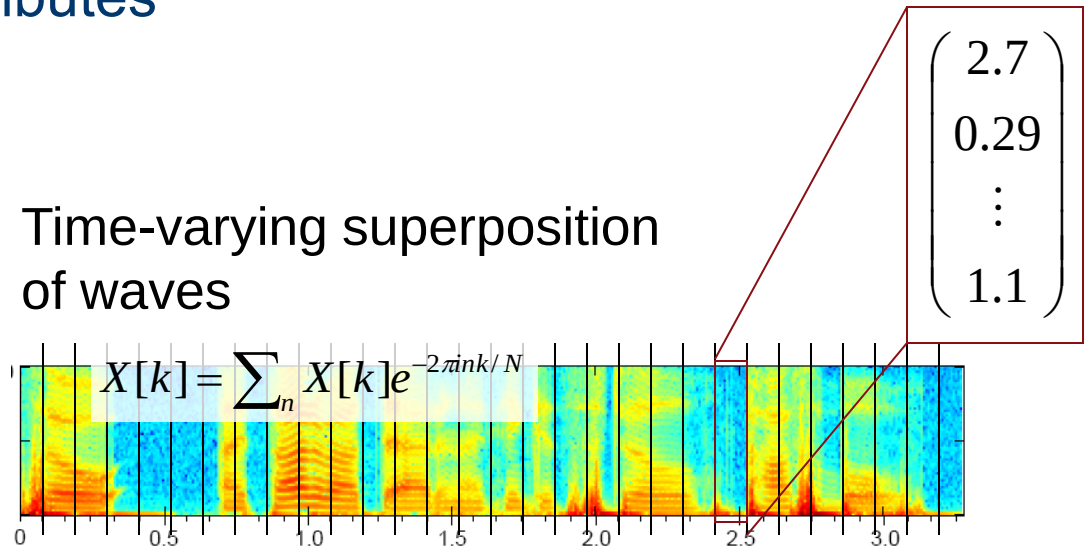
- One model per word.
  - Works for small volabularies (e.g., digit recognition).
  - Cannot recognize unknown words.
  - Ignores that distinct words often share phones.
- Syllables
  - Works for Japanese (50 syllables),
  - But generally bad (English has 30,000 syllables).
- Phone:
  - Smallest phonetic unit; English has 50 phones.
  - Phoneme: smallest phonetic unit that distinguishes word meaning.
  - Pronounciation of phones varies by context.

Acoustic Model

- Short-Time-Fourier-Transform:
  - Signal × sin(frequency φ) = amplitude of φ in signal.
- For each point in time:
  - Amplitudes of ~ 24 frequency bands.
  - Decorrelation, feature selection → ~20-50 continuous "cepstral attributes"



Time-varying superposition of waves



Acoustic Model: Varyations

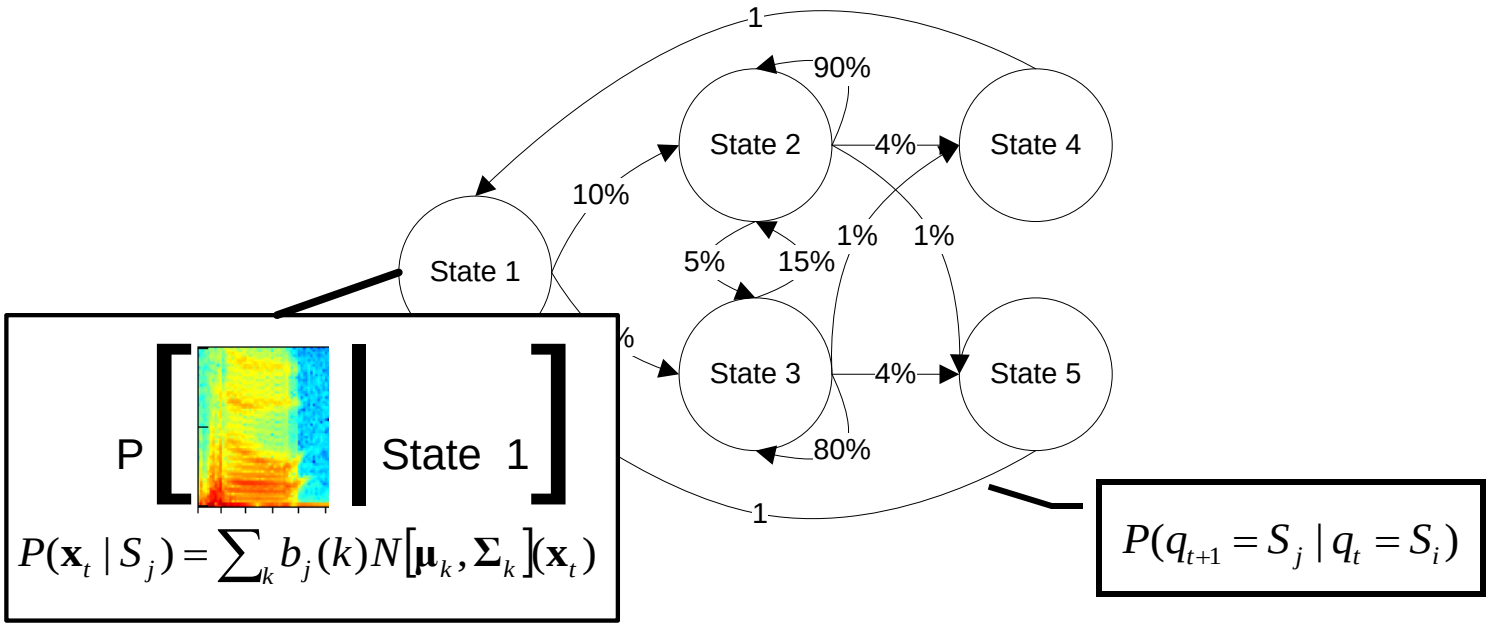
- One separate model per phone.
- Alternatively: Triphone class model:
  - Pronounciation depends on neighboring phones.
  - For instance, “b” and “p” have a similar influence on the following vowel.
  - Group such context phones into classes, one acoustic model per phone with each class of neighbors.

Acoustic Model

- Each phone (or phone + context) is represented by a hidden Markov model.
- ~ 2-20 states per phone.
- Time steps of ~ 10 milliseconds.
- Observations $b_i(O_t)$: vector of cepstral attributes.
- Mostly linear structure, but
  - Shortcuts and parallel branches reflect the possible pronounciation alternatives.

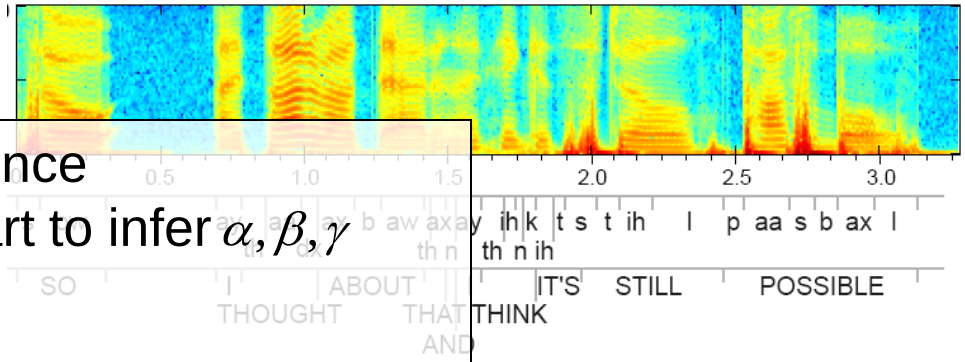
Acoustic Model: Hidden Markov Model

- State sequence cannot be observed, only sequence of acoustic signals.
- Observation probabilities $b_i(O_t)$ modeled as mixture of gaussian distributions.



Hidden Markov Model

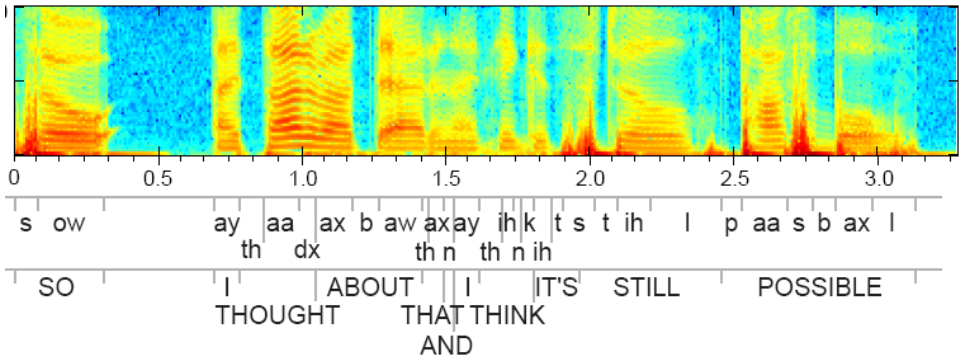
- Parameter estimation:
  - From annotated speech data
  - Baum Welch algorithm
  - Find $\arg\max_{\theta} P(\text{Data}|\theta)$.



Repeat until convergence
- Use forward-backwart to infer α, β, γ
- Infer $\xi_t(i,j)$
- Estimate $\pi_i^{(k)} = \gamma_1(i)$
- Estimate $a_{ij}^{(k)} = \sum_t \xi_t(i,j) \Big/ \sum_t \gamma_t(i)$
- Estimate $b_i^{(k)}(O) = \sum_{t:O_t=O} \gamma_t(i) \Big/ \sum_t \gamma_t(i)$

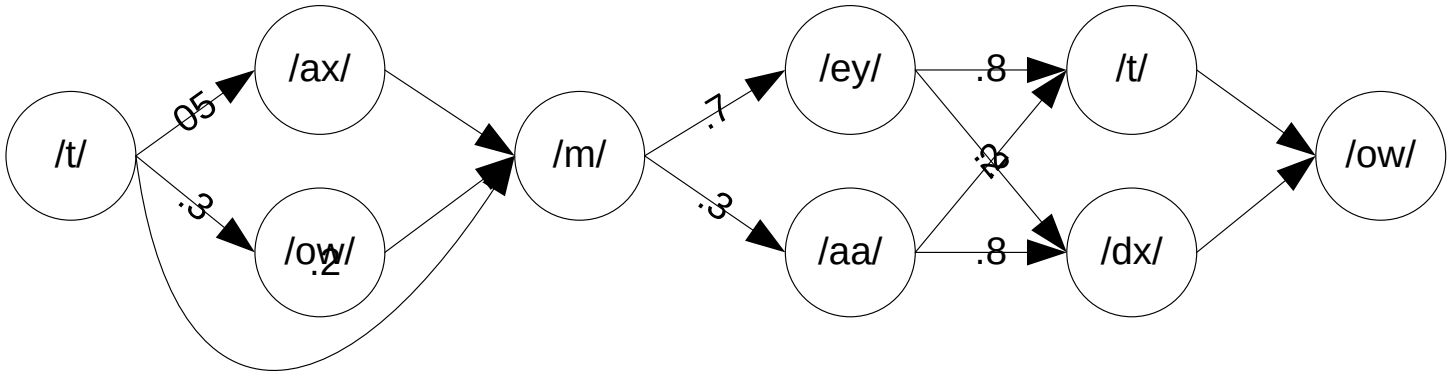
Hidden Markov Acoustic Models

- Training data: phonetically annotated speech.
- Alignment is critical. HMM that models a phone can only be trained with speech signals of that phone.
- Phonetic annotation is a cumbersome process that cannot easily be scaled.



Pronounciation Networks

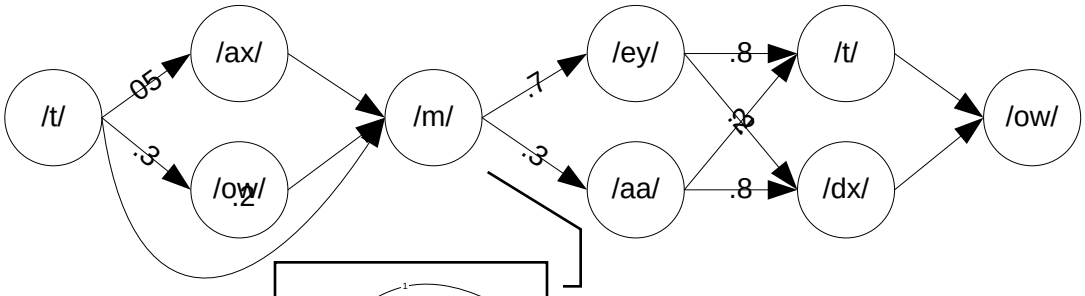
- Probabilistic finite automaton combines models of phones into models of words.
- Each node is a HMM that models a phone.
- Transition probabilities are estimated from annotated corpus.



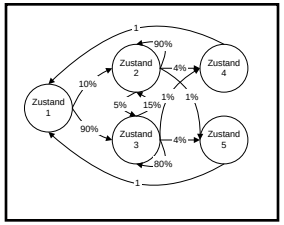
Acoustic Model: Decoding

- Decoding the acoustic model:
  - Find state sequence given the acoustic signal.
  - n-best Viterbi algorithm.
  - Finds n most likely words given the signal.

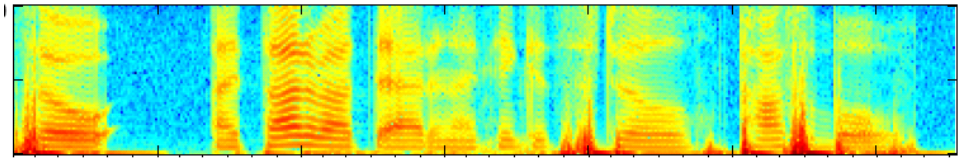
Pronounciation network



Phonetic HMMs



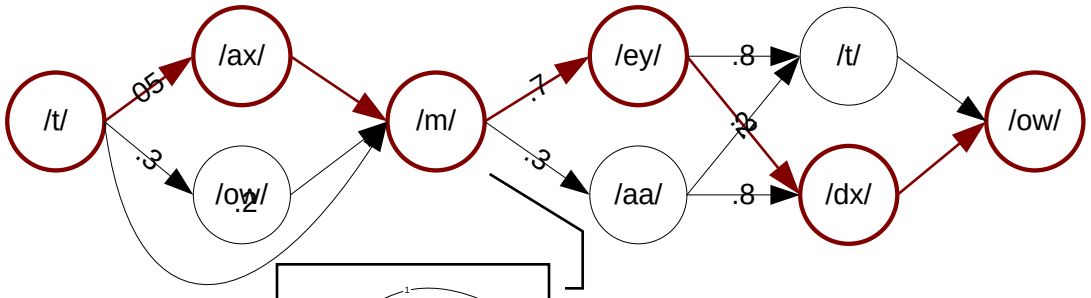
Signal / attributes



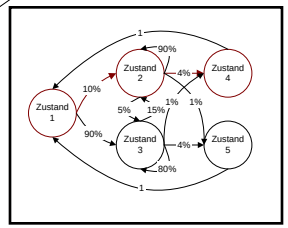
Acoustic Model: Decoding

- Decoding the acoustic model:
  - Find state sequence given the acoustic signal.
  - n-best Viterbi algorithm.
  - Finds n most likely words given the signal.

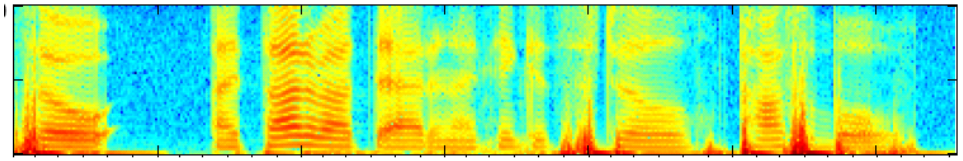
Pronounciation network



Phonetic HMMs



Signal / attributes



Language Model

- Model of $P(\text{sequence of words})$.
- Language models in speech recognition:
  - n-gram models.
  - Probabilistic contect-free grammars (PCFGs).

Joint Decoding Problem

- Maximize acoustic model + language model scores.

argmax: best word sequence.

Given n candidates for each of T words, there are n^T combinations

$$\arg\max_{(\text{words})} P(\text{words} \mid \text{signal})$$

$$= \arg\max_{(\text{words})} P(\underbrace{\text{signal} \mid \text{words}}_{\text{acoustic model}}) \underbrace{P(\text{words})}_{\text{language model}}$$

Bayes

Acoustic model provides n candidate words for each word in sequence

Language model allows to infer probability of each sequence

Joint Decoding Problem

- Beam search algorithm.

1. Input candidates $w_t[1 \dots n]$, beam width k.
2. Let $s_t[1 \dots k] = \emptyset$
3. For $t = 1 \dots T$:
    1. Let candidates $= \{s_{t-1}[i] + w_t[j] | 1 \leq i \leq k, 1 \leq j \leq n\}$
    2. Infer $P(w_t[j])P(s_{t-1}[i] + w_t[j])$ for all candidates and let $s_t[1 \dots k]$ be the highest-scoring candidates.
4. Return $s_T[1]$.

Limitations of HMM-Based Recognition

- Signal preprocessing may steps may lose valuable information from the signal.
- First-order Markov assumption + assumption of mixture of Gaussians for $b_i(O_t)$ probably not satisfied: model error even with infinite data.
- Exact alignment of speech training data to phonemes needed → data collection not easily scalable.

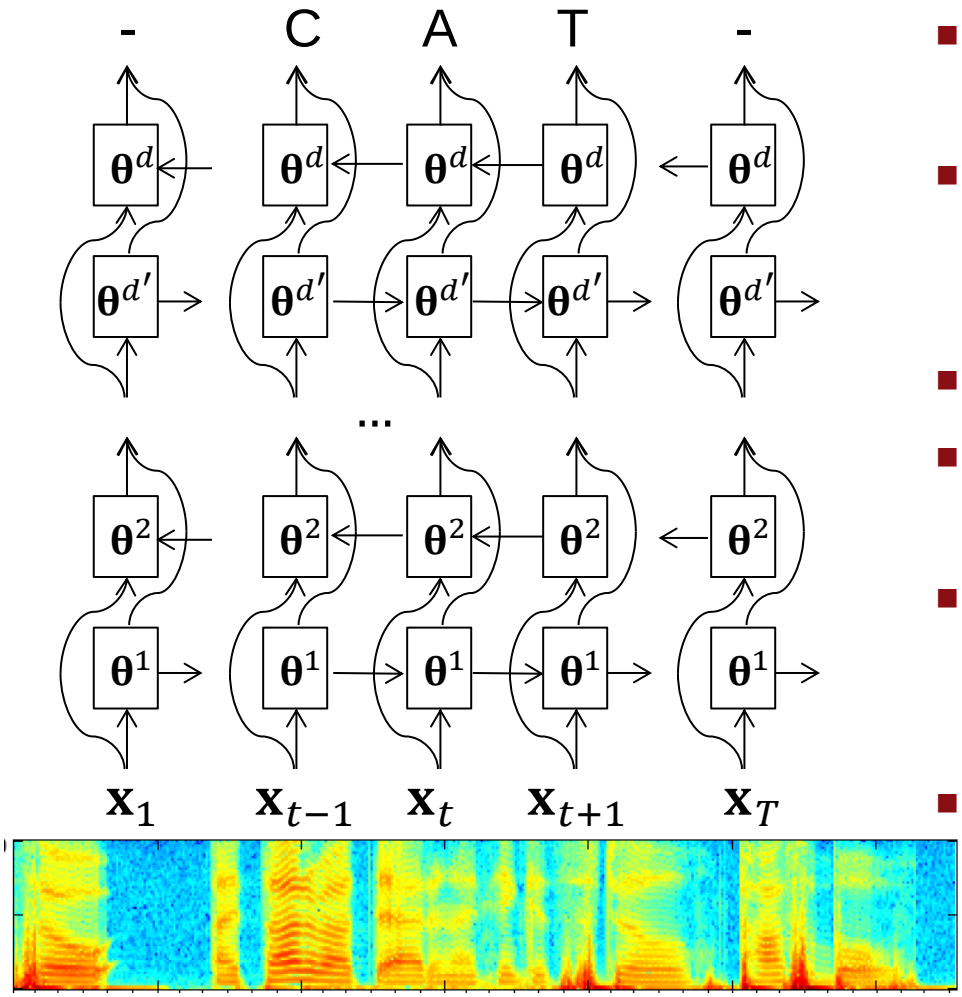
Overview

- Speech recognition usind hidden Markov models.
- Speech recognition using deep recurrent neural networks.
- Voice portals.

Neural Networks for Speech Recognition

- For several years, small parts a complex processing pipeline have been replaced by neural networks.
- Trend: Train single large recurrent neural network for (almost) entire processing pilepeline:
  - From signal processing to word output + separate language model (language model learned from different data source).
  - Or even from signal processing to sentence output.

Neural Networks for Speech Recognition



- Input: Spectrogram (time, frequency, amplitude).
- Output: phones, letters, or words (one-hot coded, softmax output units).
- Output alphabet + “-”
- Deep network of stacked bidirectional LSTM layers.
- Ourput string contains “-” and duplicate phones (or letters, words).
- Postprocessing: beam search with language model.

Neural Networks for Speech Recognition

- Each softmax output unit t gives a probability for each of the possible discrete output values k:
  - $P_{net}(k,t|\mathbf{x}) = \frac{e^{y_t^k}}{\sum_{k'} e^{y_t^{k'}}}$
- Alignment $\mathbf{a}$ is a string of output labels and blanks.
- Network outputs is a distribution over alignments.
  - $P_{net}(\mathbf{a}|\mathbf{x}) = \prod_{t=1}^{T} P_{net}(\mathbf{a}_t, t|\mathbf{x})$

Neural Network Decoding

- Output alphabet: $1 \ldots m$ (e.g., a-z, " ").
- Forward propagation of input $\mathbf{x}_1, \ldots, \mathbf{x}_T \rightarrow$ activation of the T softmax output units with m values.
  - $P_{net}(\mathbf{a}|\mathbf{x}) = \prod_{t=1}^{T} P_{net}(\mathbf{a}_t, t|\mathbf{x})$
- Run beam search: In each step, for each of the k most likely initial sequences $\mathbf{y}[1..t-1]$,
  - Append each possible next output $\mathbf{a}_t$,
  - Calculate posterior probability $p(\mathbf{y}[1..t]|\mathbf{x})$ (this includes the language model, skipping blanks, deduplicating duplicate outputs), and
  - Store the k most likely ones.

Connectionist Temporal Classification

- Example: $\mathbf{y}$ ="CAT", $T = 7$.
- Possible alignments $\mathbf{a}$:
  - "-C-A-T-", "-CAA-TT", "CC-A-TT", …

- Let $B^{-1}(\mathbf{y})$ be the set of all possible alignments.
- $P_{net}(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in B^{-1}(\mathbf{y})} P_{net}(\mathbf{a}|\mathbf{x})$

Connectionist Temporal Classification

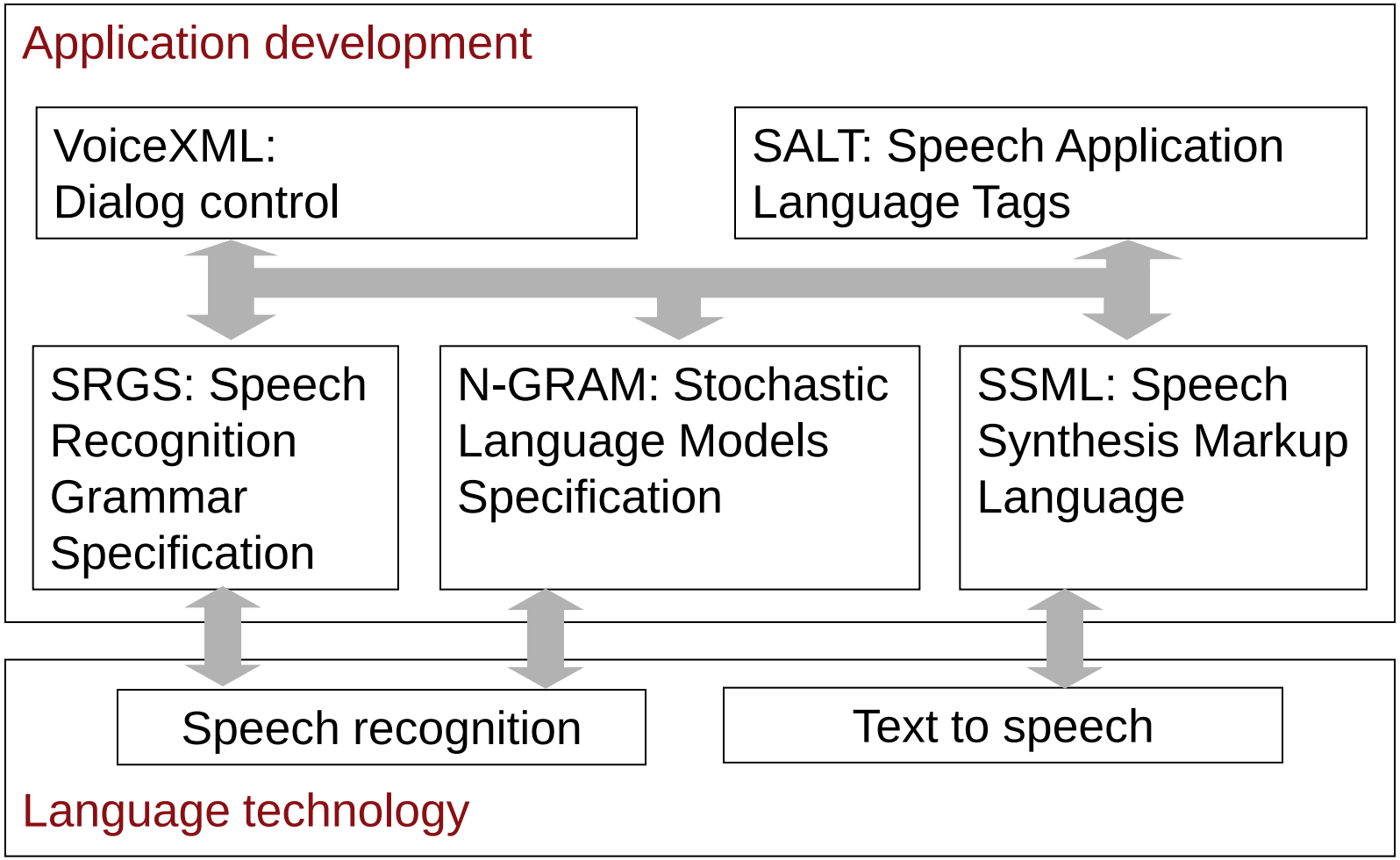
- Training on unaligned pairs $(\mathbf{x}_i, \mathbf{y}_i)$.
- Train network to output any alignment $\mathbf{a} \in B^{-1}(\mathbf{y}_i)$.

- For each example $(\mathbf{x}_i, \mathbf{y}_i)$:
  - Run forward propagation.
  - Find alignment $\mathbf{a}_i^* = \arg \max_{\mathbf{a}_i \in B^{-1}(\mathbf{y}_i)} P_{net}(\mathbf{a}_i | \mathbf{x}_i)$.
  - Run back propagation with target $\mathbf{a}_i^*$.

Overview

- Speech recognition usind hidden Markov models.
- Speech recognition using deep recurrent neural networks.
- Voice portals.

Voice Portals



Speech Recognition Grammar Specification

- Syntax for probabilistic, context-free grammar:
  - Augmented BNF or XML.
  - Speech and DTMF input.
- Basic elements
  - Rule definitions,
  - Rule expansions.

SRGS: Elements

- Rule definition:
  - Associates rule to identifier.

```
<rule id = order>
        [rule expansion]
</rule>
```

SRGS: Elements

- Rule definition:
  - Associates rule to identifier.
- Rule reference:
  - Reference to rule or n-gram.
  - VOID, NULL, GARBAGE.

```
<rule id = order>
   <ruleref uri = #greeting/>
            …
</rule>
```

SRGS: Elements

- Rule definition:
  - Associates rule to identifier.
- Rule reference:
  - Reference to rule or n-gram.
  - VOID, NULL, GARBAGE.

```
<one-of>
        <item>Caipirinha</item>
        <item>Mojito</item>
        <item>Zombie</item>
</one-of>
```

- Alternative:
  - Accepts any of the alternatives.

SRGS: Elements

```
<one-of>
   <item weight=10 >Caipirinha
   </item>
   <item weight=5>Mojito</item>
   <item weight=1>B52</item>
</one-of>
```

- Rule reference:
  - Reference to rule or n-gram.
  - VOID, NULL, GARBAGE.
- Weights:
  - Transformed into probabilities
  - $\rightarrow$ PCFG.
- Alternative:
  - Accepts any of the alternatives.

N-GRAM: Stochastic Language Models

- Syntactic scheme for the representation of
  - Dictionaries,
  - Counts of n-gram tuples.
- Elements
  - <lexicon> dictionary declaration
  - <token> token declaration,
  - <tree> occurrance counter,
  - <interpolation> linear interpolation weights.

SSML: Speech Synthesis Markup

- Markup conventions for all stages of speech generation.

- ACSS: Aural Cascading Style Sheets.
  - Komplex markup definitions,
  - Assognments of speakers to markup tags,
  - Spacial positioning of speakers.

SSML: Elements

- Normalization (expansion of abbreviations, currencies:
  - <say-as>- and <sub> elements.
  - <p>- and <s> elements (paragraph, sentence).
- Conversion text → phoneme:
  - <phoneme> element.
  - Description in the IPA alphabet.
- Prosodic analysis:
  - <emphasis>-, <break>- and <prosody> elements.
- Signal generation:
  - <voice> element, voice selection.
  - Attribute gender, age, variant, name.

VoiceXML

- Goal:
  - Separation of application development and language technology.
- Elements:
  - Dialog control,
  - Speech recognition and generation, DTML.
  - Recording and playing messages.

```
<vxml version="2.0">
<form>
   <block>
      <prompt> Hello world!
      </prompt>
   </block>
</form> </vxml>
```

VoiceXML: Concepts

- Dialogs and subdialogs:
  - Menus: branching points
  - Forms: collect information in fields.
- Events: exception handling.
- Links:
  - Grammar that is active in a certain scope.
  - Triggers event or refers to target URI.
- Procedural elements:
  - <var>, <assign>, <goto>,
  - <if> <else/> </if>.
  - <objekt>: Calls plattform-specific element.

VoiceXML: Generation

- VoiceXML file is typically generated,
  - Just like HTML,
  - For instance using XSLT from XML content

```
<cocktail_menu>
    <cocktail>
        <name>Caipirinha</name>
        <descr>National drink of Brasilia
        </descr>
    </cocktail>
    …
</cocktailkarte>
```

```
<xsl:template match="cocktail_menu">
  Cocktail menu
  <xsl:template match="cocktail">
    <voice gender = male>
        <xsl:value-of select="name"/>
    </voice>
    <p/>
    <voice gender = female>
        <xsl:value-of select="descr"/>
    </voice>
  </xsl:template>
  …
</xsl:template>
```

```
<voice gender=male>Caipirinha
</voice>
<voice gender=female>
  National drink of Brasilia</voice>
```

VoiceXML Elements: Forms

- <form>: Form; <field>: Fillable field.
- Attributes and methods:
  - Identifier to be used for reference.
  - Variable that can be filled with input.
  - Scope: local for dialog or global.
  - Event handling, actions such as call forwarding.
  - Requirements for fields.

VoiceXML Elements: Forms

- Interpretation:
  - Until every <field> is filled,
  - Choose <field>, read <prompt>, wait for input.

```
<form id=order>
    <block>Ready to take your order.</block>
    <field name="drink">
        <prompt>What would you like to drink?
        </prompt>
        <grammar src=„cocktail.grxml“ type=„application/srgs+xml“/>
    </field>
</form>
```

VoiceXML Forms: Mixed Initiative

- <initial>, initial element: first prompt is read at first iteration.
- User can fill one or several fields.
- Interpreter then reads prompt of first field that has not yet been filled.

> Where would you like that delivered?
To August-Bebel-Str. 89,
my name if Tobias Scheffer
> What is your Zip code?

VoiceXML - Example

```
<vxml version="2.0">
<form id="cocktail">
    <field name="drink">
        <prompt> This is the Humboldt cocktail delivery service.
                 What would you like to order, Sir?
        </prompt>
        <grammar xml:lang="en-US" type="application/srgs+xml" version="1.0"
        mode="voice">
            <rule id="options" scope="public">
                <count number="optional">i would like a</count>
                <one-of>
                    <item tag="mai tai">mai tai</item>
                    <item tag="sling">singapore sling</item>
                    <item tag="zombie">zombie</item>
                </one-of>
                <count number="optional">please</count>
            </rule>
        </grammar>
        <catch event="nomatch noinput">
          Sorry, we don't have that. <reprompt/>
       </catch>
       <filled>
        <if cond="drink=='sling'">
          <prompt>
             Sorry, we are out of soda.
          </prompt>
          <exit/>
        <else/>
           <value expr="drink"/>. Good choice.
        </if>
     </filled>
</field>
```