Advanced methods in Automatic Speech Recognition

Dr. Ralf Schlüter, Prof. Dr.-Ing. Hermann Ney

Lehrstuhl für Informatik 6
Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University
D-52056 Aachen, Germany

August 5, 2010
## Schedule

**Course:** Introduction to Automatic Speech Recognition

<table>
<thead>
<tr>
<th>Event</th>
<th>Times</th>
<th>Room</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture</td>
<td>Tue (biweekly)</td>
<td>6124</td>
<td>April 20, 2010</td>
</tr>
<tr>
<td></td>
<td>15:45–17:15h</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>6124</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14:00–15:30h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exercises</td>
<td>Tue (biweekly)</td>
<td>6124</td>
<td>April 27, 2010</td>
</tr>
<tr>
<td></td>
<td>15:45–17:15h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Room 6124: seminar room of the Lehrstuhl für Informatik 6

See course site:

http://www-i6.informatik.rwth-aachen.de/web/Teaching/Lectures/SS10/advasr

for

- news
- downloads (documents, exercise sheets, etc.)
- course information
- contacts
0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Outline

0. Lehrstuhl für Informatik 6
  0.1 Research Topics
  0.2 Projects
  0.3 Courses
  0.4 Textbooks

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Research Topics

Method: Stochastic Modelling

- Modelling dependencies and vague knowledge (contrast: rule-based approach)
- Decision making, in particular in context
- Automatic learning from data/examples

Applications:

Human Language Technology and Pattern Recognition
Applications: Examples

- Speech recognition
  - small vocabulary
  - large vocabulary
- Machine translation
- Natural language processing
  - text/document classification
  - information retrieval
  - parsing and syntactic analysis
- Language understanding and dialog systems
- Image recognition
  - object recognition
  - handwriting recognition
Applications: Examples

- Diagnosis and expert systems
- Other applications:
  - speaker verification and identification
  - fingerprint verification and identification
  - DNA sequence identification
  - gesture recognition
  - lip reading
  - geological analysis
  - high-energy physics: bubble chamber tracks
  - ...

Outline

0. Lehrstuhl für Informatik 6
  0.1 Research Topics
  0.2 Projects
  0.3 Courses
  0.4 Textbooks

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Lehrstuhl für Informatik 6 (i6): Projects

- **ARISE (EU):**
  Automatic Railway Information Systems across Europe
  – Speech Recognition and Language Modelling

- **EuTrans II (EU):** Translation of Spoken Language
  – Speech Recognition and Translation

- **Institut für deutsche Sprache (IdS):**
  – Language Modelling for Newspapers

- **Audio Document Retrieval (NRW):**
  – Speech Recognition and Information Retrieval

- **Verbmobil II (BMBF):** Speech Recognition and Translation for Appointment Scheduling and Traveling Information
  – Speech Recognition
  – Speech Translation
  – Prototype Modules
Projects i6

- **Image Object Recognition (RWTH):**
  - OCR (optical character recognition)
  - Medical Images

- **Advisor (EU):**
  - Speech Recognition for German Broadcast News

- **EGYPT follow-up (NSF):**
  - Basic Algorithms for Statistical Machine Translation

- **Audio Document Retrieval (NRW ?):**
  - German Broadcast News: Recognition and Information Retrieval

- **Bilateral Projects with Companies (including start-ups)**

- **German DFG:**
  - Improved Acoustic Modelling using Structured Models
  - Statistical Methods for Written Language Translation
  - Statistical Modeling for Image Object Recognition
Projects i6

- Coretex (EU):
  - Improving Core Technology for Speech Recognition
  - Applications: Broadcast News in Several Languages

- LC-Star (EU):
  - Lexical and Corpora Resources for Recognition, Translation and Synthesis
  - Prototype system for machine translation of spoken sentences

- TC-Star (EU):
  - Technology and Corpora for Speech to Speech Translation
  - Applications: Broadcast News and Speeches/Lectures

- Transtype-2 (EU):
  - Machine translation of written text
  - Application: interactive machine-aided translation

- PF-Star (EU):
  - Machine translation of spoken dialogues
  - Application: tourism and travelling
Projects i6

- **JUMAS** (EU):
  - Judicial Management by digital libraries Semantics
  - Application: audio and video search of court proceedings

- **LUNA** (EU):
  - spoken Language UNderstanding in multilingual communication systems
  - Application: real-time understanding of spontaneous speech in advanced telecom services

- **GALE** (US-DARPA):
  - Global Autonomous Language Exploitation
  - Application: Information Processing in Multiple Languages

- **QUAERO** [lat.: to search] (OSEO/France)
  - multimedia and multilingual indexing
  - Application: extract information from written texts, speech and music audio, images, and video
Outline

0. Lehrstuhl für Informatik 6
   0.1 Research Topics
   0.2 Projects
   0.3 Courses
   0.4 Textbooks

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Courses

- Introductory lectures (L3/4) with exercises (E2) for Bachelor, Master, and Diploma students:
  - ASR: (Introduction to) Automatic Speech Recognition
  - PRN: (Introduction to) Pattern Recognition and Neural Networks
  - NLP: (Introduction to) Natural Language Processing

- Advanced lectures (L3) with exercises (E1/2) for Master and Doctoral students:
  - advASR: Advanced Automatic Speech Recognition
  - advPRN: Advanced Pattern Recognition
  - advNLP: Advanced Natural Language Processing

- Further Lectures (L2) with exercises (E1):
  - MIP: Medical Image Processing
    ('Ringvorlesung', each WS)
Courses (ctd.)

- Seminars:
  - Bachelor Degree (SS, Block)
  - Diplom Degree (SS, Block)
  - Doctor Degree (WS+SS)
- Laboratory Courses (WS, Block)
- Study Groups (WS+SS: speech, language, image)

New course cycles:

<table>
<thead>
<tr>
<th>year</th>
<th>term</th>
<th>lectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/10</td>
<td>WS</td>
<td>PRNN (L4/3,E2)</td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td>NLP (L4/3,E2)</td>
</tr>
<tr>
<td>10/11</td>
<td>WS</td>
<td>PRNN (L4/3,E2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASR (L4/3,E2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>advASR (L3,E1)</td>
</tr>
</tbody>
</table>
area of specialization (Vertiefungsgebiet) i6 with the topics:

- Automatic Speech Recognition (ASR)
- Pattern Recognition and Neural Networks (PRNN)
- Natural Language Processing (NLP)
- ...

select 12 hours (SWS) out of i6 lectures
practical computer science (Prakt. Informatik) (3 areas): recommendation: 12 hours (SWS) out of two L4 from: ASR, PRNN, NLP
one L4 from i6-external lectures:
  - data bases
  - artificial intelligence
  - ... additional alternatives: on demand
Examinations i6

- Master in Informatik, Media Informatics or Software Systems Engineering:
  credit system: oral exam after each course/at end of lecture period

- ERASMUS students of Computer Science:
  oral exam/colloquium for graded certificate at end of lecture period

Note: consult Prof. Ney before June 2010 for exam dates, and before registering for the exam.
Outline

0. Lehrstuhl für Informatik 6
  0.1 Research Topics
  0.2 Projects
  0.3 Courses
  0.4 Textbooks

1. Large Vocabulary Speech Recognition
2. Search using Lexical Pronunciation Tree
3. Across-Word Models
4. Word graphs and Applications
5. Time Conditioned Search
6. Normalization and Adaptation
7. Discriminative Training
Textbooks: Topics i6

Textbooks on Speech Recognition:

► emphasis on signal processing and small-vocabulary recognition:

► emphasis on large vocabulary and language modelling:

► introduction to both speech and language:
  D. Jurafsky, J. H. Martin: Speech and Language Processing.

► advanced topics:
  R. De Mori: Spoken Dialogues with Computers.
Textbooks on Signal Processing:


Further reading on Signal Processing:

Textbooks on Natural Language Processing (statistical/corpus-based):

▶ introduction to both speech and language:

▶ emphasis on statistical methods for written language:

▶ related field: artificial intelligence:
Textbooks: Topics i6

Textbooks on Statistical Learning (Pattern Recognition, Neural Networks, Data Mining, ...):

- best introduction (including modern concepts):

- emphasis on statistical concepts:

- emphasis on modern statistical concepts:

- emphasis on theory and principles:
Textbooks on mathematical methods (vector spaces and matrices, statistics, optimization methods, ...):

▶ best overall summary:

▶ introduction to modern statistics:

▶ good overview of numerical algorithms and implementations:
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Overview: Architecture

Starting point: Bayes decision rule

- results in a minimum number of recognition errors (under certain conditions)
- more details: see lecture Pattern Recognition and Neural Networks
Speech Recognition: Bayes’ Decision Rule

Speech Input

Acoustic Analysis

Global Search:
maximize
Pr(w₁...wₙ) \cdot Pr(x₁...xₜ | w₁...wₙ)

over w₁...wₙ

Recognized Word Sequence

Pr(x₁...xₜ | w₁...wₙ)

Phoneme Inventory

Pronunciation Lexicon

Language Model

Schlüter/Ney: Advanced ASR

August 5, 2010
Speech Recognizer: Sources of Errors

Why does a recognition system make errors?
Reasons from the viewpoint of Bayes’ decision rule:

▶ incorrect acoustic model:
  – poor acoustic analysis
  – poor phoneme models
  – poor pronunciation model

▶ incorrect language model

▶ incorrect search procedure:
  the maximum is not found

▶ decision rule: discrepancy between evaluation measure (word error rate) and decision rule (minimizes sentence error rate)
Importance of higher level knowledge and its integration in the search process. Test results on the Wall Street Journal 5k task:

<table>
<thead>
<tr>
<th>knowledge sources used</th>
<th>perplexity PP</th>
<th>phoneme error rate [%]</th>
<th>word error rate [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>unconstrained</td>
<td>–</td>
<td>36.3</td>
<td>–</td>
</tr>
<tr>
<td>phoneme recognition</td>
<td>5000</td>
<td>13.9</td>
<td>40.0</td>
</tr>
<tr>
<td>+ pronunciation lexicon</td>
<td>746</td>
<td>8.4</td>
<td>22.9</td>
</tr>
<tr>
<td>bigram</td>
<td>107</td>
<td>2.8</td>
<td>6.9</td>
</tr>
<tr>
<td>trigram</td>
<td>56</td>
<td>1.9</td>
<td>4.5</td>
</tr>
</tbody>
</table>
## Effect of Knowledge Sources

Example from the Wall Street Journal 5k task:

<table>
<thead>
<tr>
<th>LM</th>
<th>recognized</th>
<th>errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>no lexicon</td>
<td>k t k t dh ey d v eh d ey n ey ih z n un k oh sh h ee ey d ih ng n dh uh dh s ey l uh f s ur n d h aa s dh aa t s UH b dh uh b r oh k r ih j y ooh n ih t p p</td>
<td>28</td>
</tr>
<tr>
<td>0-gram</td>
<td>h ih t s eh n uh t ur z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw n t uh b r oh k ur ih j y ooh n ih t s HIT SENATORS — — NEGOTIATING — SALE — CERTAIN ASSETS ONTO — BROKERAGE UNIT'S</td>
<td>11</td>
</tr>
<tr>
<td>1-gram</td>
<td>ih t s s eh n ih t ih z n ih g oh sh ee ey t ih ng — — s ey l — — s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t ITS SENATE — IS NEGOTIATING — SALE — CERTAIN ASSETS OF THE BROKERAGE UNIT</td>
<td>6</td>
</tr>
<tr>
<td>2-gram</td>
<td>ih t s eh d ih t ih z n ih g oh sh ee ey t ih ng dh uh s ey l aw v s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t IT SAID IT IS NEGOTIATING THE SALE OF CERTAIN ASSETS OF THE BROKERAGE UNIT</td>
<td>0</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
For large vocabularies, it is prohibitive to use whole-word models for each word of the vocabulary:

- There are not enough training samples for each word.
- The memory requirements increase linearly with the number of words (today: no real problem).

Solution: create word models by concatenating sub-word units, such as phonemes, context dependent phonemes, demi-syllables, syllables, ... 

Advantages:

- Training data is shared between words.
- Words not seen in training (i.e. without training examples) can be recognized by using a pronunciation lexicon.
Zipf’s Law

The problem of sparse data is related to “Zipf’s law”:

The frequency $N(w)$ of a word $w$ is (approximately) inversely proportional to some power $\gamma$ of its rank $r(w)$.

$$N(w) = \text{const} \cdot r(w)^{-\gamma}$$

Example from the Verbmbobil corpus:

<table>
<thead>
<tr>
<th>rank</th>
<th>word</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ich</td>
<td>18648</td>
</tr>
<tr>
<td>2</td>
<td>ja</td>
<td>16613</td>
</tr>
<tr>
<td>3</td>
<td>das</td>
<td>14288</td>
</tr>
<tr>
<td>4</td>
<td>wir</td>
<td>13532</td>
</tr>
<tr>
<td>4440</td>
<td>Abendtermine</td>
<td>1</td>
</tr>
<tr>
<td>4441</td>
<td>Aberglaubens</td>
<td>1</td>
</tr>
<tr>
<td>10000</td>
<td>zwingend</td>
<td>0</td>
</tr>
</tbody>
</table>
Phonetic (Phonemic) Models

Distinguish the various levels:
– acoustic realization: acoustic signal
– class of equivalent sounds: phone (allophone, triphone)
– (more) abstract level: phoneme

Speech sounds may be categorized according to different ’features’:

for consonants
  ▶ voiced / voiceless
  ▶ manner of articulation
    stop, nasal, fricative, approximant
  ▶ place of articulation
    labial, dental, alveolar, palatal, velar, glottal

for vowels
  ▶ position of tongue:
    high/low, front/back
  ▶ rounded or not
Subword Units

speech \iff temporal sequence of sounds
\downarrow
acoustic signal \iff temporal sequence of acoustic vectors,
(acoustic realization of the sounds)

Model of speech production:

- Every sound has a program for the movements of the vocal tract.
- Movements of individual sounds merge into one continuous sequence of movements.
- Ideal positioning of the vocal tract is only approximated (depending on the amount of coarticulation).
  \Rightarrow the real acoustic signal differs from the 'ideal' signal.
The Vocal Tract

Drawing by Laszlo Kubinyi ©Scientific American 1977
Subword Units

Criteria for sound classification

- type of articulation (fricative, plosive)
- location of articulation ([p]: labial, [s]: dental)
- consonants and vowels
- voiced and unvoiced sounds
- stationary and non stationary sounds
  (vowels vs. diphthongs, plosives)

Perception of sounds:

- loudness
- tone (smoothed spectrum = formant spectrum)
- unvoiced, voiced (fundamental frequency, pitch)
The pronunciation of a word is usually described in a less detailed way using phonemes.

A phoneme is an abstraction over different phonetic realizations.

Two sounds correspond to different phonemes if they can occur in the same context and distinguish different words.

The phoneme inventory of a language can be inferred from “minimal pairs”.

A minimal pair is a pair of words whose phonetic transcriptions have an edit distance of one.
Phonemes

Examples of minimal pairs for German

<table>
<thead>
<tr>
<th>Vowels:</th>
<th>Consonants:</th>
</tr>
</thead>
<tbody>
<tr>
<td>i:  / o:</td>
<td>p  / b</td>
</tr>
<tr>
<td>I  / E</td>
<td>t  / m</td>
</tr>
<tr>
<td>e:  / Y</td>
<td>k  / ts</td>
</tr>
<tr>
<td>a:  / a</td>
<td>f  / v</td>
</tr>
<tr>
<td>Y  / 9</td>
<td>s  / S</td>
</tr>
<tr>
<td>o:  / aU</td>
<td>s  / z</td>
</tr>
<tr>
<td>e:  / Ù:</td>
<td>l  / –</td>
</tr>
</tbody>
</table>
Characteristics of the phoneme set:

- The phoneme set is language specific. Examples:
  
  Chinese  
  \[ l \] – \[ r \]  
  one phoneme

  Arabic  
  \[ k_{i} \] – \[ k_{u} \]  
  different phonemes

- Humans are trained to distinguish sounds of specific languages.

- The acoustic realizations of phonemes are context dependent (coarticulation):
  
  - static dependencies on surrounding phonemes
  - dynamic dependency:
    temporal overlap of the articulation of subsequent phonemes
### Phoneme System for German in SAMPA notation

#### Consonants
- **Plosives**
  - p  `Pein`  p  a l  n
  - b  `Bein`  b  a l  n
  - t  `Teich`  t  a l  C
  - d  `Deich`  d  a l  C
  - k  `Kunst`  k  U  n  s  t
  - g  `Gunst`  g  U  n  s  t

#### Fricatives
- f  `fast`  f  a s  t
- v  `was`  v  a s
- s  `Tasse`  t  a  s  @
- z  `Hase`  h  a:  z  @
- S  `waschen`  v  a  S  @  n
- Z  `Genie`  Z  e  n  i:
- C  `sicher`  z  i  C  6
- j  `Jahr`  j  a:  6
- x  `Buch`  b  u:  x
- h  `Hand`  h  a  n  t

#### Consonants
- **Sonorants**
  - m  `mein`  m  a l  n
  - n  `nein`  n  a l  n
  - N  `Ding`  d  l  N
  - l  `Leim`  l  a l  m
  - R  `Reim`  R  a l  m

#### Vowels
- **“checked” (short) vowels**
  - l  `Sitz`  z  l  t  s
  - E  `Gesetz`  g  @  z  E  t  s
  - a  `Satz`  z  a  t  s
  - O  `Trotz`  t  r  O  t  s
  - U  `Schutz`  S  U  t  s
  - Y  `hübsch`  h  Y  p  S
  - 9  `plötzlich`  p  l  9  t  z  l  l  C

- **“free” (long) vowels**
  - i:  `Lied`  l  i:  t
  - e:  `Beet`  b  e:  t
  - E:  `spät`  S  p  E:  t
  - a:  `Tat`  t  a:  t
  - o:  `rot`  r  o:  t
  - u:  `Blut`  b  l  u:  t
  - y:  `süß`  z  y:  s
  - 2:  `blöd`  b  l  2:  t

- **Diphthongs**
  - al  `Eis`  a l  s
  - aU  `Haus`  h  aU  s
  - OY  `Kreuz`  k  r  OY  t  s

- **“schwa” vowels**
  - @  `bitte`  b  l  t  @
  - 6  `besser`  b  E  s  6
Phonemes

Function of the phonemes:

- acoustic signal: continuous, infinite number of realizations
  \[\uparrow (1:\infty)\]

- (allo-)phones: discrete sounds, approx. 40,000
  \[\uparrow (1:1000)\]

- phonemes: discrete, alphabet: 40 – 60, depending on the language
  \[\uparrow (1:1)\]

- words of the language: several 100,000 words
- pronunciation and meaning
Context Dependent Subword Units

The acoustic realization of phonemes is context dependent.

Context dependent modelling is more accurate:

- **Diphones**

  

  \[
  \begin{align*}
  &A & B & C & D & E \\
  &\#A | AB | BC | CD | DE | E\#
  \end{align*}
  \]

- **Syllables**: group of phonemes, standard form consonant–vowel–consonant, about 20000 syllables for German.

- **Demi syllables** (syllables split at the vowel)

- **Consonant clusters**

\[
\begin{align*}
\text{energy} & \uparrow \\
\text{time} & \uparrow \\
\text{vowel} & \\
\text{consonant} & \quad \text{consonant}
\end{align*}
\]
Subword Units

Example: possible subword units for German

<table>
<thead>
<tr>
<th>Subword Unit</th>
<th>Number (approx.)</th>
<th>Representation of the acoustic signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>phonemes</td>
<td>50</td>
<td>inaccurate</td>
</tr>
<tr>
<td>consonant clusters</td>
<td>250</td>
<td>...</td>
</tr>
<tr>
<td>and vowels</td>
<td>2500</td>
<td>...</td>
</tr>
<tr>
<td>diphones</td>
<td>2500</td>
<td>...</td>
</tr>
<tr>
<td>demi-syllables</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>syllables</td>
<td>20 000</td>
<td>accurate</td>
</tr>
</tbody>
</table>

note terminology: consonant cluster = consonant sequence
Subword Units

Practical reasons for using subword units in speech recognition:

- Not enough training data for whole word models.
- More observations for subword units (better training).
- Vocabulary can be extended without new acoustic training.
  Specifying the corresponding subword units is sufficient.

Important issues when using subword units:

- Define and specify subword units.
- Map the continuous signal to the discrete sequence of units
  i.e. specify the units and the pronunciation lexicon.
- Train the subword units.
- Use the subword units and the pronunciation lexicon for recognition.
HMMs for phonemes

Layers of the acoustic modelling

words: THIS BOOK IS GOOD

phonemes: th i s b u h k i z g u h d

subphonemes: ... b_{cl} b_{rel} u_{on} u_{h} u_{off} k_{cl} k_{rel} ...

acoustic vectors: ... ...

speech signal: ... ...

Speech can be modeled on any of these layers
Different HMM topologies for phonemes can be used, define

- number of states and
- allowed transitions.

Usually three sub phonemes are used:

Begin – Middle – End

3 state model

or
HMMs for phonemes

“IBM model”

Properties:

- Transition assigned emissions: the emission probability distributions are assigned to the transitions (not to the states).
- Number of possible paths is restricted for short vector sequences:
  1: $B$
  2: $BM$
  3: $BME$
  4: $BMME$
  5: $BBMME$, $BMMEE$, $BMMME$
  6: ...
HMMs for phonemes

- 6 state model

![Diagram of 6 state model for phonemes]
A pronunciation lexicon with phonetic transcriptions is required when using subword units. Usually phonemes are used as subword units.

Example: English digits

<table>
<thead>
<tr>
<th>word</th>
<th>phonemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>Z IH R OW</td>
</tr>
<tr>
<td>One</td>
<td>W AH N</td>
</tr>
<tr>
<td>Two</td>
<td>T UW</td>
</tr>
<tr>
<td>Three</td>
<td>TH R IY</td>
</tr>
<tr>
<td>Four</td>
<td>F OW R</td>
</tr>
<tr>
<td>Five</td>
<td>F AY V</td>
</tr>
<tr>
<td>Six</td>
<td>S IH K S</td>
</tr>
<tr>
<td>Seven</td>
<td>S EH V AX N</td>
</tr>
<tr>
<td>Eight</td>
<td>EY T</td>
</tr>
<tr>
<td>Nine</td>
<td>N AY N</td>
</tr>
<tr>
<td>Oh</td>
<td>OW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>phoneme</th>
<th>number of occurances</th>
</tr>
</thead>
<tbody>
<tr>
<td>AH</td>
<td>1</td>
</tr>
<tr>
<td>AX</td>
<td>1</td>
</tr>
<tr>
<td>AY</td>
<td>2</td>
</tr>
<tr>
<td>AY</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>EH</td>
<td>1</td>
</tr>
<tr>
<td>EY</td>
<td>1</td>
</tr>
<tr>
<td>IH</td>
<td>2</td>
</tr>
<tr>
<td>IY</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>4</td>
</tr>
<tr>
<td>OW</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>3</td>
</tr>
<tr>
<td>SR</td>
<td>2</td>
</tr>
<tr>
<td>TH</td>
<td>2</td>
</tr>
<tr>
<td>UW</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>1</td>
</tr>
</tbody>
</table>
Pronunciation Lexicon

Context dependencies:

▸ Context independent (real) phoneme models:
  International phonetic alphabet defines 74 phonemes for English.
  In practical applications about 40–50 phonemes are used typically.
  
  \[
  z \quad e \quad r \quad o \\
  Z \quad IH \quad R \quad OW
  \]

▸ Context dependent phoneme models:
  Coarticulation is considered:
  
  \[
  z \quad e \quad r \quad o \\
  \#Z_{IH} \quad zIHR \quad IHROW \quad ROW\#
  \]

▸ “Diphone” \(_A B, B_C\): context dependent phoneme in diphone context

▸ “Triphone” \(_A B_C\): context dependent phoneme in triphone context
Pronunciation Lexicon

Terminology:

- context independent phonemes ('monophones', more or less the 'real' phonemes as defined in linguistics)
- phonemes in (left or right) diphone context ('diphones')
- phonemes in triphone context ('triphones')
- phonemes in word context ('wordphones')

The context dependency only determines the labels for emission probabilities which have to be specified for each state of a phoneme model.

The emission probabilities can be trained independent from the phonetic context.

The sequence of states is used for recognition:

- word: sequence of phoneme models,
- phoneme model: sequence of HMM states

⇒ word: sequence of HMM states.
Training Phoneme Models

- The sequence of HMM state indices for a word depend on the phoneme sequence.
- The training procedure corresponds to the one used for word models:
  - Time alignment: assign a state to every acoustic vector.
  - Parameter estimation:
    - Collect all observations for every mixture $m$ of every phoneme model based on the time alignment.
    - Estimate the model parameters for all densities $l$ of the mixture $m$:
      - Reference or prototype vector $\mu_{lm}$
      - Pooled variance vector $\sigma^2_m$
      - Mixture weight $p(l|m)$
Training Phoneme Models

Practical considerations:
All possible triphones $50^3 = 125000$ are too many!

- Combine monophones, diphones and triphones: only use diphones and triphones that occur more than e.g. 100 times in the training data.

- Generalized context dependent phoneme models: use phoneme classes (nasals, fricatives, vowels, stop consonants, ...)
  \[ g(A)B g(C) \text{ instead of } _A B_C, \quad g(X) \text{ phoneme class } X \text{ belongs to} \]

- Parameter tying: use clustering or Classification And Regression Trees (CART) to tie “similar” phonemes

⇒ a few thousand models that are actually used
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Phonetic Decision Trees

Motivation

Classification and Regression Trees (CART)

▶ Used in the acoustic modelling of phonemes
▶ 50 phonemes ⇒ $50^3 = 125000$ possible phonemes in triphone context ("triphones")
▶ Problem:
  ▶ too many triphones to be trained reliably
  ▶ many triphones are not seen in training
  ▶ considering across-word contexts, this effect increases
▶ Solution:
  ▶ tie parameters of similar triphones
  ▶ a decision tree determines similarity
A phoneme $X$ has 2500 possible triphone contexts $aX_b$. Phonetic decision tree for a phoneme $X$:

A path through the decision tree is defined by the answers to phonetic questions $Q_0, Q_1, Q_2, \ldots$

e.g.: - “is the left context a fricative?”
- “is the right context a plosive?”
Example
Properties of the tree:

- Every leaf of the tree stands for a generalized phonetic context and has a corresponding HMM emission probability.
- An adequate generalization for triphones not seen in training can be expected.
Motivation

General application for CART:

Given the two variables

\[ c = \text{class index} \]

\[ x \in \mathbb{R}^D, \text{or discrete observation} \]

model conditional probability

\[ p(c|x) \]

\[ \sum_c p(c|x) = 1 \]

“Classification tree” vs. “estimation tree”:
an estimation tree models the conditional probability without classification.
Training Principle

Given the training data

\[ [x_n, y_n], \quad n = 1, \ldots, N; \]

with \( x \rightarrow \) independent variable

\( y \rightarrow \) dependent variable

two subsets of \( \{x\} \) are considered

\( t, t_L, t_R \subset \{x\} \)

Define a tree by binary splitting of a node or subtree:

\[ t = t_L \cup t_R, \quad t_L \cap t_R = \emptyset. \]
Training Principle

Define:

- A “score” \( g(y_n|t) \) for every observation \((x_n, y_n)\) with \(x_n \in t\);
- A score for the node \( t \):

\[
G(t) := \sum_{n: x_n \in t} g(y_n|t).
\]

The score function \( g(y_n|t) \) shall be additive.

Note the change in the score when splitting \( t \) in the subsets \( t_L \) and \( t_R \):

\[
\Delta G(t_L|t) = G(t) - G(t_L) - G(t_R)
\]

best split \( t_L \) for given \( t \):

\[
\max_{t_L} \Delta G(t_L|t)
\]
Training Principle

Use the log-likelihood (log-probability) criterion for $G(t)$. ($\theta$ represents the parameter of the distribution):

$$g(y_n|t) := \log p_\theta(y_n|t)$$

$$G(t) := \max_\theta \sum_{n:x_n \in t} \log p_\theta(y_n|t)$$

$G(t_L)$, $G(t_R)$ correspondingly.

Optimization:

- Learn the best parameters $\hat{\theta}$ for a hypothetic split $t_L$ at a node $t$.
- Choose optimal split.

Thus:

$$\hat{\theta} = \hat{\theta} \{(x_n, y_n) : x_n \in t_L; n = 1, \ldots, N\}$$

$$G(t_L) = \sum_{n:x_n \in t_L} \log p_{\hat{\theta}}(y_n|t_L)$$
Training Principle: Discrete Observations

with the counts: $N(t, y), N(t)$

$$\Rightarrow \hat{\theta} = \hat{p}(y|t) = \frac{N(t, y)}{N(t)}$$

$$\sum_{y} \log p(y|t) = \sum_{y} N(t, y) \cdot \log p(y|t) - \lambda$$

$$\frac{\partial}{\partial \lambda} \left[ \sum_{y} p(y|t) N(t, y) - 1 \right] = 0$$

Then:

Then:

with discrete values:

the parameters $\theta$ are the distribution $p(y|t)$ itself

(non parametric model)

$\sum_{y} \log p(y|t) = \sum_{y} N(t, y) \cdot \log p(y|t) - \lambda$
Training Principle: Discrete Observations

For the optimum:

\[
G(t) = \sum_{n:x_n \in t} \log p_{\hat{\theta}}(y_n | t)
\]

\[
= \sum_{n:x_n \in t} \log \frac{N(t, y)}{N(t)}
\]

\[
= \sum_y N(t, y) \cdot \log \frac{N(t, y)}{N(t)}
\]

\[
= N(t) \sum_y \hat{p}(y | t) \cdot \log \hat{p}(y | t)
\]

entropy
Training Principle: Continuous Observations

With continuous values, especially Gaussian distribution:

\[ p_\theta(y|t) = \mathcal{N}(y|\mu_t, \Sigma_t) \]

\[
\mathcal{N}(y|\mu_t, \Sigma_t) = \frac{1}{\sqrt{\text{det}(2\pi \Sigma_t)}} \cdot \exp \left[ -\frac{1}{2} (y - \mu_t)^T \Sigma_t^{-1} (y - \mu_t) \right]
\]

\[ G(t) = \sum_{n:x_n \in t} \log \mathcal{N}(y_n|\hat{\mu}_t, \hat{\Sigma}_t) \]

\[ = -\frac{N(t)}{2} \log \det \left[ 2\pi \hat{\Sigma}_t \right] - \frac{1}{2} \sum_{n:x_n \in t} (y_n - \hat{\mu}_t)^T \hat{\Sigma}_t^{-1} (y_n - \hat{\mu}_t) \]

with

\[ N(t) := \sum_{n:x_n \in t} 1 \]
Training Principle: Continuous Observations

Maximum-likelihood estimation for $\hat{\mu}_t$ and $\hat{\Sigma}_t$:

\[
\hat{\mu}_t = \frac{1}{N(t)} \sum_{n : x_n \in t} y_n
\]

\[
\hat{\Sigma}_t = \frac{1}{N(t)} \sum_{n : x_n \in t} (y_n - \hat{\mu}_t)(y_n - \hat{\mu}_t)^T
\]
Training Principle: Continuous Observations

Using a diagonal covariance matrix:

\[
\hat{\Sigma}_t = \begin{bmatrix}
\hat{\sigma}_{t1}^2 & 0 \\
0 & \hat{\sigma}_{t2}^2 \\
& \ddots \\
0 & \hat{\sigma}_{tD}^2
\end{bmatrix}
\]

\[
\hat{\sigma}_{td}^2 = \frac{1}{N(t)} \sum_{n:x_n \in t} (y_{nd} - \hat{\mu}_{td})^2
\]

\[
\sum_{n:x_n \in t} (y_n - \hat{\mu}_t)^T \hat{\Sigma}_t^{-1} (y_n - \hat{\mu}_t) = \sum_{n:x_n \in t} \sum_d \left( \frac{y_{nd} - \hat{\mu}_{td}}{\hat{\sigma}_{td}} \right)^2 = \sum_d \frac{1}{\hat{\sigma}_{td}^2} \cdot \sum_{n:x_n \in t} (y_{nd} - \hat{\mu}_{td})^2 = N(t) \cdot D
\]
Training Principle: Continuous Observations

General case, full covariance matrix (the index \( t \) of \( \hat{\mu}_t \) and \( \hat{\Sigma}_t \) is dropped here for simplification)

\[
\begin{align*}
z_n &:= y_n - \hat{\mu} \\
\hat{\Sigma} &:= \frac{1}{N(t)} \sum_{n: x_n \in t} z_n \cdot z_n^T \\
\sum_{n: x_n \in t} z_n^T \hat{\Sigma}^{-1} z_n &\quad = \quad \sum_n \sum_{i=1}^{D} \sum_{j=1}^{D} z_{ni} \left( \hat{\Sigma}^{-1} \right)_{ij} z_{nj} \\
&\quad = \quad \sum_{ij} \left[ \sum_{n: x_n \in t} z_{ni} z_{nj} \right] \left( \hat{\Sigma}^{-1} \right)_{ij} \\
&\quad = \quad N(t) \cdot \sum_{ij} \hat{\Sigma}_{ij} \left( \hat{\Sigma}^{-1} \right)_{ij} \\
&\quad = \quad N(t) \cdot \sum_j \sum_i \hat{\Sigma}_{ji} \left( \hat{\Sigma}^{-1} \right)_{ij} \\
&\quad = \quad N(t) \cdot \sum_{j=1}^{D} \delta_{jj} \quad = \quad N(t) \cdot D
\end{align*}
\]
Training Principle: Continuous Observations

Thus:

\[
G(t) = \sum_{n: x_n \in t} \log \mathcal{N}(y_n | \hat{\mu}_t, \hat{\Sigma}_t)
\]

\[
= -\frac{N(t)}{2} \log \det \left[ 2\pi \hat{\Sigma}_t \right] - \frac{1}{2} \sum_{n: x_n \in t} (y_n - \hat{\mu}_t)^T \hat{\Sigma}_t^{-1} (y_n - \hat{\mu}_t)
\]

\[
= -\frac{N(t)}{2} \log \det \left[ 2\pi \hat{\Sigma}_t \right] - \frac{N(t)}{2} D
\]

\[
= -\frac{N(t)}{2} \log \det \left( 2\pi \hat{\Sigma}_t \right)
\]

\[
= -\frac{N(t)}{2} \left[ D \log(2\pi) + D + \log(\det \hat{\Sigma}_t) \right]
\]
Training Principle: Continuous Observations

The improvement of the log-likelihood score by splitting $t = t_L \cup t_R; t_L \cap t_R = \emptyset$ is:

$$
\Delta G(t) = G(t) - G(t_L) - G(t_R)
$$

$$
= \ldots
$$

$$
= \frac{N(t_L)}{2} \log \left[ \frac{\det \hat{\Sigma}_{t_L}}{\det \hat{\Sigma}_t} \right] + \frac{N(t_R)}{2} \log \left[ \frac{\det \hat{\Sigma}_{t_R}}{\det \hat{\Sigma}_t} \right]
$$

For a diagonal covariance matrix:

$$
\log \det \hat{\Sigma}_t = \log \prod_d \hat{\sigma}_{td}^2
$$

$$
= \sum_{d=1}^{D} \log \hat{\sigma}_{td}^2
$$
Leaving One Out

So far without leaving one out:

$$\max_{\theta} \sum_{n} \log p_{\theta}(y_n) \Rightarrow \hat{\theta} = \hat{\theta}(y_1^N)$$

The optimal value is substituted in the log likelihood:

$$\sum_{n} \log p_{\hat{\theta}(y_1^N)}(y_n),$$

so every observation is considered twice:

1. to determine $\hat{\theta}$

2. to determine how good the model $p_{\hat{\theta}}(y)$ explains the observations $y_1, \ldots, y_N$.

$\Rightarrow$ score evaluation is too optimistic.
Leaving One Out

Leaving one out:

Idea: take $y_n$ out of the training set when evaluating $p_{\hat{\theta}}(y_n)$

Use $\hat{\theta}(y_1^N \setminus y_n)$ instead of $\hat{\theta}(y_1^N)$ for the leaving one out score evaluation:

$$\sum_n \log p_{\hat{\theta}(y_1^N \setminus y_n)}(y_n)$$

For Gaussian distributions the calculation of $\hat{\theta}(y_1^N \setminus y_n)$ is easy:

$$\hat{\mu}_n := \frac{1}{N - 1} \sum_{m \neq n} y_m$$

$$\hat{\Sigma}_n := \frac{1}{N - 1} \sum_{m \neq n} (y_m - \hat{\mu}_n)(y_m - \hat{\mu}_n)^t$$
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Language Modelling

Goal:
model syntax and semantics of natural language (spoken or written)
needed in automatic systems that process speech (= spoken language) or language (= written language):

- speech recognition
- speech and text translation
- (spoken and written) language understanding
- spoken dialog systems
- text summarization
- ...

Schlüter/Ney: Advanced ASR  75  August 5, 2010
Finite state networks for digit strings:

Syntactical constraints can be expressed with formal grammars, here represented as networks

- **String of three digits**

  ![Diagram of a string of three digits]

  $V \in \{\text{zero, one, two, three, \ldots, nine}\}$

- **String with even number of digits**

  ![Diagram of a string with even number of digits]

  $V \in \{\text{zero, one, two, three, \ldots, nine}\}$

- **Unconstrained digit string**

  ![Diagram of an unconstrained digit string]

  $V \in \{\text{zero, one, two, three, \ldots, nine}\}$
Silence Model

Allow silence between the words:

- **String of three digits**

```
    Sil  V  Sil  V  Sil  V  Sil
   1    2    3    4
V \in \{0, 1, 2, 3, \ldots, 9\}
```

- **String with even number of digits**

```
    Sil  V  Sil  V  Sil
   1    2    3
V \in \{0, 1, 2, 3, \ldots, 9\}
```

- **Unconstrained digit string**

```
V \mid Sil
1
V \in \{0, 1, 2, 3, \ldots, 9\}
```
Unfolding the network

For the recognition, the network has to be unfolded along the time axis:

The computational complexity is proportional to the number of acoustic transitions.

As shown later, it is favorable to reduce the number of “real” i.e. acoustic transitions.
Language model networks

As shown in the example, a network consists of transitions and nodes.

▶ **Transitions:**
  correspond to spoken words (including silence)

▶ **Nodes:**
  Every word has a start and end node, they define the syntactic (linguistic) context of the transition.

As shown on the next page, a word A can occur in four different contexts.
Language model networks

Possible contexts for a word $A$:

a)

In case a) the automaton is non deterministic.
Bayes Decision Rule and Perplexity

Bayes decision rule and perplexity

- Bayes decision rule using maximum approximation:

\[
[w_1^N]_{opt} = \text{arg max}_{[w_1^N]} \left\{ Pr(w_1^N) \cdot \max_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t \mid s_{t-1}, w_1^N) \right\}
\]

- The perplexity (corpus perplexity / test perplexity) of a language model and a test corpus \([w_1^N]\) is defined as

\[
PP = Pr(w_1^N)^{-\frac{1}{N}} = \left( \prod_{n=1}^{N} Pr(w_n \mid w_1^{n-1}) \right)^{-\frac{1}{N}}
\]
The logarithm of the perplexity then is:

\[
\log PP = \log \left[ Pr(w_1^N)^{-\frac{1}{N}} \right]
\]

\[
= -\frac{1}{N} \sum_{n=1}^{N} \log(Pr(w_n|w_1^{n-1}))
\]

A small perplexity corresponds to strong language model restrictions. Properties of the perplexity:

- normalization: probability per word
- inverse probability: number of possible choices per word position
- probability zero: infinite penalty
Bayes Decision Rule and Perplexity

Now assume constant probabilities (no dependence on the word and between words):

\[ Pr(w_1^N) = \prod_{n=1}^{N} \frac{1}{W} = \left( \frac{1}{W} \right)^N \]

with \( W \) = size of the vocabulary

Then the perplexity becomes:

\[ PP = Pr(w_1^N)^{-\frac{1}{N}} = \left( \frac{1}{W} \right)^{-\frac{1}{N}} = W \]

Note: In this case, the perplexity only depends on the vocabulary size \( W \). Nevertheless, in the general case the perplexity depends on the test corpus it is computed on.
Language Model Networks

Language model $Pr(w_1^N)$ in networks:
A deterministic finite state automaton (DFA) is defined by a transition function $\delta$

$$\begin{align*}
\mathcal{V} &= \{1, \ldots, V\} \quad \text{nodes (linguistic contexts)} \\
\mathcal{W} &= \{1, \ldots, W\} \quad \text{arcs (words including silence)} \\
\delta : \mathcal{V} \times \mathcal{W} &\to \mathcal{V} \\
(v, w) &\to v' = \delta(v, w)
\end{align*}$$

A path in the network defines a word sequence $[w_1^N]$.

For every word $w$ given a node $v$, a probability $p(w|v)$ is defined:

$$p(w|v) = \begin{cases} 
0 & \text{if word } w \text{ does not leave node } v \\
\leq 1 & \text{else}
\end{cases}$$
Language Model Networks

The sum of the probabilities $p(w|v)$ over all words $w$ for each node $v$ is:

$$\sum_{w=1}^{W} p(w|v) = 1$$

Index convention:
$$v_{n+1} := \delta(v_n, w_n)$$
Language model networks

In the general case of non deterministic finite state automata (NFA) the sum over all paths corresponding to a word sequence \( w_1^N \) has to be calculated:

\[
Pr(w_1^N) = \sum_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|v_n) \right\}.
\]

Often the maximum approximation is used:

\[
Pr(w_1^N) \approx \max_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|v_n) \right\}
\]

\[
= \max_{v_1^N} \left\{ \prod_{n=1}^{N} p(w_n|\delta(v_{n-1}, w_{n-1})) \right\}
\]

Note the hierarchical structure of the grammars:

- HMM: acoustic modelling for each word defining the correspondence between word classes and acoustic vectors
- LM: network
Language model networks

Non deterministic and deterministic finite state automata (NFA and DFA):

▶ general case NFA:

\[ p(w, v|v') = p(v|v') \cdot p(w|v', v) \]

Transitionprob. Emissionprob.

▶ special case DFA: given a pair \((v', w)\) the successor state \(v\) is determined by \(v = \delta(v', w)\),

therefore a different factorization of \(p(w, v|v')\) is useful:

\[ p(w, v|v') = p(w|v') \cdot p(v|v', w) \]

with

\[ p(v|v', w) = \begin{cases} 1 & v = \delta(v', w) \\ 0 & v \neq \delta(v', w) \end{cases} \]

For an allowed transition \((v', w) \rightarrow v = \delta(v', w)\) the probability is: \(p(w, v|v') = p(w|v')\)
Dynamic Programming Recursion

Search using language model networks: dynamic programming.

Here the auxiliary quantity $Q_v(t, s; w)$ used to derive the dynamic programming recursion is defined as:

$$Q_v(t, s; w) := \text{probability of the best path at time } t \text{ leading to the state } s \text{ of word } w \text{ with starting node } v.$$ 

Note the additional index $v$. 
Dynamic Programming Recursion

- Within words: acoustic search

\[ Q_v(t, s; w) = \max_{s'} \{ Q_v(t - 1, s'; w) \cdot p(x_t, s|s', w) \} \]

\[ \sigma_v^{opt}(t, s; w) := \arg \max_{s'} \{ Q_v(t - 1, s'; w) \cdot p(x_t, s|s', w) \} \]

\[ B_v(t, s; w) = B_v(t - 1, \sigma_v^{opt}(t, s; w); w) \]

- Word boundaries: language model recombination

\[ Q_v(t - 1, 0; w) = \max_{\{v', w': \delta(v', w') = v\}} \{ Q_{v'}(t - 1, S(w'); w') \cdot p(w|v) \} \]

\[ = p(w|v) \cdot \max_{\{v', w': \delta(v', w') = v\}} \{ Q_{v'}(t - 1, S(w'); w') \} \]

\[ B_v(t - 1, 0, w) = t - 1 \]
Dynamic Programming Recursion

Word boundaries: language model recombination

\[ v' \quad w' \quad w = 1 \]

\[ v' \quad w' \quad w = W \]

\[ v_1 \quad w_1' \]

\[ v_2 \quad w'_N \]

\[ v_3 \]
Dynamic Programming Recursion

The dynamic programming recursion is carried out for every word \( w \) and node \( \nu \). The context defined by \( \nu \) has to be considered in the traceback arrays:

\[
H(\nu, t) = \max_{\{\nu', w' : \delta(\nu', w') = \nu\}} Q_{\nu'}(t, S(w'); w')
\]

Starting node, word:

\[
(V, W)(\nu, t) = \arg \max_{\{\nu', w' : \delta(\nu', w') = \nu\}} Q_{\nu'}(t, S(w'); w')
\]

Backpointer:

\[
B(\nu, t) = B(t, S(W(\nu, t)); W(\nu, t))
\]

The index pair (predecessor node, word) is stored in the traceback array \((V, W)(\nu, t)\).
It can be interpreted as linguistic copy of word \( w \) in the context \( \nu \).
Example

Language model network and corresponding language model transitions:
\(m\)-Gram Language Models

Factorization without restrictions (\(w \in \mathcal{W} \cup \{\$\}; \quad \$ \equiv \text{sentence end})): 

\[
Pr(w_1^N) = \prod_{n=1}^{N} Pr(w_n|w_1^{n-1})
\]

Limit the dependence:

- Unigram LM: 
  \[
  Pr(w_1^N) = \prod_{n=1}^{N} p(w_n)
  \]

- Position Unigram LM: 
  \[
  Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|n)
  \]

- Bigram LM: 
  \[
  Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1})
  \]

- Trigram LM: 
  \[
  Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1})
  \]
Training Bigram LMs

Training bigram language models:
Count words and word pairs:

\[ p(w|v) = \frac{N(v, w)}{N(v)} \]

\( N(v, w) : \) word pair count \((v, w)\) in the training text
\( N(v) : \) count for word \(v\)
Training Bigram LMs

Motivation:

Bigram probability:

\[ Pr(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1}) \]

maximize log-likelihood function:

\[ F = \sum_{n=1}^{N} \log p(w_n|w_{n-1}) \]

with \( \sum_w p(w|v) = 1 \) \( \forall \ v \)

\[ F = \sum_{v,w} N(v,w) \log p(w|v) - \sum_v \mu_v \left[ \sum_w p(w|v) - 1 \right] \]
Training Bigram LMs

Set derivative of log-likelihood w.r.t. \( p(w|v) \) and \( \mu_v \) to zero to obtain maximum:

\[
\frac{\partial F}{\partial p(w|v)} = \frac{N(v, w)}{p(w|v)} - \mu_v = 0
\]

\[
\frac{\partial F}{\partial \mu_v} = \sum_w p(w|v) - 1 = 0
\]

Solution:

\[
p(w|v) = \frac{N(v, w)}{\sum_{w'} N(v, w')} = \frac{N(v, w)}{N(v)}
\]
Discounting

Problem: many pairs \((v, w)\) are not seen in training
\[
N(v, w) = 0,
\]
relative frequency is zero.

**Discounting**: shift probability mass from seen to unseen events.

- Linear discounting:

\[
p(w|v) = \begin{cases} 
(1 - \lambda) \cdot \frac{N(v, w)}{N(v)} & N(v, w) > 0 \\
\lambda \cdot \frac{p(w)}{\sum_{w' : N(v, w') = 0} p(w')} & N(v, w) = 0 
\end{cases}
\]

Estimate \(0 < \lambda < 1\) by Leaving One Out:
Leave \((v, w)\) out of the corpus
\[
\rightarrow \text{change counts: } N(v, w) \rightarrow N(v, w) - 1 \text{ for } N(v, w) > 1
\]
Linear Discounting

Leaving-one-out distribution with linear discounting:

\[
p_{-1}(w|v) = \begin{cases} 
(1 - \lambda) \cdot \frac{N(v, w) - 1}{N(v) - 1} & N(v, w) > 1 \\
\lambda \cdot \frac{p(w)}{\sum_{w': N(v, w') = 1} p(w')} & N(v, w) = 1 
\end{cases}
\]

Log-likelihood criterion:

\[
F(\lambda) = \sum_{v, w} N(v, w) \cdot \log p_{-1}(w|v)
\]

\[
= \sum_{v, w: N(v, w) > 1} N(v, w) \cdot \log(1 - \lambda) \frac{N(v, w) - 1}{N(v) - 1} + \sum_{v, w: N(v, w) = 1} N(v, w) \cdot \log \lambda \frac{p(w)}{\sum_{w': N(v, w') = 1} p(w')}
\]
Linear Discounting

Rewrite log-likelihood criterion:

\[
F(\lambda) = \sum_{v,w:N(v,w)>1} N(v, w) \cdot \log(1 - \lambda) + \sum_{v,w:N(v,w)=1} N(v, w) \cdot \log \lambda
\]

\[
+ \sum_{v,w:N(v,w)>1} N(v, w) \cdot \log \frac{N(v, w) - 1}{N(v) - 1}
\]

\[
+ \sum_{v,w:N(v,w)=1} N(v, w) \cdot \log \frac{p(w)}{\sum_{w' : N(v,w')=1} p(w')}
\]

\[
= \left[ N - \sum_{v,w:N(v,w)=1} N(v, w) \right] \cdot \log(1 - \lambda)
\]

\[
+ \sum_{v,w:N(v,w)=1} N(v, w) \cdot \log \lambda + \text{const}(\lambda)
\]

\[
= (N - n_1) \cdot \log(1 - \lambda) + n_1 \cdot \log(\lambda) + \text{const}(\lambda)
\]
Linear Discounting

Log-likelihood criterion:

\[ F(\lambda) = (N - n_1) \cdot \log(1 - \lambda) + n_1 \cdot \log(\lambda) + \text{const}(\lambda) \]

with \( n_1 \) := \( \sum_{\nu, w: N(\nu, w) = 1} 1 \)

\( \hat{=} \) number of bigram singletons

\( N \) := size of the corpus

Differentiate and set to zero to obtain maximum w.r.t. \( \lambda \):

\[ \lambda = \frac{n_1}{N} \]
Absolute Discounting

- Absolute discounting

\[
p(w|v) = \begin{cases} 
\frac{N(v, w) - b}{N(v)} & N(v, w) > 0 \\
b \cdot \frac{W - W_0(v)}{N(v)} \sum_{w':N(v,w')=0} \frac{p(w)}{p(w')} & N(v, w) = 0
\end{cases}
\]

\(W := \) vocabulary size

\(W_0(v) := \) number of words that do not occur as successor of \(v\).
Absolute Discounting

Leaving-One-Out approach with maximum-likelihood estimation:

\[ F(b) = n_1 \cdot \log(b) + \sum_{v,w:N(v,w) > 1} N(v, w) \log \left( \frac{N(v, w) - 1 - b}{N(v) - 1} \right) \]

\[ = n_1 \cdot \log(b) + \sum_{r > 1} r \cdot n_r \cdot \log(r - 1 - b) + \text{const}(b) \]

with \( n_r := \sum_{v,w:N(v,w)=r} 1 \)

\( \overset{\wedge}{=} \text{Number of word pairs seen } r \text{ times} \)

Differentiate \( F(b) \) by \( b \) and rewrite:

\[ \frac{n_1}{b} - \frac{2n_2}{1 - b} = \sum_{r > 2} \frac{rn_r}{r - 1 - b} \]
Absolute Discounting

There is no closed form solution, but the following estimate can be proven:

\[
\frac{n_1}{n_1 + 2n_2 + \frac{1}{2}[N - n_1 - 2n_2]} \leq b \leq \frac{n_1}{n_1 + 2n_2}
\]

Usually the upper bound is a sufficient estimate

\[
b \doteq \frac{n_1}{n_1 + 2n_2}
\]

For corpora of 10 – 20 million words and a vocabulary of 10,000 – 20,000 words \( b \approx 0.95 \)

Result:
A LM where all \( w_1^N \) are possible to be recognized, i.e. \( Pr(w_1^N) > 0 \).

Ideal case:
- typical word sequence \( w_1^N \): high probability \( Pr(w_1^N) \)
- possible word sequence \( w_1^N \): low probability \( Pr(w_1^N) \)
- untypical word sequence \( w_1^N \): very low probability \( Pr(w_1^N) \)
Using m–grams homophones (phonetically equal words with different spellings) can be distinguished.

Examples from the IBM TANGORA system:

- To, too, two:
  Twenty–two people are too many to be put in this room.

- Right, Wright, write:
  Please write to Mrs. Wright right away.
Bigram LM Complexity

A bigram for the three words A, B, and C represented as network. With $W$ words the network has $W^2$ arcs plus $W$ arcs for silence.

Problem: the computational complexity rises like $W^2$

Introducing empty transitions can help.
Bigram LM Complexity

Bigram LM with empty transitions

Start node: 0             End node: 1,2,3

Schlüter/Ney: Advanced ASR 106 August 5, 2010
Bigram LM Complexity

Bigram LM: silence as part of the words

Start node: 0   End node: 1,2,3
Unfolding the bigram over the time:

- Acoustic transitions
- Empty transitions
- Language model transitions
Unfolding the Bigram LM

Unfolding the bigram over the time (silence as part of the words):

Schlüter/Ney: Advanced ASR 109 August 5, 2010
Bigram LM in Recognition

The network has the following transitions:

\[
\begin{align*}
\text{Sil} & \quad (\text{silence at the beginning of the sentence}) \\
A & \quad \text{word A} \\
B & \quad \text{word B} \\
\vdots & \\
_{A}\text{Sil} & \quad (\text{silence after word } A) \\
_{B}\text{Sil} & \quad (\text{silence after word } B) \\
\vdots & \\
\end{align*}
\]

We augment the vocabulary \( w \) as follows:

\[
w \in \{ \text{Sil}, \ A, \ B, \ \ldots, \ _{A}\text{Sil}, \ _{B}\text{Sil}, \ \ldots \} \]
The auxiliary quantity $Q(t, s, w)$ for dynamic programming is defined as:

$$Q(t, s, w) := \text{probability of the best partial path at time } t \text{ leading to state } s \text{ of word } w.$$ 

The recursion then is:

- Within words:
  $$Q(t, s; w) = \max_{s'} \{ Q(t - 1, s'; w) \cdot p(x_t, s|s', w) \}$$

- Word boundaries:
  $$Q(t - 1, 0; w) = \max_v \{ Q(t - 1, S(v); v) \cdot p(w|v) \}$$

(The special handling of the silence word is not expressed in the equations)
Bigram LM in Recognition

In principle the probability $p(w|v)$ is the LM, but silence transitions require special interpretation:

| Transition | LM Probability $p(w|v)$ |
|------------|-------------------------|
| $A$ $-$ $B$ | $p(B|A)$ |
| $A$ $-$ $A\text{Sil}$ | 1 |
| $A\text{Sil}$ $-$ $B$ | $p(B|A)$ |
| $A\text{Sil}$ $-$ $A\text{Sil}$ | 1 |
| $\text{Sil}$ $-$ $B$ | $p(B)$ : unigram |
| $A\text{Sil}$ $-$ $B\text{Sil}$ | 0 : not possible |
Traceback arrays:

it is easiest to store the decisions about starting words $w$ at time $t$:

\[
\begin{align*}
\text{score:} & \quad H(w, t) = \max_v \{ Q(t, S(v); v) \cdot p(w|v) \} \\
\text{predecessor:} & \quad V(w, t) = \arg \max_v \{ Q(t, S(v); v) \cdot p(w|v) \} \\
\text{backpointer:} & \quad B(w, t) = B(t, S(V(w, t)); V(w, t))
\end{align*}
\]
Remarks:

▶ Due to the regular structure of the bigram it is sufficient to store either the LM nodes or the predecessor words in the traceback arrays.

▶ The traceback at the end of the sentence has to start at the word ends not the word beginnings.

▶ The real implementation is different to optimize memory efficiency:
  ▶ traceback arrays with one index instead of a pair \((w, t)\)
  ▶ when using beam search (following section) the number of word ends reached is smaller, instead of storing word beginnings it is more efficient to store word ends.
Trigram LM

The trigram language model probability is given by:

\[ Pr(w_n | w_1^{n-1}) = p(w_n | w_{n-2}, w_{n-1}) \]

Notation: \((u, v, w) = (w_{n-2}, w_{n-1}, w_n)\)

\(u, v\) are the predecessor words of \(w\), these have to be considered in the LM recombination.

The auxiliary quantity for dynamic programming is defined as:

\[ Q_v(t, s; w) := \text{probability of the best path at time } t \]

leading to the state \(s\) of word \(w\) with predecessor word \(v\).

- For each word \(w\) a copy for every predecessor word \(v\) has to be made.

- The cost of an arc only depend on the arc itself. This allows the practical implementation of dynamic programming.
Unfolding the Trigram LM

Trigram LM recombination:
Trigram LM

Dynamic programming recursion:

► within words:

\[ Q_v(t, s; w) = \max_{s'} \{ Q_v(t - 1, s'; w) \cdot p(x_t, s|s', w) \} \]

► word boundaries:

\[ Q_v(t - 1, 0; w) = \max_u \{ Q_u(t - 1, S(v); v) \cdot p(w|u, v) \} \]

Traceback arrays (at word beginnings):

score: \[ H(v, w, t) = \max_u \{ Q_u(t, S(v); v) \cdot p(w|u, v) \} \]

predecessor: \[ U(v, w, t) = \arg \max_u \{ Q_u(t, S(v); v) \cdot p(w|u, v) \} \]

backpointer: \[ B(v, w, t) = B_U(v, w, t)(t, S(v); v) \]

Silence: in principle, silence is treated as in the bigram LM,

the implementation is more complex.
Trigram LM: Traceback Implementation

- **word string:** $w_1, \ldots, w_n, \ldots, w_N$
- **with word boundaries:** $t_1, \ldots, t_n, \ldots, t_N$
- **sentence end symbol:** $\$ (\hat{\text{Sil}})

- **Note:** traceback in reverse order (start at end with $n = 1$)
- **Initialization:** best word end

  $$(w_2, w_1) = \arg \max_{v, w} \{ Q_v(T, S(w), w) \cdot p(\$| v, w) \}$$

  $$t_1 = T; \quad t_2 = B_{w_2}(T, S(w_1), w_1)$$

- **Loop:** $n = 2$

  while $t_n > 0$ do

  $$\{$$

  $$n = n + 1$$

  $$w_n = U(w_{n-1}, w_{n-2}, t_{n-1})$$

  $$t_n = B(w_{n-1}, w_n, t_{n-1})$$

  $$\}$$

  $$N = n - 1$$

  reverse: $(w_1, t_1), \ldots, (w_N, t_N) \leftarrow (w_N, t_N), \ldots, (w_1, t_1)$
Time complexity for full DP search (later: DP beam search):

\[ T = \text{number of time frames of test utterance} \]
\[ W = \text{number of acoustic reference words} \]
\[ S = \text{average number of states per (acoustic) word} \]
\[ K = \text{number of positions in position unigram} \]

silence model: 1 state

<table>
<thead>
<tr>
<th>language model type</th>
<th>acoustic search comparisons ((= 3 \cdot \text{HMM states}))</th>
<th>language model comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram</td>
<td>(3 \cdot T \cdot [W \cdot S + 1])</td>
<td>(T \cdot [W + 1])</td>
</tr>
<tr>
<td>position unigram</td>
<td>(3 \cdot T \cdot K \cdot [W \cdot S + 1])</td>
<td>(T \cdot K \cdot [W + 1])</td>
</tr>
<tr>
<td>bigram</td>
<td>(3 \cdot T \cdot [W \cdot S + W])</td>
<td>(T \cdot W \cdot [W + 1])</td>
</tr>
<tr>
<td>trigram</td>
<td>(3 \cdot T \cdot W \cdot [W \cdot S + W])</td>
<td>(T \cdot W \cdot [W^2 + 1])</td>
</tr>
</tbody>
</table>
Memory Complexity of DP Search

Memory requirements:

- acoustic search: one column for backpointer and score
- language model recombination: traceback arrays with one entry for each LM node

<table>
<thead>
<tr>
<th>LM type</th>
<th>acoustic search</th>
<th>language model</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram</td>
<td>$2 \cdot [W \cdot S + 1]$</td>
<td>$2 \cdot T$</td>
</tr>
<tr>
<td>position unigram</td>
<td>$2 \cdot K \cdot [W \cdot S + 1]$</td>
<td>$2 \cdot T \cdot K$</td>
</tr>
<tr>
<td>bigram</td>
<td>$2 \cdot [W \cdot S + W]$</td>
<td>$2 \cdot T \cdot [W + 1]$</td>
</tr>
<tr>
<td>trigram</td>
<td>$2 \cdot W \cdot [W \cdot S + \ldots]$</td>
<td>$2 \cdot T \cdot [W^2 + \ldots]$</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Beam Search
Dynamic Programming Beam Search along with Implementation Details

The search consists of principal components:
- language model recombination: word boundaries
- acoustic search: word interior
- bookkeeping: decisions about word (and boundary) hypotheses
- traceback: construct best scoring word sequence

Modifications for large vocabulary systems as opposed to digit string recognition:
- limit the search space by beam search
- modified bookkeeping for active hypotheses (due to beam search)
- modified bookkeeping for traceback arrays (due to beam search)
- garbage collection for traceback arrays (due to beam search)
Traceback Arrays

Traceback arrays

- Use one index:
  - less memory (exhaustive search)
  - beam search, only few word ends are reached

- Bookkeeping is possible at these stages:
  - at the word ends
  - at the LM nodes (most efficient, smallest number of hypotheses)
  - at the word beginnings

- Organization:
  - so far: one element in traceback array per time frame
  - now: backpointer does not point at the time frame the predecessor word ended, it points at the array element with the corresponding information.
Traceback Arrays

Reminder:
The entries of the traceback array define the nodes of a tree:

- Garbage collection (beam search): hypotheses can be pruned, array entries no backpointer points at are labeled as free.
- Partial traceback: if all backpointers of active hypotheses point at one entry in the traceback array the decision before this entry is determined.
- Experimental experience (beam search): delay depends on the task, 1–2 words when using partial traceback.
Beam Search: Pruning

Beam search:

- suboptimal heuristic approach: give up global optimum.
- time synchronous search: remaining cost of the path is constant for all hypotheses.
- baseline method for pruning: discard unlikely hypotheses at every time frame $t$:
  - Acoustic pruning:
    retains state hypotheses whose scores are close to the score of the best state hypothesis:
    \[
    Q_{AC}(t) := \max_{(v,s)} \{ Q_v(t,s) \},
    \]
    prune state hypothesis $(s,t;v)$ iff:
    \[
    Q_v(t,s) < f_{AC} \cdot Q_{AC}(t)
    \]
Beam Search: Pruning

- additional pruning steps:
  - Language model pruning:
    retains tree start-up hypotheses whose score is close to the score of the best tree start-up hypothesis:
    \[
    Q_{LM}(t) := \max_v \{ Q_v(t, s = 0) \}
    \]
    prune tree start-up hypothesis iff:
    \[
    Q_v(t, s = 0) < f_{LM} \cdot Q_{LM}(t),
    \]
  - Histogram pruning:
    limits the number of surviving state hypotheses to a maximum number (MaxHyp).
Using pruning techniques can lead to search errors inducing recognition errors.

Remember: possible reasons for recognition errors:

▶ shortcomings of the acoustic models
▶ shortcomings of the language models
▶ search errors (when using beam search or other heuristic methods)

In general better acoustic models and language models focus the search space, i.e. they allow for tighter pruning thresholds.
Beam Search: Pruning

Illustration of the search process (DP beam search) for connected digit recognition:
Beam Search: Example

Example of the dependency between the search space and the word error rate (WER):
WSJ task, vocabulary size = 20000 words, bigram LM PP = 200.

AcuThr: acoustic pruning threshold.
States: average number of state hypotheses in HMM after pruning.

<table>
<thead>
<tr>
<th>AcuThr k (average)</th>
<th>States (average)</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>252</td>
<td>45.6</td>
</tr>
<tr>
<td>60</td>
<td>677</td>
<td>28.3</td>
</tr>
<tr>
<td>65</td>
<td>1068</td>
<td>24.2</td>
</tr>
<tr>
<td>75</td>
<td>2396</td>
<td>20.6</td>
</tr>
<tr>
<td>100</td>
<td>12908</td>
<td>18.4</td>
</tr>
<tr>
<td>110</td>
<td>21894</td>
<td>18.3</td>
</tr>
<tr>
<td>120</td>
<td>32538</td>
<td>18.2</td>
</tr>
<tr>
<td>130</td>
<td>43862</td>
<td>18.2</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Classes and Dependencies:

RWTH ASR System
- acoustic model
- language model
- corpus handling
- pronunciation lexicon handling
- general search environment

Search::SearchAlgorithm

SearchInterface

LinearSearch

Lexicon

Bookkeeping

LinearSearch::SearchSpace

RWTH ASR Teaching Patch
- interface to RWTH ASR System
- implementation of linear search, including bookkeeping, and traceback
#ifndef  _TEACHING_TYPES_HH
#define  _TEACHING_TYPES_HH

#include  <vector>
#include  <limits>

namespace  Teaching
{
    typedef  unsigned int    Time;
    typedef  unsigned short  Mixture;
    typedef  unsigned int    Word;
    typedef  unsigned short  Phoneme;
    typedef  unsigned short  State;
    typedef  unsigned int    Index;
    typedef  std::vector<Word>    WordSequence;
    typedef  std::vector<Mixture> MixtureSequence;
    typedef  float            Score;

    static  const  Word  invalidWord  = std::numeric_limits<Word>::max();
    static  const  Index invalidIndex = std::numeric_limits<Index>::max();
    static  const  Score maxScore     = std::numeric_limits<Score>::max();
}

#endif  //  _TEACHING_TYPES_HH
Interface to RWTH ASR System

General Interface to Teaching Patch

Class SearchInterface provides connection to general search work around, including handling of configuration, resources (corpus and models) as well as an overall workaround for the specific search implementation.

Main functions to be implemented here are:

- **initialize**: Search initialization
- **processFrame**: expansion of hypotheses to next time frame
- **getResult**: traceback of best recognized word sequence

Implementation:

- show SearchInterface.hh
- show SearchInterface.cc
- show LinearSearch.hh
Interface to RWTH ASR System

Phonem list and Pronunciation Lexicon

Configuration file: XML format, example:

```xml
<?xml version="1.0" encoding="ascii"?>
<lexicon>
    <phoneme-inventory>
        <phoneme><symbol>AE</symbol></phoneme>
        <phoneme><symbol>AH</symbol></phoneme>
        <phoneme><symbol>N</symbol></phoneme>
        <phoneme><symbol>D</symbol></phoneme>
        ...
    </phoneme-inventory>
    <lemma>
        <orth>AND</orth>
        <phon>AE N D</phon>
        <phon>AH N D</phon>
    </lemma>
    ...
</lexicon>

Lexicon configuration file: show an4.lexicon

Implementation: show Lexicon.hh  show Lexicon.cc
Example: Implementation of Dynamic Programming

Beam Search for Bigram LM

Consider:

- Acoustic transitions
- Empty transitions
- Language model transitions

Schlüter/Ney: Advanced ASR
Example: Implementation of Dynamic Programming
Beam Search for Bigram LM

Consider:

```
  A
  B
  C

  Sil

  A Sil
  B Sil
  C Sil

  Sil

  t
  t
  time
```

```
  acoustic transitions
  empty transitions
  language model transitions

  (t=0)
```
Dynamic Handling of State Hypotheses

Goal: complexity should be linear in the number of active hypotheses.
⇒ discard low probability hyps.
⇒ incomplete state hyps.

Efficient expansion of the hypotheses from $t$ to $t + 1$ requires these operations:

- search $(x, S)$
- insert $(x, S)$
- initialize $(S)$
- enumerate $(S)$

Compare methods for set representation:

- dictionary operations
- array representation of sets
- inverted lists and bit vectors
Search Space Representation

Pruning necessitates dynamic handling of word and state hyps:

- List of active words
  \{word, stateHypBegin, stateHypEnd, entryStateHypothesis\}
- List of active states for every word
  \{state, score, backpointer\}
- To address active words, a list of all words pointing into the list of active words is used.
- A list of all states of a word is used to handle active successor states during the expansion of the states of a word.

Implementation: show LinearSearch.cc
## Linear Search Implementation

### Search Space Representation: Word hypotheses

**wordHypothesisMap_**

<table>
<thead>
<tr>
<th>word</th>
<th>index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>invalid</td>
</tr>
<tr>
<td>1</td>
<td>invalid</td>
</tr>
<tr>
<td>2</td>
<td>invalid</td>
</tr>
<tr>
<td>3</td>
<td>invalid</td>
</tr>
<tr>
<td>4</td>
<td>invalid</td>
</tr>
<tr>
<td>5</td>
<td>invalid</td>
</tr>
<tr>
<td>6</td>
<td>invalid</td>
</tr>
<tr>
<td>7</td>
<td>invalid</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>nWords_</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+1</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+2</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+3</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+4</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+5</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+6</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+7</td>
<td>invalid</td>
</tr>
<tr>
<td>nWords_+8</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2*nWords_-</td>
<td>invalid</td>
</tr>
</tbody>
</table>

**wordHypotheses_**

<table>
<thead>
<tr>
<th>word</th>
<th>stateHypBegin</th>
<th>stateHypEnd</th>
<th>entryStateHypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nWords_+8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>invalid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**stateHypotheses_**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Linear Search Implementation

Search Space Representation: State Expansion

wordHypotheses_

stateHypotheses_

newStateHypotheses_

stateHypothesisMap_

word=3
stateHypBegin
stateHypEnd
entryStateHypothesis

state score backpointer
1
2
4
7

state score backpointer
1
2
5
6

state score backpointer
1
2
5
6

index state
1
2
3
4
5
6
7
8
9
10
11
12
13

lexicon[w].size
invalid
invalid
invalid

wordHypotheses_

newStateHypotheses_

stateHypothesisMap_

Schlüter/Ney: Advanced ASR 140 August 5, 2010
Linear Search Implementation
Search Space Representation: Bookkeeping

Implementation: show BookKeeping.hh  show BookKeeping.cc

<table>
<thead>
<tr>
<th>word</th>
<th>score</th>
<th>time</th>
<th>timestamp</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>silence</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>80</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>80</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>97</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>80</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>47</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>63</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>69</td>
<td>90</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>93</td>
<td>100</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>79</td>
<td>90</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>85</td>
<td>80</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>96</td>
<td>100</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>68</td>
<td>70</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>79</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>81</td>
<td>100</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>76</td>
<td>80</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

sentinelBackpointer = 0

bookKeeping_

stateHypotheses_

lastTimestamp_ = 100
Implementation example in C++ code: show Linear Search
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Experiments show that to achieve high performance, it is very important to give the language model $Pr(w_1^N)$ much more weight than the acoustic model $Pr(x_1^T|w_1^N)$.

Why?
Language Model Factor

Starting point:
Bayes decision rule with true models $Pr(w_1^N)$ and $Pr(x_1^T|w_1^N)$:

$$\arg\max_{w_1^N} \left\{ Pr(w_1^N) \cdot Pr(x_1^T|w_1^N) \right\}.$$

In training, we compute an estimate of the true models:

$$Pr(w_1^N) \rightarrow p(w_1^N),$$
$$Pr(x_1^T|w_1^N) \rightarrow p(x_1^T|w_1^N).$$

The shapes (i.e. the weights) of the model distributions are changed by exponentiation with exponents $\alpha$ and $\beta$:

$$p(w_1^N) \rightarrow p^\alpha(w_1^N)$$
$$p(x_1^T|w_1^N) \rightarrow p^\beta(x_1^T|w_1^N).$$
Instead of re-normalizing each individual model separately, we re-normalize by defining the following posterior probability:

\[
p(w^N_1 | x^T_1) = \frac{p^\alpha(w^N_1) \cdot p^\beta(x^T_1 | w^N_1)}{\sum_{\tilde{w}^N_1} p^\alpha(\tilde{w}^N_1) \cdot p^\beta(x^T_1 | \tilde{w}^N_1)} = \frac{p^\alpha(w^N_1) \cdot p^\beta(x^T_1 | w^N_1)}{\text{const}(w^N_1)}
\]
Language Model Factor

Decision rule with weight exponents:

\[ r(x_1^T) = \arg \max_{w_1^N} \left\{ \frac{p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N)}{\text{const}(w_1^N)} \right\} \]

\[ = \arg \max_{w_1^N} \left\{ p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N) \right\} \]

\[ = \arg \max_{w_1^N} \left\{ \log \left[ p^\alpha(w_1^N) \cdot p^\beta(x_1^T|w_1^N) \right] \right\} \]

\[ = \arg \max_{w_1^N} \left\{ \alpha \log p(w_1^N) + \beta \log p(x_1^T|w_1^N) \right\} \]

\[ = \arg \max_{w_1^N} \left\{ \frac{\alpha}{\beta} \log p(w_1^N) + \log p(x_1^T|w_1^N) \right\} \]

The factor \(\alpha/\beta\) is referred to as language model factor (e.g. \(\approx 10 - 15\)).
Consider the posterior probability with suitable word dependent exponents $\beta(w)$:

$$p(w_1^N|x_1^T) = \frac{\prod_n \left[ p^\alpha(w_n|w_1^{n-1}) \cdot p^\beta(w_n)(x'_n|w_n) \right]}{\sum_{\tilde{w}_1^N} \prod_n \left[ p^\alpha(\tilde{w}_n|\tilde{w}_1^{n-1}) \cdot p^\beta(\tilde{w}_n)(x'_n|\tilde{w}_n) \right]}$$
Decision rule with word-dependent weight exponents:

\[
 r(x_1^T) = \arg \max_{w_1^N} \left\{ p(w_1^N|x_1^T) \right\} \\
= \arg \max_{w_1^N} \left\{ \prod_n \left[ p^\alpha(w_n|w_1^{n-1}) \cdot p^{\beta(w_n)}(x_n'|w_n) \right] \right\} \\
= \arg \max_{w_1^N} \left\{ \sum_n \left[ \alpha \log p(w_n|w_1^{n-1}) + \beta(w_n) \log p(x_n'|w_n) \right] \right\} \\
= \arg \max_{w_1^N} \left\{ \sum_n \left[ \log p(w_n|w_1^{n-1}) + \frac{\beta(w_n)}{\alpha} \log p(x_n'|w_n) \right] \right\}
\]

**Effect:** word dependent scale factors \( \beta(w)/\alpha \).

**Training:** like maximum entropy training.
Scale Factors for Each Knowledge Source

Apply scale exponents to each of the knowledge sources: language model, transition and emission probabilities:

\[ p(w_1^N|x_1^T) = \frac{\prod_{n=1}^{N} p^\alpha(w_n|w_{n-2}^n) \cdot \max_{s_1^T} \prod_{t=1}^{T} [p^\beta(s_t|s_{t-1}, w_1^N) \cdot p^\gamma(x_t|s_t, w_1^N)]}{\sum \prod_{\tilde{w}_1^N} \prod_{n=1}^{N} p^\alpha(\tilde{w}_n|\tilde{w}_{n-2}^{n-1}) \cdot \max_{s_1^T} \prod_{t=1}^{T} [p^\beta(s_t|s_{t-1}, \tilde{w}_1^N) \cdot p^\gamma(x_t|s_t, \tilde{w}_1^N)]]} \]
Scale Factors for Each Knowledge Source

Resulting Bayes decision rule:

\[
    r(x_1^T) = \arg \max_{w_1^N} \left\{ p(w_1^N | x_1^T) \right\}
\]

\[
    = \arg \max_{w_1^N} \left\{ \alpha \sum_{n=1}^{N} \log p(w_n | w_{n-1}^{n-2}) + \max_{s_1^T} \sum_{t=1}^{T} [\beta \log p(s_t | s_{t-1}, w_1^N)] + \gamma \log p(x_t | s_t, w_1^N) \right\}
\]
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition
   1.1 Overview: Architecture
   1.2 Phoneme Models and Subword Units
   1.3 Phonetic Decision Trees
   1.4 Language Modelling
   1.5 Dynamic Programming Beam Search
   1.6 Implementation Details
   1.7 Excursion (for experts): Language Model Factor
   1.8 Excursion (for experts): Length Modelling

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Excursion (for experts): Length Modelling

Explicit length models: for $x_1^T$ and $w_1^N$, the lengths $T$ and $N$ are random variables themselves.

- language model: $p(N, w_1^N)$, check normalization:

$$p(N, w_1^N) = p(N) \cdot p(w_1^N | N)$$

$$\sum_{N, w_1^N} p(N, w_1^N) = \sum_N p(N) \sum_{w_1^N} p(w_1^N | N)$$

$$= \sum_N p(N) \cdot \sum_{w_1^N} \prod_{n=1}^{N} p(w_n | w_1^{n-1} | N)$$

$$= \sum_N p(N) \cdot \prod_{n=1}^{N} \sum_{w_n} p(w_n | w_1^{n-1} | N)$$

$$= \sum_N p(N) \cdot 1 = 1$$

- acoustic model: $p(T, x_1^T | w_1^N)$ (check normalization)
Length Modelling

Language model:

\[ p(N, w_1^N) = p(N) \cdot p(w_1^N | N) \]

model assumptions:

\[ = p(N) \cdot \prod_{n=1}^{N} p(w_n | w_{n-1}^{n-1}, N) \]
Length Modelling

- Acoustic model with word boundaries $t_1^N$ (with $t_0 = 0$, $t_N = T$):

\[
p(T, x^T_1 | w_1^N) = \sum_{t_1^{N-1}} p(t_1^N, x^T_1 | w_1^N)
\]

\[
p(t_1^N, x^T_1 | w_1^N) = p(t_1^N | w_1^N) \cdot p(x^T_1 | t_1^N, w_1^N)
\]

model assumptions:

\[
= \prod_{n=1}^{N} \left[ p(t_n | t_{n-1}, w_n) \cdot p(x_{t_{n-1}+1}^{t_n} | w_n, t_{n-1}^n) \right]
\]
Length Modelling: Bayes Decision Rule

Optimization criterion (maximum approximation) using trigram LM $p(w_n|w_{n-1}^{n-2}, N)$ and word segmentation $t_1^N$ (with $t_0 = 0$, $t_N = T$):

$$\max_N \left\{ p(N) \cdot \max_{w_1^N, t_1^N} \prod_{n=1}^{N} \left[ p(w_n|w_{n-1}^{n-2}, N) \cdot p(t_n|t_{n-1}, w_n) \cdot p(x_{t_{n-1}+1}^t|w_n, t_{n-1}^n) \right] \right\}$$

with the length models:

- length dependencies in language models: $p(N)$ and $p(w_n|w_{n-1}^{n-2}, N)$,
- duration models of acoustic models: $p(t_n|t_{n-1}, w_n)$.

Experimental results: rarely tested and no significant improvements.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Review: Linear Search

Bayes decision rule:

\[
\text{Global Search:} \\
\text{maximize} \\
Pr(w_1 \ldots w_N) \cdot Pr(x_1 \ldots x_T | w_1 \ldots w_N) \\
\text{over } w_1 \ldots w_N
\]

Speech Input

Acoustic Analysis

\[x_1 \ldots x_T\]

Global Search:

\[\text{maximize } Pr(x_1 \ldots x_T | w_1 \ldots w_N) \cdot Pr(w_1 \ldots w_N)\]

Recognized Word Sequence

Phoneme Inventory

Pronunciation Lexicon

Language Model
The search process has to find the spoken word sequence in the set of possible word sequences.

The set of possible word sequences can be represented as a tree.
Review: Linear Search

Maximize over all possible wordsequences:

\[
\left[ w_1^N \right]_{opt} = \argmax_{w_1^N} \left\{ Pr(w_1^N) \cdot \sum_{s_1^T} Pr(x_1^T, s_1^T | w_1^N) \right\}
\]

typical assumptions:

- Maximum approximation on state level:

\[
\left[ w_1^N \right]_{opt} = \argmax_{w_1^N} \left\{ Pr(w_1^N) \cdot \max_{s_1^T} Pr(x_1^T, s_1^T | w_1^N) \right\} .
\]

Note:
Only consider state sequences \( s_1^T \) that are consistent with \( w_1^N \).
Review: Linear Search

Assumptions used:

- Bigram language model: \( p(w_n|w_{n-1}) \) or \( p(w|v) \).

When recognizing continuous speech, language model recombination is carried out every 10ms because the word boundaries are not known.
Words are represented as phoneme sequences:

- Typically about 40 phonemes (German, French, English).
- A pronunciation lexicon gives the transcription.
- Words are modelled as concatenation of phoneme models.
- Automatic training.

- Context dependent phonemes:
  - diphones
  - triphones

  i.e. one phoneme in the respective context (not two or three phonemes)
Review: Linear Search

Dynamic programming with beam search:

- left-to-right time synchronous evaluation: direct comparison of scores
- beam search:
  - find best hypothesis at time $t$
  - prune unlikely hypotheses
- experiments: fraction of the potential space (depends on perplexity, acoustic models,...)
- implementation: dynamic search space
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Tree Search

- Every word end hypothesis activates the initial states of all following words.
- Experiments show that in the case of large vocabulary speech recognition (20000 word vocabulary) 90% of the search effort is in the first two phonemes.
- Solution: represent the vocabulary as a lexical prefix tree.

Example with letters instead of phonemes:
- a) tree lexicon, b) linear lexicon.
The phonemes (triphones) at the arcs of the lexical tree are modeled as usual with 6-state HMMs.

At the tree nodes the following transitions are possible:
The following table shows some statistics for the tree lexicon for the so called Wall Street Journal task (19,978 words):

- 44 context independent (CI) phoneme models
- 4,688 context dependent (CD) phoneme models
- A word: typically 6 phonemes

\[ \Rightarrow \text{compression factor: CI: } \frac{6 \cdot 19,978}{44,587} \approx 2.7 \]

\[ \text{CD: } \frac{6 \cdot 19,978}{63,155} \approx 1.9 \]

Terminology:
- Lexicon: linear vs. tree
- Search: linear vs. tree
## Tree Search

<table>
<thead>
<tr>
<th>phoneme generation</th>
<th>number of word ends</th>
<th>number of arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CD tree</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>544</td>
</tr>
<tr>
<td>2</td>
<td>231</td>
<td>3626</td>
</tr>
<tr>
<td>3</td>
<td>1476</td>
<td>9335</td>
</tr>
<tr>
<td>4</td>
<td>2959</td>
<td>12180</td>
</tr>
<tr>
<td>5</td>
<td>3791</td>
<td>11653</td>
</tr>
<tr>
<td>6</td>
<td>3576</td>
<td>9402</td>
</tr>
<tr>
<td>7</td>
<td>2845</td>
<td>6829</td>
</tr>
<tr>
<td>8</td>
<td>2240</td>
<td>4563</td>
</tr>
<tr>
<td>9</td>
<td>1414</td>
<td>2632</td>
</tr>
<tr>
<td>10</td>
<td>778</td>
<td>1359</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>sum</td>
<td>19978</td>
<td>63155</td>
</tr>
</tbody>
</table>
Tree Search

- The average compression factor of a tree lexicon compared to a linear lexicon is about 2 to 3.
- Since the first phonemes require the most search effort the search space reduction using beam search is much larger (a factor of 15 was determined in experiments on the WSJ corpus with a vocabulary of only 5000 words).
- A fundamental problem: the word identity is not known till a word end node is reached. This requires a different language model recombination: Use a separate tree for every predecessor word (only a few trees are active due to beam search).
Tree Search

Bigram language model recombination for a three word vocabulary:

---

Schlüter/Ney: Advanced ASR 170 August 5, 2010
Trigram language model recombination for a three word vocabulary:
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Dynamic Programming

In the previous picture every arc has costs that only depend on the arc itself, not on the history of the preceding path.

Therefore dynamic programming can be used:

Indices: $t$ time, $v$ predecessor word, $w$ word index (defined only at word end states), $s$ state, $S_v$ final state of words $v$ in the tree.

Auxiliary quantity:

$$Q_v(t, s) := \text{probability of the best partial path producing the acoustic vectors } x_1, \ldots, x_t \text{ and ending in state } s \text{ of the tree lexicon with } v \text{ as predecessor word.}$$
Within a word (a tree):

\[ Q_v(t, s) = \max_{\sigma} \{ p(x_t, s|\sigma) \cdot Q_v(t - 1, \sigma) \} \]

Maximize \( \sigma \) over the best predecessor state. At phoneme boundaries the complex transitions have to be considered.

After finishing the computation for the states within a word, the recombination is carried out at the word boundaries:

\[ Q_v(t, 0) = \max_u \{ p(v|u) \cdot Q_u(t, S_v) \} \]

were \( s = 0 \) is a virtual starting state and \( S_v \) the final state of word \( v \).
Silence Copies

The silence model requires a special handling:

- silence has to be allowed between words
- the word before silence has to be stored

Solution:

- a silence arc is added to every tree
- during language model recombination only the root of the same tree can be reached from the silence node.
The silence handling for a trigram language model
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
      2.5 Language Model Look-Ahead
      2.6 Phoneme Look-Ahead
      2.7 Subtree Dominance Recombination (SDR)
      2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Pruning Methods

- Acoustic pruning after acoustic recombination:
  1. determine the score of the best hypothesis at time $t$:
     \[ Q_{AC}(t) := \max_{(v,s)} Q_v(t,s). \]
  2. prune, eliminate a hypothesis $(t, s; v)$ if
     \[ Q_v(t, s) < f_{AC} \cdot Q_{AC}(t). \]
     \[ f_{AC} < 1 \] is the acoustic pruning threshold.

- Effect of acoustic pruning:
  a) narrow minimum of the path scores
     → only a few hypotheses survive the pruning;
  b) broad minimum → many hypotheses survive.
Pruning Methods

- Language model pruning (tree start pruning)
  prune at the tree roots directly after language model recombination:
  - determine the score of the best starting tree hypothesis $t$:
    \[ Q_{LM}(t) := \max_v Q_v(t, s = 0). \]
  - prune a tree hypothesis $(t, s = 0; v)$ if:
    \[ Q_v(t, s = 0) < f_{LM} \cdot Q_{LM}(t). \]
    $f_{LM} < 1$ is the language model pruning threshold.
Pruning Methods

- Histogram pruning is applied after acoustic and language model pruning, if the number of active hypotheses \( N(t) \) at time \( t \) is larger than a value MaxHyp:

\[
N(t) > \text{MaxHyp}.
\]

Histogram pruning only retains the MaxHyp best hypotheses (using bucket sort).
Temporary peaks in the number of hypotheses are eliminated.

Experimental result:
Acoustic pruning is by far the most important method. The others eliminate temporary peaks in the number of hypotheses.
Experimental Results

Experimental results on the WSJ corpus:

- Training corpus: WSJ0,1;
- Test corpus: NAB ’94 H1 development set;
- speaker independent;
- word conditioned tree copies;
- vocabulary:
  - $W = 20\,000$ words;
  - tree: 63,155 arcs (4,688 context dependent phoneme models)
  - 6 states per phoneme
- bigram language model perplexity 215
- $MaxHyp$: 100,000
Experimental Results: Effect of Acoustic Pruning

The acoustic pruning threshold was varied. The number of active states, arcs, trees, and words was measured along with the word error rate (WER).

<table>
<thead>
<tr>
<th>threshold</th>
<th>average number of active states (after pruning)</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 k</td>
<td>252, 80, 4, 9</td>
<td>45.6</td>
</tr>
<tr>
<td>60 k</td>
<td>677, 213, 6, 17</td>
<td>28.3</td>
</tr>
<tr>
<td>65 k</td>
<td>1068, 332, 9, 24</td>
<td>24.2</td>
</tr>
<tr>
<td>75 k</td>
<td>2396, 703, 15, 49</td>
<td>20.6</td>
</tr>
<tr>
<td>100 k</td>
<td>12908, 3554, 38, 238</td>
<td>18.4</td>
</tr>
<tr>
<td>120 k</td>
<td>32538, 8838, 59, 564</td>
<td>18.2</td>
</tr>
<tr>
<td>130 k</td>
<td>43862, 11784, 67, 745</td>
<td>18.2</td>
</tr>
<tr>
<td>170 k</td>
<td>79836, 20649, 87, 1363</td>
<td>18.1</td>
</tr>
</tbody>
</table>
Experimental Results: Effect of Acoustic Pruning

- The word error rate reaches saturation at about 10000 active phoneme arcs or 40000 states.
- The potential search space consists of $20000 \cdot 63000 \cdot 6 \approx 8 \cdot 10^9$ states.
Experimental Results: Effect of Language Model

- Corpus: WSJ (Wall Street Journal) Nov '92, development and eval.
- 5k vocabulary.
- 1.44 h speech data (516984 time frames).
- 18 speakers, 740 sentences, 12137 spoken words.
- 2001 mixtures, 96277 densities.
- Pooled variance vector.
- Feature vector with 33 components.
- LDA (linear discriminant analysis).
- CART (phonetic decision tree).
- Pruning parameter was optimized for trigram LM and kept constant for bi-, uni- und zerogram LM.
Experimental Results: Effect of the Language Model

- LM look-ahead:
  - zero-gram-LM: no LM look-ahead
  - unigram-LM: unigram look-ahead
  - bigram-LM: bigram look-ahead
  - trigram-LM: bigram look-ahead

- real time factor (RTF) measured on a 500 Mhz Alpha-PC with 512 MB memory.
## Experimental Results

### Effect of language model:

<table>
<thead>
<tr>
<th>model</th>
<th>PP</th>
<th># states after expansion</th>
<th>search space after pruning</th>
<th>error rates [%]</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>states</td>
<td>del</td>
<td>ins</td>
</tr>
<tr>
<td>zero-gram</td>
<td>4988</td>
<td>10702</td>
<td>7537</td>
<td>14.6</td>
<td>0.1</td>
</tr>
<tr>
<td>unigram</td>
<td>746</td>
<td>6521</td>
<td>3880</td>
<td>4.6</td>
<td>1.1</td>
</tr>
<tr>
<td>bigram</td>
<td>107</td>
<td>33775</td>
<td>6472</td>
<td>1.4</td>
<td>0.5</td>
</tr>
<tr>
<td>trigram</td>
<td>56</td>
<td>55769</td>
<td>8013</td>
<td>0.7</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Language Model Look-Ahead

- Language model look-ahead
  
  **Problem**: word identity needed for language model in general only know at word ends.

  **Idea**: use the knowledge from the language model as early as possible in the search process. From a certain state (or arc) only a small subset of leaves i.e. word ends can be reached.

\[
W(s) := \text{set of word ends that can be reached from state } s.
\]
Language Model Look-Ahead

Only a few language model probabilities are possible. Define best language model probability reachable from state $s$:

$$\pi_v(s) := \max_{w \in W(s)} p(w|v).$$

For acoustic pruning these values are combined with the original auxiliary scores $Q_v(t, s)$:

$$\tilde{Q}_v(t, s) := \pi_v(s) \cdot Q_v(t, s)$$

and acoustic pruning is applied to $\tilde{Q}_v(t, s)$ instead of $Q_v(t, s)$.

Problem:

- Vocabulary $W = 20000$.
- 4688 context dependent phoneme models.
- Lexikal prefix tree with 63000 phoneme arcs.
- 20000 $\cdot$ 63000 factorized bigram language model probabilities:
  $\Rightarrow$ huge look-up table!
Solution to huge potential number of LM probabilities needed:

- Path compression:
  - upper limit: \( W \) words \( \Rightarrow \) \( 2 \cdot W \) tree nodes
    \( \rightarrow \) compressed LM look-ahead tree
  - optional: only consider the first 2-4 phoneme arc generations of the LM look-ahead tree.

- LM tree factorization on demand.
Examples of a language model look-ahead tree:

LM tree before compression.  

LM tree after compression.
Language Model Look-Ahead

To reduce memory and computation, a unigram language model can be used instead of the bigram model to calculate the look-ahead.

\[
\pi_v(s) := \max_{w \in W(s)} p(w)
\]

Then only one value has to be stored for every node.

**Experimental results: NAB’94 H1 development set**

- 20 speakers (10 men and 10 women).
- 310 utterances with 7387 words, (199 out of vocabulary words).
- RTF (real time factor) measured on an SGI R5000.
## Language Model Look-Ahead

<table>
<thead>
<tr>
<th>LM for look-ahead</th>
<th>LM look-ahead tree gen.</th>
<th>arcs</th>
<th>trees</th>
<th>search space states</th>
<th>arcs</th>
<th>trees</th>
<th>recog. errors [%]</th>
<th>DEL / INS</th>
<th>WER</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>65568</td>
<td>16932</td>
<td>26</td>
<td>2.4 / 2.5</td>
<td>16.3</td>
<td></td>
<td>110.0</td>
</tr>
<tr>
<td></td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>50020</td>
<td>13034</td>
<td>20</td>
<td>2.5 / 2.5</td>
<td>16.6</td>
<td></td>
<td>91.4</td>
</tr>
<tr>
<td>unigram</td>
<td>17</td>
<td>63155</td>
<td>1</td>
<td>16960</td>
<td>4641</td>
<td>32</td>
<td>2.5 / 2.5</td>
<td>16.4</td>
<td></td>
<td>68.1</td>
</tr>
<tr>
<td>(PP = 972.6)</td>
<td></td>
<td></td>
<td></td>
<td>9443</td>
<td>2599</td>
<td>22</td>
<td>2.6 / 2.5</td>
<td>16.8</td>
<td></td>
<td>54.4</td>
</tr>
<tr>
<td>bigram</td>
<td>17</td>
<td>29270</td>
<td>300</td>
<td>3312</td>
<td>935</td>
<td>13</td>
<td>2.5 / 2.6</td>
<td>16.5</td>
<td></td>
<td>32.9</td>
</tr>
<tr>
<td>(PP = 198.4)</td>
<td>3</td>
<td>12002</td>
<td>300</td>
<td>3277</td>
<td>924</td>
<td>13</td>
<td>2.4 / 2.6</td>
<td>16.5</td>
<td></td>
<td>31.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4097</td>
<td>300</td>
<td>3611</td>
<td>1012</td>
<td>12</td>
<td>2.4 / 2.6</td>
<td>16.9</td>
<td></td>
<td>32.2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>544</td>
<td>300</td>
<td>5786</td>
<td>1643</td>
<td>11</td>
<td>2.6 / 2.8</td>
<td>17.0</td>
<td></td>
<td>36.2</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Phoneme Look-Ahead

Phoneme look-ahead:

- look ahead into the 'future';
- by one phoneme (CD, CI);
- for the acoustic data.
Definitions and notation:

\( \alpha \): a phoneme arc in the lexical prefix tree that shall be started.

\( \tilde{\alpha} \): predecessor arc of \( \alpha \). One of its final states has been reached during search.

\( \hat{q}(\alpha; t, \Delta t) \): probability that \( \alpha \) produces the vectors \( x_{t+1}, \ldots, x_{t+\Delta t} \).

The look-ahead time \( \Delta t \) is constant and approximately equals the average phoneme length:

\[
\hat{q}(\alpha; t, \Delta t) := \max_{[s_{t+\Delta t}]} p(x_{t+1}^{t+\Delta t}, s_{t+1}^{t+\Delta t} | \alpha).
\]
Phoneme Look-Ahead

Pruning strategy:

- determine the best state hypothesis $\tilde{Q}_0(t)$:

$$\hat{Q}_v(t, \alpha) := \hat{q}(\alpha; t, \Delta t) \cdot Q_v(t, S_{\tilde{\alpha}})$$

$$\tilde{Q}_0(t) := \max_{(v, \alpha)} \{ \hat{Q}_v(t, \alpha) \}$$

- prune the arc hypothesis $(\alpha, v)$ at time $t$, if:

$$\hat{Q}_v(t, \alpha) < f_{PA} \cdot \tilde{Q}_0(t)$$

- approximate $\tilde{Q}_0(t)$:

$$\tilde{Q}_0(t) \approx \max_{\alpha} \{ \hat{q}(\alpha; t, \Delta t) \} \cdot \max_{(v, \beta)} \{ Q_v(t, S_{\beta}) \}$$
Calculating the look-ahead probability $\hat{q}(\alpha; t, \Delta t)$:

- Time alignment for every hypothesized phoneme $\alpha$:

\[
\Phi_\alpha(\tau, s; t) := \text{probability that the phoneme } \alpha \text{ with the states } 1, \ldots, s \text{ generates the vectors } x_{t+1}, \ldots, x_{t+\tau}.
\]
Phoneme Look-Ahead

- Hidden Markov model with $S = 6$ states:

Phoneme look-ahead probability:

$$\hat{q}(\alpha; t, \Delta t) := \max \left\{ \max_s \{ \Phi_\alpha(\Delta t, s; t) \}, \max_\tau \{ \Phi_\alpha(\tau, S; t)^{\Delta t/\tau} \} \right\}$$
Efficient calculation of the phoneme look-ahead probability:

- approximations:
  - CI phoneme models instead of CD models (44 CI vs. 4688 CD models);
  - smaller number of densities;
  - calculation of the look-ahead every second time frame;
  - collapsed one-state HMM.
Collapsed Hidden Markov Model with one state
Phoneme Look-Ahead

Test conditions: NAB’94-H1-Development 20k

- Acoustic search ('detailed search'):
  - 4 688 CD phoneme models,
  - 290 000 Laplacian densities per gender.
- Phoneme look-ahead:
  - 43 CI phoneme models + 1 silence model,
  - $\Delta t = 6$ Time Frames,
  - 6 state HMM (497 / 1 926 Laplacian mixtures),
  - 1 state HMM (175 / 675 Laplacian mixtures).
- Bigram LM look-ahead.
- RTF (real time factor) measured on a SGI R5000.
**Experimental results:**

<table>
<thead>
<tr>
<th>phoneme look-ahead</th>
<th>phoneme search space</th>
<th>recognition errors</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>[#] states arcs trees</td>
<td>DEL / INS WER</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>-</td>
<td>3312 935 13</td>
<td>16.5</td>
</tr>
<tr>
<td>1-state</td>
<td>175</td>
<td>1901 452 10</td>
<td>16.6</td>
</tr>
<tr>
<td>175</td>
<td>1551 359 9</td>
<td>2.5 / 2.7</td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>1808 434 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>675</td>
<td>1926</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-state</td>
<td>497</td>
<td>2472 605 12</td>
<td>16.5</td>
</tr>
<tr>
<td>497</td>
<td>1671 379 10</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
<tr>
<td>497</td>
<td>2160 516 11</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
<tr>
<td>675</td>
<td>1926</td>
<td></td>
<td></td>
</tr>
<tr>
<td>675</td>
<td>1508 354 10</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
<tr>
<td>6-state</td>
<td>1926</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1926</td>
<td>1784 413 11</td>
<td>2.5 / 2.7</td>
<td></td>
</tr>
<tr>
<td>1926</td>
<td>1808 434 11</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
<tr>
<td>1926</td>
<td>1508 354 10</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
<tr>
<td>1926</td>
<td>1508 354 10</td>
<td>2.5 / 2.6</td>
<td></td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Consider a hypothesis \((s, t; v)\) (bigram LM):
Subtree Dominance Recombination

Consider the acoustic score for the best path from a state $s$ at time $t$ to a final state $S_w$ at time $\tau$:

$$h((t, s) \rightarrow (\tau, S_w)) := \max_{s^\tau_t} \{ Pr(x^\tau_t, s^\tau_t | \cdot) : s_t = s, s_\tau = S_w \}$$

**Note:** $h((t, s) \rightarrow (\tau, S_w))$ does not depend on the preceding word $v$.

Because of the recombination on word level we can eliminate or recombine the hypotheses $(s, t; v)$ if:

$$\forall w \in W(s) := \text{set of leaves i.e. words that can be reached from } s:$$

$$p(w | v) \cdot Q_v(t, s) \cdot h(\ldots) < \max_{v' \neq v} \{ p(w | v') \cdot Q_{v'}(t, s) \cdot h(\ldots) \}$$

$$\iff$$

$$p(w | v) \cdot Q_v(t, s) < \max_{v' \neq v} \{ p(w | v') \cdot Q_{v'}(t, s) \}$$
Subtree Dominance Recombination

The partial tree \((s, t, v_0)\) with

\[
  v_0 = \arg \max_{v'} \{ p(w|v') \cdot Q_{v'}(t, s) \}
\]

dominates the rivaling hypotheses \((s, t; v)\).

Due to the complex \((w, v)\) dependency this equation is not very useful. Simplify both sides by approximation using an upper bound for the left side and a lower bound for the right side:

\[
  \max_{w \in W(s)} p(w|v) \cdot Q_v(t, s) < \max_{v' \neq v} \left\{ \min_{w \in W(s)} p(w|v') \cdot Q_{v'}(t, s) \right\}.
\]

Especially at the beginning of a tree \((s = 0)\)

\[
  \max_{w} p(w|v) \cdot Q_v(t, s = 0) < \max_{v' \neq v} \left\{ \min_{w} p(w|v') \cdot Q_{v'}(t, s = 0) \right\}.
\]
Subtree Dominance Recombination

Note:

a) easy implementation:
   only two arrays with index $v$ are needed:

   $\pi_{\text{max}}(v, s) := \max_{w \in W(s)} p(w|v)$  (cf. LM look-ahead)

   $\pi_{\text{min}}(v, s) := \min_{w \in W(s)} p(w|v)$

b) similarity to LM look-ahead

c) acoustic pruning is sufficient for an efficient focusing search, LM pruning is not needed.
Subtree Dominance Recombination

Experimental results:

- NAB’94 H1 development corpus.
- Vocabulary: 20000 words.
- Bigram language model.
- Language model pruning threshold $f_{LM} = \infty$.
- RTF (real time factor) measured on an SGI R5000.

<table>
<thead>
<tr>
<th>bigram look-ahead</th>
<th>MaxHyp</th>
<th>search space states</th>
<th>arcs</th>
<th>trees</th>
<th>DEL / INS</th>
<th>WER</th>
<th>error rate [%]</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o SDR</td>
<td>150 000</td>
<td>3312</td>
<td>935</td>
<td>13</td>
<td>2.5</td>
<td>2.6</td>
<td>16.5</td>
<td>32.9</td>
</tr>
<tr>
<td>with SDR</td>
<td>$\infty$</td>
<td>3496</td>
<td>887</td>
<td>24</td>
<td>2.4</td>
<td>2.6</td>
<td>16.5</td>
<td>39.3</td>
</tr>
<tr>
<td></td>
<td>150 000</td>
<td>3093</td>
<td>859</td>
<td>12</td>
<td>2.4</td>
<td>2.6</td>
<td>16.5</td>
<td>29.0</td>
</tr>
<tr>
<td></td>
<td>10 000</td>
<td>2594</td>
<td>743</td>
<td>12</td>
<td>2.4</td>
<td>2.6</td>
<td>16.5</td>
<td>25.9</td>
</tr>
<tr>
<td></td>
<td>5 000</td>
<td>2091</td>
<td>606</td>
<td>11</td>
<td>2.4</td>
<td>2.6</td>
<td>16.6</td>
<td>25.6</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree
   2.1 Review: Linear Search
   2.2 Tree Search
   2.3 Dynamic Programming
   2.4 Pruning Methods
   2.5 Language Model Look-Ahead
   2.6 Phoneme Look-Ahead
   2.7 Subtree Dominance Recombination (SDR)
   2.8 Implementation Details

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Implementation in RWTH ASR teaching patch:

- Bigram LM.
- Bookkeeping: at the beginning of each tree.
- Dynamic generation of tree copies.
- "Direct memory access" for trees and phoneme arcs, (compare to corresponding methods in linear search).
- Language model look-ahead.
- NO phoneme look-ahead.
- NO subtree dominance recombination.
RWTH ASR System: Tree Search Teaching Patch

Classes and Dependencies similar to linear search:

- acoustic model
- language model
- corpus handling
- pronunciation lexicon handling
- general search environment

RWTH ASR System

Search::SearchAlgorithm

SearchInterface

WordConditionedTreeSearch

WordConditionedTreeSearch::SearchSpace

Lexicon

Bookkeeping

- interface to open source RWTH ASR System
- implementation of tree search, including bookkeeping, and traceback
Interface to RWTH ASR System

General Interface to Teaching Patch

Class `SearchInterface` provides connection to general search work around as in the linear search implementation (cf. Slides 130ff). It includes handling of configuration, resources (corpus and models) as well as an overall workaround for the specific search implementation.

As for linear search, the main functions to be implemented are:

- **initialize**: Search initialization
- **processFrame**: expansion of hypotheses to next time frame
- **getResult**: traceback of best recognized word sequence

Implementation:

- `show SearchInterface.hh`
- `show SearchInterface.cc`
- `show WordConditionedTreeSearch.hh`
Example: Implementation of Dynamic Programming

Beam Search for Bigram LM

Bigram language model recombination for a three word vocabulary:
Example: Implementation of Dynamic Programming

Beam Search for Bigram LM

Bigram language model recombination for a three word vocabulary including silence:
Bigram LM and Word Conditioned Tree Copies

<table>
<thead>
<tr>
<th>proceed over time $t$ from left to right</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACOUSTIC LEVEL</strong>: process state hypotheses</td>
</tr>
<tr>
<td>- initialization: $Q_v(t-1,0) = H(v; t-1)$</td>
</tr>
<tr>
<td>- time alignment: $Q_v(t,s)$ using DP</td>
</tr>
<tr>
<td>- propagate back pointers in time alignment</td>
</tr>
<tr>
<td>- prune unlikely hypotheses</td>
</tr>
<tr>
<td>- purge bookkeeping lists</td>
</tr>
</tbody>
</table>

**WORD PAIR LEVEL**: process word end hypotheses

**single best**: for each pair $(w; t)$ do

$$H(w; t) = \max_v [p(w|v) Q_v(t, S(w))]$$
$$v_0(w; t) = \arg \max_v [p(w|v) Q_v(t, S(w))]$$

- store best predecessor $v_0 := v_0(w; t)$
- store best boundary $\tau_0 := \tau(t; v_0, w)$
Word Conditioned Tree Search Implementation

Search Space Representation

Pruning necessitates dynamic handling of word and state hyps:

- List of active trees
  \{predecessorWord, arcHypBegin, arcHypEnd\}
- List of active (phoneme/triphone) arcs for every tree
  \{arc, stateHypBegin, stateHypEnd\}
- List of active states for every (phoneme/triphone) arc
  \{state, score, backpointer\}
- To address active trees, a list of all trees pointing into the list of active trees is used, which also hold potential word end hypotheses (re)starting a tree
  \{isActive, *wordHyp\}
- A list of all arcs of a tree is used to handle active successor arcs during the expansion of the states within a tree.

Implementation: show WordConditionedTreeSearch.cc
Tree Search Implementation

Search Space Representation: Tree Hypotheses

**treeMap_**

<table>
<thead>
<tr>
<th>predecessorWord</th>
<th>*wordHyp</th>
<th>isActive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>nWords_-2</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>nWords_-1</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

**treeHypotheses_**

<table>
<thead>
<tr>
<th>predecessorWord</th>
<th>archHypBegin</th>
<th>archHypEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>23</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>24</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>25</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>nWords_-2</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

**archHypotheses_**

<table>
<thead>
<tr>
<th>arc</th>
<th>stateHypBegin</th>
<th>stateHypEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>23</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>24</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>25</td>
<td>21</td>
<td>21</td>
</tr>
</tbody>
</table>

**stateHypotheses_**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>nWords_-2</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>nWords_-1</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>
Tree Search Implementation

Search Space Representation: Within-Tree Arc Hypotheses

- **treeMap_**
  - predecessorWord
    - isActive
    - wordHyp
  - wordHypotheses_
    - word=12 score=a backpointer=b

- **treeLexicon_**
  - a...d e f g

- **SearchSpace::ActiveArcs**
  - activeArcs_
    - d e h a b c f g

- **predecessorMap_**
  - a
    - inv TR
    - s b
    - TR inv
    - s3 b3
  - b
    - inv TR
    - s b
    - TR inv
    - s3 b3
  - c
    - inv TR
    - s b
    - TR inv
    - s3 b3
  - d
    - s FA
    - inv
    - 00
    - inv
  - e
    - s FA
    - 6
    - 5
    - s2
    - a2
    - b2
    - h2
  - f
    - s TR
    - inv
    - 05
    - 00
    - s3
    - inv
    - b3
  - g
    - s TR
    - inv
    - 05
    - 00
    - s3
    - inv
    - b3
  - h
    - s7
    - FA
    - 6
    - 5
    - s2
    - a2
    - b2
    - h2
Tree Search Implementation

Search Space Representation: Initial State Hypotheses per Tree Arc

```
SearchSpace::ActiveArcs
activeArcs_

predecessorMap_

arcHypotheses_

stateHypotheses_

SearchSpace::initTimeAlignment

```

```

hmm (arc=b)

state  | score | backpointer
-------|-------|------------
-1     |     0 |            
 0     | a     | b          
 1     |       |            
 2     |       |            
 3     |       |            
 4     |       |            
 5     |       |            
 6     |       |            

hmm (arc=d)

state  | score | backpointer
-------|-------|------------
-1     |     0 |            
 0     | a     | b          
 1     |       |            
 2     | s4    | b4         
 3     |       |            
 4     | s5    | b5         
 5     | a     | b          
 6     |       |            

hmm (arc=h)

state  | score | backpointer
-------|-------|------------
-1     |     a | b          
 0     | a     | b          
 1     |       |            
 2     | s6    | b6         
 3     |       |            
 4     |       |            
 5     |       |            
 6     |       |            
```
Tree Search Implementation

Search Space Representation: State Expansion per Tree Arc

**SearchSpace::computeTimeAlignment**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**hmm (arc=b)**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

**hmm (arc=d)**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

**hmm (arc=h)**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

**SearchSpace::pruneStatesAndFindWordEnds**

**stateHypotheses_**

<table>
<thead>
<tr>
<th>state</th>
<th>score</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**arcHypotheses_**

<table>
<thead>
<tr>
<th>stateHypBegin</th>
<th>archHypBegin</th>
<th>archHypEnd</th>
<th>predecessorword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**treeHypotheses_**

<table>
<thead>
<tr>
<th>archHypBegin</th>
<th>archHypEnd</th>
<th>predecessorword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
Tree Search Implementation

Search Space Representation: Bookkeeping (as in Linear Search)

Implementation: show BookKeeping.hh show BookKeeping.cc

<table>
<thead>
<tr>
<th>word</th>
<th>score</th>
<th>time</th>
<th>timestamp</th>
<th>backpointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>silence</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>80</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>80</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>97</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>80</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69</td>
<td>90</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93</td>
<td>100</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>79</td>
<td>90</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>85</td>
<td>80</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>100</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>68</td>
<td>70</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>79</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>81</td>
<td>100</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>76</td>
<td>80</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

sentinelBackpointer = 0

<table>
<thead>
<tr>
<th>stateHypotheses_</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

lastTimestamp_ = 100

bookKeeping_
### Trigram LM and Word Conditioned Tree Copies

#### Trigram LM:

- **ACOUSTIC LEVEL:** process states of lexical trees
  - initialization: $Q_{uv}(t-1, 0) = H(u, v; t-1)$
  \[ B_{uv}(t-1, 0) = t - 1 \]
  - time alignment: $Q_{uv}(t, s)$ using DP
  - propagate back pointers $B_{uv}(t, s)$
  - prune unlikely hypotheses
  - purge bookkeeping lists

- **WORD PAIR LEVEL:** process word ends
  - for each triple $(v, w; t)$ do
    - $H(v, w; t) = \max_u \{ p(w|u, v) Q_{uv}(t, S_w) \}$
    - $u_0(v, w; t) = \arg \max_u \{ p(w|u, v) Q_{uv}(t, S_w) \}$
    - store best predecessor $u_0 := u_0(v, w; t)$
    - store best boundary $\tau_0 := B_{u_0v}(t, S_w)$
Generalization to Trigram and $n$-gram

Implementation of tree search for trigram or higher order language models:

▶ General structure of search remains the same - see implementation for bigram (especially within each tree).

▶ Most important difference: hashing instead of full maps for context handling.

<table>
<thead>
<tr>
<th>LM</th>
<th>bigram</th>
<th>trigram</th>
<th>$n$-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>word</td>
<td>word pair</td>
<td>$n - 1$ words</td>
</tr>
<tr>
<td>maps</td>
<td>by word</td>
<td>hash index for context</td>
<td></td>
</tr>
</tbody>
</table>

▶ Generalization to $n$-gram straightforward:

▶ Hashing can be applied to any LM context length, therefore:
▶ Same implementation for arbitrary LM context length.
▶ If larger context helps (lower perplexity), search space will benefit.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Motivation for Across-Word Models

- The words of the vocabulary are modeled with phonemes (example: Airline – ae r l ah ih n).
- The acoustic realization depends on the context of a phoneme (coarticulation).
- This variability is modeled with triphones, i.e. phonemes in a triphone context.
- A model for phoneme $a$ with predecessor $x$ and successor $y$ ($xa_y$) is different from the model of $a$ with predecessor $w$ and successor $z$ ($wa_z$).
Motivation for Across-Word Models

The models for words consist of concatenated phoneme models.

When using triphone context, in principle two ways of modelling are possible:

● Triphone context is only considered within words. At word boundaries the so called “word boundary context” is used (conventional within word modelling).
  + Advantage: easy implementation of the modelling; word models can be taken directly from the lexicon.
  – Disadvantage: coarticulation at word boundaries is not modelled well.

● Triphone context is also considered across word boundaries (across-word modelling).
  + Advantage: coarticulation at word boundaries is taken into consideration.
  – Disadvantage: higher search effort is required. The word models depend on the predecessor and successor words.
Motivation for Across-Word Models

Example for the word: Airline – ae r l ah ih n

- Transcription with context dependencies only within the word:

  #ae_r ae_r l rl ah l ah ih ah ih n ih n#

- Transcription with context dependent models both within the word and at the word boundaries:

  aae_r  bae_r  ih n a

  ae_r l rl ah l ah ih ah ih n

  $ae_r  ih n$

Note: across-word modeling includes potential case $ for no coarticulation at word boundaries (e.g. due to pause).
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Search Implementation with Across-Word Models

- Starting point: word conditioned tree search, context dependent models only within words.
- In principle, there are two possibilities for using across-word models:
  1. Multi pass search:
     - generate a word graph from an initial search without across-word models;
     - extract the $N$ best word sequences from the graph;
     - rescore the $N$ best hypothesized sentences using across-word models.
  2. Single pass search:
     - extend word conditioned tree search to allow processing of context dependencies beyond word boundaries
       - extend the tree lexicon;
       - extend the language model recombination.

- In the following the single pass search strategy will be described.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Extension of the Tree Lexicon

- The conventional tree lexicon only uses within-word models.
- At word boundaries the “word boundary context” is used:

\[ \text{AIRLIFT} \]
\[ \text{AIRLINE} \]
\[ \text{ABOUT} \]

- Note: the consistent word boundary context \# enables the organization of the pronunciations in a lexical prefix tree, with a single path through the tree for each pronunciation.
Extension of the Tree Lexicon

To model coarticulation across word boundaries, the word beginnings and ends have to be expanded (fan-in and fan-out):

![Diagram showing the expansion of the tree lexicon for the words AIRLIFT, AIRLINE, and ABOUT.](image-url)
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Extension of the Language Model Recombination

Review: language model recombination for within-word modelling with a bigram language model.

Notation:
- $A, E, I$: ending/beginning words
- $y$: last generation phoneme
- $x$: phoneme of second to last generation
- $a, b, c$: first generation phonemes
- $e, f, g$: second generation phonemes
- #: general word boundary context
- $\$: silence context, for word transitions with silence between the words
Extension of the Language Model Recombination

When using across-word models:

- Word ends generate more than one arc in the lexical prefix tree.
- For every possible phonetic right across-word context there is an individual arc.
- The hypothesis of a word end is combined with the hypothesis of a successor word.
- The first phoneme of the successor is determined at the ending of a word.
- The hypothesized context influences the future search space.
- Language model recombination can only combine word end hypotheses with the same right across-word context.

The following picture only shows the ending and beginning of word $E$ with the last phoneme $y$. 
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Activating the First Phoneme Generation

At the root of a “tree” the set of arcs that will be activated depends on:

- The last phoneme of the predecessor word (left context of the new arc).
- The right context with which the predecessor word has ended (central phoneme of the new arc).
- special case silence ($$)$: in case of long pauses between the ending and the starting word it is assumed that no coarticulation occurs. Activate all arcs of the first phoneme generation with silence as left context.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
The across-word context of the ending word only influences the first phoneme of the successor.

It is possible to recombine the arcs of the non-coarticulation transition with those of the coarticulation transition.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Dynamic Programming Recursion

- Within-word modelling:
  1. Within words:

\[
Q_v(t, s) = \max_{\sigma} \{ Q_v(t - 1, \sigma) \cdot p(x_t, s|\sigma) \}
\]

2. At word boundaries:

\[
Q_v(t, s = 0) = \max_{u} \{ Q_u(t, S_v) \cdot p(v|u) \}
\]
Dynamic Programming Recursion

- Across-word modelling:
  1. Within words (as before):

\[
Q_v(t, s) = \max_{\sigma} \left\{ p(x_t, s|\sigma) \cdot Q_v(t - 1, \sigma) \right\}
\]

  2. At word boundaries:

The right across-word context \(\chi\) of the ending word \(v\) word has to be considered during language model recombination for this word:

\[
Q_v(t, s_0(\chi)) = \max_u \left\{ p(v|u) \cdot Q_u(t, S_{(v, \chi)}) \right\}
\]

with \(\chi\) right context of the predecessor word \(v\)

\(S_{(v, \chi)}\) last state in fan-out arc of word \(v\) with right context \(\chi\)

\(s_0(\chi)\) virtual start state of subtree following word end hypothesis \(v\) with context \(\chi\). There is only one virtual start state for within-word models, for across-word modelling a different virtual start state \((s_0(\chi))\) for every right context \(\chi\) of predecessor word \(v\) has to be considered.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Extended Language Model Look-Ahead

The hypothesis of a word end with a certain right across-word context restricts the set of possible successor words to the set of words which begin with the phoneme that was hypothesized as right context.

This restricts the set of possible language model probabilities and allows additional pruning.

Define:

\[ W(\chi) := \text{set of words, with } \chi \text{ as first phoneme.} \]

For every ending word \( v \) and every possible right context \( \chi \) the maximal language model probability at the end of a (yet unknown) word \( w \) following \( v \) can be calculated:

\[
\pi(v, \chi) := \max_{w \in W(\chi)} p(w|v)
\]
Extended Language Model Look-Ahead

\[ \pi(v, \chi) := \max_{w \in W(\chi)} p(w | v) \]

- This maximal probability can be considered as soon, as the right context \( \chi \) at the end of the predecessor word \( v \) is hypothesised.
- Let \( \Sigma(v, \chi) \) be the set of states in a fan-out arc of an ending word hypothesis \( v \), with right context \( \chi \).
- The score \( Q_{u}^{FO}(t, \tilde{s}) \) for a state \( \tilde{s} \in \Sigma(v, \chi) \) within a fan-out arc of word \( v \), with right context \( \chi \) in the tree of the predecessor word \( u \) then is:

\[ Q_{u}^{FO}(t, \tilde{s}) := \pi(v, \chi) \cdot Q_{u}(t, \tilde{s}) \]
Compute the overall best fan-out score:

\[ Q^{FO}(t) := \max_{u,v,\chi} \left\{ Q_u^{FO}(t, \tilde{s}) : \tilde{s} \in \Sigma(v, \chi) \right\} \]

A state in the arc of an ending word is pruned if:

\[ Q_u^{FO}(t, \tilde{s}) < f_{AC,FO} \cdot Q^{FO}(t) \]

with the fan-out pruning threshold \( 0 < f_{AC,FO} < 1 \).
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models
   3.1 Motivation for Across-Word Models
   3.2 Implementation of Across-Word Models
   3.3 Extension of the Tree Lexicon
   3.4 Extension of the Language Model Recombination
   3.5 Activating the First Phoneme Generation
   3.6 Recombination after the First Phoneme Generation
   3.7 Dynamic Programming Recursion
   3.8 Extended Language Model Look-Ahead
   3.9 Experimental Results

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Experimental Results

- Verbmobil eval’96, 5k vocabulary
- 35 speakers, 305 utterances, 5421 words
- within-word models (gender independent): 2001 mixtures, 195451 densities
- across-word models (gender independent): 3001 mixtures, 209484 densities
- trigram perplexity 36.4
- **Without** extended language model look-ahead
- RTF (real time factor) on a 500 Mhz AlphaPC

<table>
<thead>
<tr>
<th>models</th>
<th>search space</th>
<th>error rate [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
<td>del.</td>
</tr>
<tr>
<td>within-word</td>
<td>31040</td>
<td>3.3</td>
</tr>
<tr>
<td>across-word</td>
<td>85729</td>
<td>3.7</td>
</tr>
</tbody>
</table>
Experimental Results

▶ NAB’94 H1 development-set, 20k vocabulary
▶ 20 speaker (10 men and 10 woman), 310 utterances with 7387 words (199 out of vocabulary words)
▶ within-word models:
  ▶ female: 2001 mixtures, 279079 densities
  ▶ male: 2001 mixtures, 269141 densities
▶ across-word models:
  ▶ female: 4001 mixtures, 281022 densities
  ▶ male: 4001 mixtures, 354607 densities
▶ trigram perplexity 121.8
▶ With extended language model look-ahead
▶ RTF (real time factor) on a 533 Mhz AlphaPC

<table>
<thead>
<tr>
<th></th>
<th>search space</th>
<th>error rate [%]</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
<td>arcs</td>
<td>trees</td>
</tr>
<tr>
<td>within-word</td>
<td>19599</td>
<td>5821</td>
<td>136</td>
</tr>
<tr>
<td>across-word</td>
<td>17811</td>
<td>2822</td>
<td>37</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Why Word Graphs?

**Goal:** generate search space representation, covering all hypotheses from beam search, which were not pruned.

- **M-best list** of sentences: extract list of $M$ recognized sentences with best recognition score/probability.

**Problem:** inefficient due to redundancy and combinatorial complexity.

- **Word graph:** compact search space representation, containing those word hypotheses during beam search, which were not pruned.

The edges of word graphs contain varying degrees of information about the search space:

- word identity
- acoustic score
- word start and end times
- context
- language model score
- etc., e.g. derived information like confidence score.
Why Word Graphs?

Word graphs are useful for a number of methods:

- multi-pass search: restrict search space for
  - language model rescoring,
  - acoustic model rescoring,
  - unsupervised adaptation (see Chapter 7),
- confidence scoring;
- system combination;
- word error minimization;
- discriminative training (see Chapter 8);
- interface to subsequent applications like machine translation, or natural language understanding.
Bayes’ Decision Rule

Speech Input

Acoustic Analysis

$Pr(w_1 \ldots w_N) \cdot Pr(x_1 \ldots x_T | w_1 \ldots w_N)$

Global Search:

$\text{maximize}$

$Pr(x_1 \ldots x_T | w_1 \ldots w_N)$

over $w_1 \ldots w_N$

Recognized Word Sequence

Phoneme Inventory

Pronunciation Lexicon

Language Model
Original interest in word graphs: language model rescoring in a two-pass approach.

Howe to apply a refined language model?

- One-pass approach: find single-best sentence performing the full recognition process directly using the refined language model.

- Two-pass approach:
  - first pass: use baseline language model (e.g. trigram language model), output (interface): word graph or $M$-best list.
  - second pass: rescore output with refined/final language model, e.g. context-free grammar.

- Extension: multi-pass approach.
Search: Two-Pass Approach

1st Pass:
Baseline Recognition

- Acoustic Vectors $x_1^T$
- Phoneme Inventory
- Pronunciation Lexicon
- Baseline Language Model

$P(x_1^T | w_1^N)$
$P(w_1^N)$

M-best List of Sentence Hypotheses

$w_1^N : P(x_1^T | w_1^N)$

2nd Pass:
Rescoring

- Refined Language Model

$P_2(w_1^N)$

Single Best Sentence
Search: \( M \)-best List of Sentences vs. Word Graph

\( M \)-best list of sentences:

- list \( \mathcal{L}(x_1^T) \) of \( M \) best sentences (\( M = 100, \ldots, 1000 \)),
- for each sentence \( w_1^N \), report acoustic probability \( Pr(x_1^T | w_1^N) \).

The \( M \)-best list \( \mathcal{L}(x_1^T) \) can be considered as a filter:

- first pass: generate list \( \mathcal{L}(x_1^T) \subset \{ w_1^N \} \),
- second pass: find the maximum in this subset of word sequences with refined models

\[
\arg \max_{\tilde{w}_1^N \in \mathcal{L}(x_1^T)} \left\{ Pr(\tilde{w}_1^N) Pr(x_1^T | \tilde{w}_1^N) \right\}
\]

with more complex language models (language model rescoring) or refined acoustic models like across word models (acoustic model model rescoring).
More compact alternative to $M$-best list: word graph.
Attach to each word graph arc:

- word identity $w$,
- beginning time $t_b$ and end time $t_e$,
- acoustic probability $p(x_{t_b}^{t_e} | w)$ of acoustic vectors $x_{t_b}, \ldots, x_{t_e}$.

Example: simple word graph

![Diagram of a simple word graph](image_url)
Why Word Graphs?

Summary of language model rescoring:

▶ separation of two levels:
  ▶ acoustic recognition,
  ▶ postprocessing: refined language model.

▶ Restricted search space representations:
  ▶ list of $M$ best sentences,
  ▶ word graph: network.

▶ Example: 2 word hypotheses per position in a 10-word sentence
  ▶ $M$-best list: $2^{10} = 1024$ sentence hypotheses,
  ▶ word graph: $2 \cdot 10 = 20$ arcs.

▶ Problem for word graph: word boundaries may vary
Examples of Word Graphs

Spoken:
“What is wrong with this picture?”

Spoken:
“Can this network be saved?”
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Word Boundary Function

Definitions:

\[ h(w; \tau, t) := Pr(x_{\tau+1}^t | w) = \text{probability that word } w \text{ produces} \]
\[ \text{the acoustic vectors } x_{\tau+1}^t. \]

\[ G(w_1^n; t) := Pr(w_1^n) \cdot Pr(x_1^t | w_1^n) \]
\[ = (\text{joint}) \text{ probability of generating the acoustic} \]
\[ \text{vectors } x_1^t \text{ and a word sequence } w_1^n \text{ with} \]
\[ \text{ending time } t. \]

Decomposition:

\[ G(w_1^{n-1}; \tau) \]
\[ h(w_n; \tau, t) \]
\[ \cdots \]

\[ W_1 \cdots W_{n-1} \quad W_n \cdots \quad \text{time} \]

\[ \tau \quad t \quad T \]

\[ W_1 \cdots W_{n-1} \quad W_n \cdots \]

\[ W \]

\[ \text{Schlüter/Ney: Advanced ASR 264 August 5, 2010} \]
Word Boundary Function

- Optimization over word boundary $\tau$:

$$G(w^n_1; t) = \max_{\tau} \{ Pr(w_n|w^{n-1}_1) \cdot G(w^{n-1}_1; \tau) \cdot h(w_n; \tau, t) \}$$

$$= Pr(w_n|w^{n-1}_1) \cdot \max_{\tau} \{ G(w^{n-1}_1; \tau) \cdot h(w_n; \tau, t) \}$$

- Define word boundary function:

$$\tau(t; w^n_1) := \arg \max_{\tau} \{ G(w^{n-1}_1; \tau) \cdot h(w_n; \tau, t) \}$$
Recombination using the Word Boundary Function

Consider an $m$-gram language model:

$$p(w_n|w_{n-m+1}^{n-1}) \quad \text{or} \quad p(u_m|u_1^{m-1}).$$

To apply dynamic programming it is sufficient to consider only the last $m - 1$ words $u_2, \ldots, u_{m-1}$.

Define a corresponding auxiliary quantity:

$$H(u_2^m; t) := \max_{w_1^{n-m+1}} \left[ Pr(w_1^n) \cdot Pr(x_1^t|w_1^n) : w_{n-m+2}^n = u_2^m \right]$$

= maximum conditional probability for the generation of the acoustic vectors $x_1\ldots x_t$ by a word sequence that ends with the words $u_2^m$ at time $t$. 
Recombination using the Word Boundary Function

The dynamic programming recursion to calculate the auxiliary quantity $H(u_2^m; t)$ efficiently optimizes over the word $u_1$ and the word boundary $\tau$:

$$H(u_2^m; t) = \max_{u_1} \left[ p(u_m | u_1^{m-1}) \cdot \max_{\tau} \left\{ H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t) \right\} \right].$$

Maximum approximation is used here to avoid the sum over all word boundaries $\tau$.

The definition of the corresponding word boundary function becomes:

$$\tau(t; u_1^m) := \arg \max_{\tau} \left\{ H(u_1^{m-1}; \tau) \cdot h(u_m; \tau, t) \right\}$$

leading to:

$$H(u_2^m; t) = \max_{u_1} \left[ p(u_m | u_1^{m-1}) \cdot H(u_1^{m-1}; \tau(t; u_1^m)) \cdot h(u_m; \tau(t; u_1^m), t) \right].$$
Recombination using the Word Boundary Function

Example: word boundary function with a bigram language model:

\[ \tau(t; u_1, u_2) := \arg \max_{\tau} \{ H(u_1; \tau) \cdot h(u_2; \tau, t) \} \]

The word boundary depends on the identity of the surrounding words.
Word Pair Approximation

Remember: \( \tau = \) start time of hypothesis \((t, w)\)

**Assumption:** \( \tau \) only depends on predecessor word \( v \)

(in addition to \((t, w)\)): \( \tau = \tau(t; v, w) \)

Experiments show: approximation correct if predecessor word \( v \) sufficiently long.
Two examples for word pair approximation:

**good example:** predecessor word $u_{m-1} := v$ of word $u_m := w$ is sufficiently long.

**bad example:** predecessor word $u_{m-1} := v$ of word $u_m := w$ is too short.
Algorithm in words:

- For each time frame $t$, consider all word pairs $(v, w)$ using beam search.
- For each triple $(t; v, w)$, keep track of:
  - the word boundary $\tau(t; v, w)$,
  - the word score $h(w; \tau(t; v, w), t)$.
- End of sentence: traceback.
Algorithm for Word Graph Construction

<table>
<thead>
<tr>
<th>proceed over time $t$ from left to right</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACOUSTIC LEVEL</strong>: process states of lexical trees</td>
</tr>
</tbody>
</table>
| - initialization: $Q_v(t-1,0) = H(v; t-1)$  
  $B_v(t-1,0) = t-1$ |
| - time alignment: $Q_v(t,s)$ using DP  
  - propagate back pointers $B_v(t,s)$  
  - prune unlikely hypotheses  
  - purge bookkeeping lists |
| **WORD PAIR LEVEL**: process word ends |
| single best: for each pair $(w; t)$ do  
  $H(w; t) = \max_v \{ p(w|v) Q_v(t, S_w) \}$  
  $v_0(w; t) = \arg \max_v \{ p(w|v) Q_v(t, S_w) \}$ |
| - store best predecessor $v_0 := v_0(w; t)$  
  - store best boundary $\tau_0 := B_{v_0}(t, S_w)$ |
| word graph: for each triple $(t; v, w)$ store  
  - word boundary $\tau(t; v, w) := B_v(t, S_w)$  
  - word score $h(w; \tau, t) := Q_v(t, S_w)/H(v; \tau)$ |
| **Traceback after last time frame $T$ is processed:** |
| - single best: start at best sentence end: $w = \arg \max_v \{ p($$|v) H(v, T) \}$  
  - word graph: start at all possible sentence ends. |
Algorithm for Word Graph Construction

Depending on how they are constructed, different types of word graphs can be defined:

- **Word pair approximation** ⇒ word/predecessor conditioned word graphs
  - Word sequence implies sequence of word boundaries (unique).
  - Each word sequence occurs only once.
- **Generalization**: word triple, quadruple, etc. approximation (cf. language model context).
- **Other methods** ⇒ time conditioned word graphs
  - A word sequence might occur more than once with different word boundary times.

Depending on what is to be done with the word graph, one or the other method might be suitable.

Is it possible to guarantee that the $M$-best sentences are contained in the word graph?

- It is highly probable, but
- there is no formal proof.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Pruning methods can also be applied on word graphs.

- **Forward Pruning:**
  - Word end pruning retains word hypotheses \((v, w; t)\) whose scores are relatively close to score of best word hypothesis at time \(t\):
    \[
    H_{\text{max}}(t) = \max_w \{ H(w; t) \}
    \]
    with
    \[
    H(w; t) = \max_v \{ p(w|v) \cdot H(v; \tau(t; v, w)) \cdot h(w; \tau(t; v, w), t) \}.
    \]
  - Prune hypothesis \((v, w; t)\) iff:
    \[
    p(w|v) \cdot H(v; \tau(t; v, w)) \cdot h(w; \tau(t; v, w), t) < f_{LAT} \cdot H_{\text{max}}(t),
    \]
    with the word graph pruning threshold \(f_{LAT} < 1\).
  - In addition, histogram pruning limits the number of surviving word hypotheses to a maximum number (*typically 1000 per time frame*).
Word Graph Pruning

- During acoustic search, forward pruning is the standard approach.
- Similar to acoustic and language model pruning, forward pruning does not guarantee to find the best scoring word sequence.
- Often the need arises to prune word graphs after generating initial, usually larger word graphs.
- Observation: when further pruning word graphs after their generation, forward pruning becomes less effective, since in contrast to the initial generation of the word graph, the score of the overall best word sequence already is known.
- Optimal approach: prune word graph relative to the best recognized/best scoring word sequence.
- Advantage: the best scoring word sequence will not be pruned itself, which is not guaranteed in forward-only pruning.
Word Graph Pruning

Forward-Backward Pruning:

- **Idea:** for every edge \((w; \tau, t)\) in the word graph the score \(Q(w; \tau, t)\) of the best path through the word graph including this arc is calculated.

- An arc \((w; \tau, t)\) is pruned iff:

\[
Q(w; \tau, t) > F_{LAT\_FB} + \max_{(w; \tau, t)} \{Q(w; \tau, t)\},
\]

where \(F_{LAT\_FB} > 0\) is the forward-backward pruning threshold in the log domain.

- The term “forward-backward” relates to the corresponding twofold dynamic programming algorithm to compute this pruning step efficiently.
To calculate $Q(w; \tau, t)$ efficiently, it is decomposed into the sum of a forward and a backward score:

$$Q(w; \tau, t) = Q_f(w; \tau, t) + Q_b(w; \tau, t),$$

with

$$Q_f(w; \tau, t) := -\log p(w; \tau, t | x_1^t) = -\log \text{prob. to end in an edge (w; \tau, t) given } x_1^t,$$

$$Q_b(w; \tau, t) := -\log p(w; \tau, t | x_{t+1}^T) = -\log \text{prob. to start with edge (w; \tau, t) followed by } x_{t+1}^T.$$

The forward score $Q_f(w; \tau, t)$ and backward score $Q_b(w; \tau, t)$ can then be computed efficiently by a dynamic programming search pass each on the word graph:

1. forward pass from $t = 1$ to $t = T$
2. backward pass from $t = T$ to $t = 1$
Word Graph Pruning: Experimental Results

Experimental results on NAB’94 (ARPA Benchmark 1994)

- NAB’94 (development 20k)
- \( W = 20,000 \) word vocabulary
- 199 Out of vocabulary words (2.7%)
  - 20 speakers (10 female, 10 male)
  - 310 sentences (155 per gender)
  - 7387 spoken words
- Recognizer setup
  - 2001 context dependent phoneme models (gender independent)
  - 216,000 densities
  - Trigram language model: perplexity 121.8
  - Single best word error rate: 14.5%

Additional evaluation measure for word graphs:

- GER [%]: graph error rate. Error rate of that hypothesis in the word graph having the smallest Levenshtein distance to the spoken word sequence.
## Word Graph Pruning: Experimental Results

### Forward Pruning

<table>
<thead>
<tr>
<th>$-\log f_{Lat}$</th>
<th>arcs</th>
<th>nodes</th>
<th>word boundaries</th>
<th>GER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>500.00</td>
<td>105.07</td>
<td>45.70</td>
<td>10.54</td>
<td>4.44</td>
</tr>
<tr>
<td>300.00</td>
<td>99.74</td>
<td>45.12</td>
<td>10.20</td>
<td>4.44</td>
</tr>
<tr>
<td>250.00</td>
<td>89.11</td>
<td>43.69</td>
<td>9.50</td>
<td>4.45</td>
</tr>
<tr>
<td>200.00</td>
<td>71.87</td>
<td>40.85</td>
<td>8.39</td>
<td>4.51</td>
</tr>
<tr>
<td>180.00</td>
<td>59.31</td>
<td>35.55</td>
<td>7.42</td>
<td>4.62</td>
</tr>
<tr>
<td>160.00</td>
<td>46.52</td>
<td>29.34</td>
<td>6.35</td>
<td>4.81</td>
</tr>
<tr>
<td>140.00</td>
<td>33.74</td>
<td>22.40</td>
<td>5.20</td>
<td>5.08</td>
</tr>
<tr>
<td>120.00</td>
<td>22.86</td>
<td>15.96</td>
<td>4.20</td>
<td>5.40</td>
</tr>
<tr>
<td>100.00</td>
<td>14.07</td>
<td>10.31</td>
<td>3.29</td>
<td>5.96</td>
</tr>
<tr>
<td>80.00</td>
<td>8.23</td>
<td>6.36</td>
<td>2.59</td>
<td>6.85</td>
</tr>
<tr>
<td>60.00</td>
<td>4.78</td>
<td>3.89</td>
<td>2.10</td>
<td>7.89</td>
</tr>
<tr>
<td>40.00</td>
<td>2.97</td>
<td>2.55</td>
<td>1.76</td>
<td>9.39</td>
</tr>
<tr>
<td>20.00</td>
<td>2.05</td>
<td>1.85</td>
<td>1.54</td>
<td>10.79</td>
</tr>
<tr>
<td>10.00</td>
<td>1.77</td>
<td>1.65</td>
<td>1.45</td>
<td>11.63</td>
</tr>
<tr>
<td>5.00</td>
<td>1.64</td>
<td>1.56</td>
<td>1.40</td>
<td>11.90</td>
</tr>
<tr>
<td>1.00</td>
<td>1.54</td>
<td>1.48</td>
<td>1.36</td>
<td>12.12</td>
</tr>
</tbody>
</table>
## Word Graph Pruning: Experimental Results

### Forward-Backward Pruning

<table>
<thead>
<tr>
<th>$F_{Lat_{fb}}$</th>
<th>arcs</th>
<th>nodes</th>
<th>word boundaries</th>
<th>GER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>750.00</td>
<td>97.07</td>
<td>43.11</td>
<td>9.96</td>
<td>4.51</td>
</tr>
<tr>
<td>500.00</td>
<td>82.47</td>
<td>39.92</td>
<td>9.96</td>
<td>4.53</td>
</tr>
<tr>
<td>250.00</td>
<td>27.55</td>
<td>17.21</td>
<td>6.00</td>
<td>4.78</td>
</tr>
<tr>
<td>200.00</td>
<td>16.95</td>
<td>11.26</td>
<td>4.58</td>
<td>4.95</td>
</tr>
<tr>
<td>180.00</td>
<td>13.45</td>
<td>9.17</td>
<td>4.03</td>
<td>4.98</td>
</tr>
<tr>
<td>160.00</td>
<td>10.51</td>
<td>7.36</td>
<td>3.54</td>
<td>5.28</td>
</tr>
<tr>
<td>140.00</td>
<td>8.03</td>
<td>5.78</td>
<td>3.08</td>
<td>5.58</td>
</tr>
<tr>
<td>120.00</td>
<td>6.06</td>
<td>4.49</td>
<td>2.66</td>
<td>5.92</td>
</tr>
<tr>
<td>100.00</td>
<td>4.58</td>
<td>3.51</td>
<td>2.32</td>
<td>6.44</td>
</tr>
<tr>
<td>80.00</td>
<td>3.48</td>
<td>2.77</td>
<td>2.03</td>
<td>7.16</td>
</tr>
<tr>
<td>60.00</td>
<td>2.68</td>
<td>2.23</td>
<td>1.79</td>
<td>8.11</td>
</tr>
<tr>
<td>40.00</td>
<td>2.09</td>
<td>1.82</td>
<td>1.59</td>
<td>9.67</td>
</tr>
<tr>
<td>20.00</td>
<td>1.65</td>
<td>1.51</td>
<td>1.42</td>
<td>11.87</td>
</tr>
<tr>
<td>10.00</td>
<td>1.45</td>
<td>1.38</td>
<td>1.33</td>
<td>13.04</td>
</tr>
<tr>
<td>5.00</td>
<td>1.35</td>
<td>1.31</td>
<td>1.29</td>
<td>13.75</td>
</tr>
<tr>
<td>1.00</td>
<td>1.25</td>
<td>1.25</td>
<td>1.24</td>
<td>14.34</td>
</tr>
</tbody>
</table>
Word Graph Pruning: Experimental Results

The diagram shows the graph error rate (%) as a function of word graph density. Two pruning methods are compared: forward pruning and forward backward pruning. As the word graph density increases, the graph error rate decreases for both methods, with forward pruning showing a more steep decrease compared to forward backward pruning.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Multi-Pass Search

Word graphs can be used as search space representation for consecutively constraining the search space to be used.

Properties of word graphs:

- incoming arcs: maximum = vocabulary size
- outgoing arcs: no maximum
- silence is included as integrated part of the words

Search over word graph:

- Dynamic programming through graph.
Multi-Pass Search

Using word graphs as (constrained) search space representations, a number of multiple pass search strategies are possible:

- **Language model rescoring:**
  - First pass full search with baseline language model to generate word graph.
  - Second pass search over word graph using advanced language model.
  - Word boundaries and acoustic scores are kept constant (from first pass).

- **Acoustic rescoring:**
  - First pass full search with baseline acoustic model, e.g. using monophone or within-word triphones.
  - Second pass search over word graph using advanced acoustic model, e.g. across-word triphones.
  - If word boundaries are kept constant, acoustic rescoring can be done per arc/word.
Multi-Pass Search

Using word graphs as (constrained) search space representations, a number of multiple pass search strategies are possible:

- **Language model rescoring:**
  - First pass full search with baseline language model to generate word graph.
  - Second pass search over word graph using advanced language model.
  - Word boundaries and acoustic scores are kept constant (from first pass).

- **Acoustic rescoring:**
  - First pass full search with baseline acoustic model, e.g. using monophone or within-word triphones.
  - Second pass search over word graph using advanced acoustic model, e.g. across-word triphones.
  - If word boundaries are kept constant, acoustic rescoring can be done per arc/word.
  - Word boundaries can also be recalculated: full search constrained to word graph (discarding first path word boundaries).
Multi-Pass Search

▶ In multi-pass search strategies, language model and acoustic model rescoring usually are combined, forming a consecutive sequence of search passes which are increasingly constrained.

▶ Potential advantage: the application of simpler models at earlier search stages with larger search spaces might help reduce the search complexity, allowing the application of more complex and computationally demanding models to be applied in a more constrained search space.

▶ Disadvantages:
  ▶ Each intermediate step in a multi-pass search strategy represents decisions to constrain the further search space based on incomplete knowledge (i.e. disregarding more advanced models to be applied at a later stage). Hypotheses discarded at some level can not or only partly be recovered in later stages of the search process.
  ▶ Less advanced models call for less severe pruning, i.e. the reduction of the search space is limited by the models used and needs to be optimized.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
### Language Model Rescoring

Word graph rescoring algorithm (2\textsuperscript{nd} pass):

<table>
<thead>
<tr>
<th>input:</th>
<th>word graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set of word boundaries $\tau(t; u^m_1)$</td>
</tr>
<tr>
<td></td>
<td>set of word scores $h(u_m; \tau, t)$</td>
</tr>
</tbody>
</table>

- proceed over time $t$ from left to right

- process each word pair $u^m_{m-1}$ in the graph
  - get word boundaries $\tau(t; u^m_1)$ and scores $h(u_m; \tau, t)$
  - process all word sequences $u^m_1$

\[
\hat{H}(u^m_1; t) := p(u_m|u^{m-1}_1) \cdot H(u^{m-1}_1; \tau) \cdot h(u_m; \tau, t)
\]

\[
H(u^m_2; t) = \max_{u_1} \hat{H}(u^m_1; t)
\]

\[
B(u^m_2; t) = \arg \max_{u_1} \hat{H}(u^m_1; t)
\]

- traceback: use back pointers $\{B(u^m_2; t)\}$

**Note:** the language model used here is different from the one used to construct the word graph.
Word Graph Rescoring: Results

Test: NAB’94 – H1 Development

- vocabulary: 20000 words
- 10 male and 10 female speakers
- total of 310 sentences = 7387 spoken words
- unknown words: 199 (out of vocabulary rate: 2.7 %)
- test set perplexities:
  - bigram: $PP_{bi} = 198.1$
  - trigram: $PP_{tri} = 130.2$

Acoustic Modelling:

- trained on: WSJ0 and WSJ1 (284 speakers; 80 h of speech)
- phoneme models: 4688 (43+1 monophones, 557 diphones, 4087 triphones)
- tied states: 4623 (6-state HMM)
- densities: 290 000 per gender
Word Graph Method: Experimental Results

Search space for word graph generation:

- search space: 27672 states, 7674 arcs, 115 trees

Rate of out of vocabulary words (OOV): 2.7%.

Word graph and evaluation measures:

WGD: word graph density (word arcs per spoken word)
NGD: node graph density (nodes per spoken word)
BGD: boundary graph density
   (different word boundaries per spoken word)
GER[%]: word graph error rate ('sub,del,ins')
WER[%]: word recognition error rate
   (trigram: $PP_{tri} = 130.2$)
## Word Graph Method: Results

<table>
<thead>
<tr>
<th>$f_{\text{Lat}}$</th>
<th>Graph density</th>
<th>Graph word error rate</th>
<th>Recognition word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WGD</td>
<td>NGD</td>
<td>BGD</td>
</tr>
<tr>
<td>300</td>
<td>1476</td>
<td>181.80</td>
<td>19.67</td>
</tr>
<tr>
<td>150</td>
<td>1415</td>
<td>175.93</td>
<td>19.25</td>
</tr>
<tr>
<td>100</td>
<td>684</td>
<td>104.22</td>
<td>14.50</td>
</tr>
<tr>
<td>90</td>
<td>460</td>
<td>77.35</td>
<td>12.31</td>
</tr>
<tr>
<td>80</td>
<td>269</td>
<td>51.75</td>
<td>9.84</td>
</tr>
<tr>
<td>70</td>
<td>137</td>
<td>31.43</td>
<td>7.42</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>17.20</td>
<td>5.33</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>8.98</td>
<td>3.77</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>4.79</td>
<td>2.66</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>2.75</td>
<td>2.01</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>1.83</td>
<td>1.60</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1.41</td>
<td>1.36</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1.29</td>
<td>1.28</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.25</td>
<td>1.24</td>
</tr>
</tbody>
</table>
Recognition Results: Word Graph vs. One Pass

(A) two pass search (word conditioned bigram search with word graph construction and subsequent word graph rescoring with a trigram).

(B) single pass time conditioned trigram search (Chapter 5)

- NAB H1 development corpus; 20k vocabulary
- 310 test sentences, 7387 words
- Bigram LM with a perplexity of 198
- Trigram LM with a perplexity of 130

<table>
<thead>
<tr>
<th>Number of:</th>
<th>sentences</th>
<th>word errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>two pass (A) / single pass (B)</td>
</tr>
<tr>
<td>identical score</td>
<td>277</td>
<td>898</td>
</tr>
<tr>
<td>(A) is better than (B)</td>
<td>1</td>
<td>11 / 11</td>
</tr>
<tr>
<td>(B) is better than (A)</td>
<td>32</td>
<td>146 / 121</td>
</tr>
<tr>
<td>total</td>
<td>310</td>
<td>1055 / 1030</td>
</tr>
</tbody>
</table>
Recognition Results: Word Graph vs. One Pass

(A) two pass search (word conditioned bigram search with word graph construction and subsequent word graph rescoring with a trigram).

(B) single pass time conditioned trigram search (Chapter 5)

- NAB H1 development corpus; 64k vocabulary
- 310 test sentences, 7387 words
- Bigram LM with a perplexity of 237
- Trigram LM with a perplexity of 172

<table>
<thead>
<tr>
<th>Number of:</th>
<th>sentences</th>
<th>word errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>two pass (A) / single pass (B)</td>
</tr>
<tr>
<td>identical score</td>
<td>220</td>
<td>452 / 452</td>
</tr>
<tr>
<td>(A) is better than (B)</td>
<td>1</td>
<td>2 / 2</td>
</tr>
<tr>
<td>(B) is better than (A)</td>
<td>89</td>
<td>439 / 423</td>
</tr>
<tr>
<td>total</td>
<td>310</td>
<td>893 / 877</td>
</tr>
</tbody>
</table>
Examples of spoken test sentences (SNT):

- integrated search method (ISM)
- word graph method (WGM)

<table>
<thead>
<tr>
<th>SNT:</th>
<th>... A FEW MUTUAL FUND INVESTORS INTO MODELS OF ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>WGM:</td>
<td>... A FEW MUTUAL FUND INVESTORS IN TWO MODELS OF ...</td>
</tr>
<tr>
<td>ISM:</td>
<td>... A FEW MUTUAL FUND INVESTORS INTO MODELS OF ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNT:</th>
<th>... WHEN YOU BUY A MUTUAL FUND FROM YOUR BROKER</th>
</tr>
</thead>
<tbody>
<tr>
<td>WGM:</td>
<td>... WHEN YOU BUY A MUTUAL FUND FROM A BROKER</td>
</tr>
<tr>
<td>ISM:</td>
<td>... WHEN YOU BUY A MUTUAL FUND FROM YOUR BROKER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNT:</th>
<th>I ALMOST WISH I HADN’T SEEN THIS PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>WGM:</td>
<td>I ALMOST WISH I HADN’T SEEN AS PART</td>
</tr>
<tr>
<td>ISM:</td>
<td>I ALMOST WISH I HADN’T SEEN THIS PART</td>
</tr>
</tbody>
</table>
Bigram Cache Model using Word Graph

Cache (unigram/bigram) language model:

- Cache LM is difficult to handle in integrated search
- Cache updated after each sentence boundary (not after each word!)
- Two modes: supervised/unsupervised

<table>
<thead>
<tr>
<th>Language model</th>
<th>PP</th>
<th>del - ins</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigram:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without cache</td>
<td>198.1</td>
<td>179 - 195</td>
<td>16.5</td>
</tr>
<tr>
<td>with cache:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsupervised</td>
<td>178.0</td>
<td>190 - 179</td>
<td>16.3</td>
</tr>
<tr>
<td>supervised</td>
<td>171.2</td>
<td>194 - 179</td>
<td>16.0</td>
</tr>
<tr>
<td>Trigram:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without cache</td>
<td>130.2</td>
<td>120 - 202</td>
<td>14.3</td>
</tr>
<tr>
<td>with cache:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unsupervised</td>
<td>117.1</td>
<td>127 - 182</td>
<td>14.0</td>
</tr>
<tr>
<td>supervised</td>
<td>113.1</td>
<td>128 - 194</td>
<td>13.8</td>
</tr>
</tbody>
</table>
Compare computational search complexity, test condition:

- subset of NAB’94 H1 development set:
  - 10 male and 10 female speakers,
  - 20 sentences = 519 spoken words
    (of which 21 words are out of vocabulary)
  - 211 sec

- perplexities:
  \[ PP_{bi} = 201.3, \quad PP_{tri} = 134.1 \]

- SGI workstation (Indy R4400, SpecInt’92: 94)
## Search effort: Word Graph vs. Integrated Method

<table>
<thead>
<tr>
<th>Method:</th>
<th>Word Graph states/arcs/trees</th>
<th>Integrated states/arcs/trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search space:</td>
<td>16274 / 4516 / 41</td>
<td>32291 / 9413 / 108</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-likelihood computation</td>
<td>14830 [sec] 78.2 [%]</td>
<td>15174 [sec] 67.2 [%]</td>
</tr>
<tr>
<td>Acoustic search</td>
<td>3603 [sec] 19.0 [%]</td>
<td>6576 [sec] 29.1 [%]</td>
</tr>
<tr>
<td>Lang. model recombination</td>
<td>139 [sec] 0.7 [%]</td>
<td>621 [sec] 2.7 [%]</td>
</tr>
<tr>
<td>Other operations</td>
<td>177 [sec] 1.0 [%]</td>
<td>208 [sec] 1.0 [%]</td>
</tr>
<tr>
<td>Overall recognition</td>
<td>18959 [sec] 100.0 [%]</td>
<td>22579 [sec] 100.0 [%]</td>
</tr>
<tr>
<td>Real time factor</td>
<td>90 [sec] – [%]</td>
<td>107 [sec] – [%]</td>
</tr>
</tbody>
</table>
Conclusions: Word Graph Generation and Rescoring

Word graph generation:
- Specification of the word graph problem.
- Word pair approximation.
- Extension of word conditioned one-pass algorithm.
- Modification: bookkeeping only.

Word graph (language model) rescoring:
- Experimental results for the 20k WSJ/NAB task (within-word modeling): 4 to 25 word arcs per spoken word sufficient.
- Comparison with integrated search: negligible degradation.
- Note: only ‘forward’ pruning using bigram LM.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
      4.7 Confidence Measures
      4.8 Bayes Risk Decoding
      4.9 System Combination
      4.10 Experiments
      4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Path probabilities over word graphs

Word graph $L$:
- promising part of the search space
- probability along a path is $p(W, X), \ W \in \Sigma^*$
- path posterior probability

$$p(W|X) = \frac{p(W, X)}{p(X)} = \frac{p(W, X)}{\sum_{V \in \Sigma^*} p(V, X)}$$

Example:

$$W_1 = \text{I like this meal} \quad p(W_1, X) = 0.2 \quad p(W_1|X) = 0.4$$
$$W_2 = \text{I like this veal} \quad p(W_2, X) = 0.15 \quad p(W_2|X) = 0.3$$
$$W_3 = \text{alike his veal} \quad p(W_3, X) = 0.1 \quad p(W_3|X) = 0.2$$
$$W_4 = \text{alike this veal} \quad p(W_4, X) = 0.05 \quad p(W_4|X) = 0.1$$

$$p(X) = \sum_W p(W, X) = 0.5$$
Path probabilities over word graphs

Posterior probability of path (edge sequence) $e_1^N$ through word graph $L$:

- word graph $L$ is acyclic
- edge $e$ has word label: $i(e) \in \Sigma \cup \{\epsilon\}$, with finite alphabet (vocabulary) $\Sigma$ and empty word $\epsilon$
- word sequence for $e_1^N \in \pi(L)$:
  \[
  W := i(e_1^N) := [i(e_n)]_{n=1}^N
  \]
- edge weight of $e$: $w(e) := p_{AM}(e) \cdot p_{LM}(e)^\beta$, where acoustic score $p_{AM}(e)$, LM score $p_{LM}(e)$ and LM-scale $\beta$
- path probability for $e_1^L \in \pi(L)$
  \[
  p(e_1^N, X) := \prod_{n=1}^N w(e)^\gamma,
  \]
  where usually $\gamma = 1/\beta$
Edge probabilities over word graphs

Posterior probability of edge \( e \) in word graph \( L \):

- probability that a path goes through edge \( e \)

\[
p(e|X) := \sum_{e_1^N \in \pi(L): \exists n: e_n = e} p(e_1^N|X)
\]

Example:

\[
\begin{align*}
W_1 &= \text{I like this meal} & p(W_1, X) &= 0.2 & p(W_1|X) &= 0.4 \\
W_2 &= \text{I like this veal} & p(W_2, X) &= 0.15 & p(W_2|X) &= 0.3 \\
W_3 &= \text{alike his veal} & p(W_3, X) &= 0.1 & p(W_3|X) &= 0.2 \\
W_4 &= \text{alike this veal} & p(W_4, X) &= 0.05 & p(W_4|X) &= 0.1
\end{align*}
\]

\[
p(e, X) = p(W_1, X) + p(W_2, X) = 0.35 & \quad p(e|X) = 0.7
\]
Forward/Backward Algorithm

**Goal:** find efficient way to calculate edge probabilities.

**Approach:** decompose edge probability into forward and backward part.

\[ p(e, X) = \sum_{e_1^N \in \pi(L) : \exists n: e_n = e} p(e_1^N, X) \]

\[ = \left[ \sum_{e_1^k: \text{to}(e_k) = \text{from}(e)} \prod_{m=1}^{k} w(e_m) \right] \cdot w(e) \cdot \left[ \sum_{e_l^N: \text{to}(e) = \text{from}(e_l)} \prod_{n=1}^{N} w(e_n) \right] \]

\[ \begin{align*}
\text{*: } & e_1^k: \text{ partial path from initial state to state from}(e) \\
\text{*: } & e_l^N: \text{ partial path from state to}(e) \text{ to a final state} \\
\text{*: } & \text{forward score of state } s: \Psi(s) \\
\text{*: } & \text{backward score of state } s: \Phi(s)
\end{align*} \]
Forward/Backward Algorithm

Recursive computation of forward score $\Phi(s)$:

$$\Phi(s) = \sum_{e^k_{1} \text{: } \text{to}(e_k) = s}^{k} \prod_{m=1}^{k} w(e_m)$$

$$= \sum_{e \in \text{In}(s)} w(e) \sum_{e^{l}_{1} \text{: } \text{to}(e') = \text{from}(e)}^{l} \prod_{n=1}^{l} w(e'_n)$$

Notation:
- $\text{In}(s)$: set of edges ending in state $s$
- $\text{from}(e), \text{to}(e)$: predecessor and successor state of edge $e$
- Efficient computation via dynamic programming
- Let $s_F$ be the unique final state:

$$\Phi(s_F) = \sum_{e^N_1 \in \pi(L)} p(e^N_1, X) = p(X)$$
Forward/Backward Algorithm

Recursive computation of backward score \( \Psi(s) \):

\[
\Psi(s) = \sum_{e_I^N: \text{from}(e_I) = s} \prod_{n=1}^{N} w(e_n)
\]

\[
= \sum_{e \in \text{Out}(s)} w(e) \sum_{e'_k^N: \text{from}(e'_k) = \text{to}(e)} \prod_{m=1}^{k} w(e'_m)
\]

\[\Rightarrow \Psi(\text{to}(e))\]

Notation:
- \( \text{Out}(s) \): set of edges leaving state \( s \)
- \( \text{from}(e), \text{to}(e) \): predecessor and successor state of edge \( e \)
- Efficient computation via dynamic programming
- Let \( s_I \) be the unique initial state:

\[
\Psi(s_I) = \sum_{e_1^N \in \pi(L)} p(e_1^N, X) = p(X)
\]
Forward/Backward Algorithm

Implementation in pseudo code:

\[
\Phi(s_I) := 1.0 \\
\text{for state } s \text{ in } L \text{ in topological order:} \\
\quad \Phi(s) := 0.0 \\
\quad \text{for edge } e \text{ in } \text{In}(s): \\
\quad \quad \Phi(s) += \Phi(\text{from}(e)) \cdot w(e) \\
\Psi(s_F) := 1.0 \\
\text{for state } s \text{ in } L \text{ in reversed topological order:} \\
\quad \Psi(s) := 0.0 \\
\quad \text{for edge } e \text{ in } \text{Out}(s): \\
\quad \quad \Psi(s) += w(e) \cdot \Psi(\text{to}(e)) \\
\text{assert } \Phi(s_F) = \Psi(s_I) \\
p(X) := \Phi(s_F) \\
\text{for edge } e \text{ in } L: \\
p(e|X) := \left[ \Phi(\text{from}(e)) \cdot w(e) \cdot \Psi(\text{to}(e)) \right] / p(X)
\]

Complexities:

- loop over [reverse] topological order of acyclic graph \( L \): \( \mathcal{O}(|E(L)|) \)
- posterior probability for each edge in graph \( L \): \( \mathcal{O}(|E(L)|) \)
Generalization: Forward/Backward over Semirings

Semiring \((\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})\):

- set \(\mathbb{K}\)
- binary operation \(\oplus\) called addition
- binary operation \(\otimes\) called multiplication
  1. \((\mathbb{K}, \oplus, \bar{0})\) is a commutative monoid
  2. \((\mathbb{K}, \otimes, \bar{1})\) is a (commutative) monoid
      (commutativity required for backward scores)
  3. multiplication distributes over addition:
      \(x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)\)
      \((x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)\)
  4. \(\bar{0}\) is an annihilator for \(\otimes\):
      \(\bar{0} \otimes x = x \otimes \bar{0} = \bar{0}\)

- forward/backward algorithm can be applied to any commutative semiring
- interpretation of result depends on semiring
Generalization: Forward/Backward over Semirings

<table>
<thead>
<tr>
<th>Semiring</th>
<th>$\mathbb{K}$</th>
<th>$x \oplus y$</th>
<th>$x \otimes y$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>$\mathbb{R}^+$</td>
<td>$x + y$</td>
<td>$x \cdot y$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>log</td>
<td>$\mathbb{R} \cup {+\infty}$</td>
<td>$- \log (e^{-x} + e^{-y})$</td>
<td>$x + y$</td>
<td>$+\infty$</td>
<td>0</td>
</tr>
<tr>
<td>tropical</td>
<td>$\mathbb{R} \cup {+\infty}$</td>
<td>$\min(x, y)$</td>
<td>$x + y$</td>
<td>$+\infty$</td>
<td>0</td>
</tr>
</tbody>
</table>

- probability semiring
  - $\text{fb}(e, X)$: edge probability $p(e, X)$
- log semiring
  - $\text{fb}(e, X)$: probability in negated log-space $- \log p(e, X)$
  - numerical stability
- tropical semiring
  - $\text{fb}(e, X)$: probability of best path through $e$
  - Viterbi approximation
  - implements single-source shortest paths/distances algorithm (Bellman-Ford)
  - forward-backward graph pruning possible/useful (generalization).
Generalization: Forward/Backward over Semirings

Expectation Semiring

**Goal:** enable efficient computation of expectations.

- **Set** $\mathbb{R}^+ \times \mathbb{R}$: elements called “probability” and “value”
- **Addition:** $(p_1, v_1) \oplus (p_2, v_2) := (p_1 + p_1, v_1 + v_2)$
- **Multiplication:** $(p_1, v_1) \otimes (p_2, v_2) := (p_1 \cdot p_2, p_1 \cdot v_2 + p_2 \cdot v_1)$
- **Define cost:**
  - Cost for each path through $e_1^N \in \pi(L)$
  - Cost is additive along paths, i.e. $c(e_1^N) = \sum_{n=1}^{N} c(e_n)$
  - Initialize value component as $w(e)[v] := w(e)[p] \cdot c(e)$
- **forward/backward computation on graph $L$**
  - $p$-component of expectation semiring realizes probability semiring:
    $fb(e, X)[p] = p(e, X)$: probability for paths through $e$
  - $fb(e, X)[v]/fb(e, X)[p]$: expected cost for paths through $e$
  - Expected cost for paths through the graph:
    $$\Phi(s_F)[v]/\Phi(s_F)[p] = \Psi(s_I)[v]/\Psi(s_I)[p] = \sum_{W \in \Sigma^*} p(W|X) c(W)$$
Expectation Semiring

Efficient computation of expectation for graph $L$:

- graph $L[p]$ stores probabilities on edges
- graph $L[c]$ stores costs on edges
- $L[p]$ and $L[c]$ are identical apart from edge weights

$$
\mathbb{E}_{L[p]}L[c] := p(X)^{-1} \sum_{e_1^N \in \pi(L)} w(e_1^N)[p] w(e_1^L)[c]
$$

$$
= \sum_{e_1^N \in \pi(L)} p(e_1^N | X) c(e_1^N)
$$

$$
= \sum_{e \in E(L)} p(e | X) c(e)
$$

- proof: exercise
Frame-wise word posteriors over word graphs

Posterior probability of word $w$ at time $t$ in word graph $L$:

- probability that word $w$ is observed at time $t$ given $X$

$$p_t(w|X) := \sum_{e_1^N \in \pi(L)} p(e_1^N|X) \quad \text{with } t_b(e) \text{ and } t_e(e) \text{ being the start and end times of edge } e.$$
Frame-wise word posteriors over word graphs

\[ p_t(\text{"this"} | X) = p(e_1 | X) + p(e_2 | X) \]
\[ = p(W_1 | X) + p(W_2 | X) + p(W_4 | X) = 0.4 + 0.3 + 0.1 = 0.8 \]
Frame-wise word posteriors over word graphs

Frame-wise word posterior probabilities from graph $L$ given $X$

- probability that word $w$ is observed at time $t$ given $X$

$$p_t(w|X) := \sum_{e_1^N \in \pi(L)} p(e_1^N|X) = \sum_{e \in E(L)} p(e|X)$$

- defines a probability distribution for each time frame

- applications:
  - confidence measures
  - graph decoding
  - acoustic model training
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Confidence Measures

- How confident is a recognizer about the correctness of a hypothesized word?

- Possible approach: confidence measure for word $w$ occurring at position $n$ in the hypothesis:

  $$ \text{conf}(w, n) := p_n(w|X) $$

- Problems:
  - What is the “position” $n$?
  - How to compute a position for a word (and edge) in a graph?
    - Theoretically given by Levenshtein alignment.
  - Ignores deletions:
    Introduce confidence that no word occurs at position $n$, i.e. $p_n(\epsilon|X)$.

- Resort: Avoid explicit word positions by using frame-wise word posterior probabilities.
Confidence Measures

Confidence measures based on frame-wise word posterior probabilities:

- **Idea**: approximate position via time segment.
- Average frame-wise word posterior for a graph edge $e$:

  $$\text{conf}(e) := d(e)^{-1} \sum_{t=t_b(e)}^{t_e(e)-1} p_t(i(e)|X),$$

  with the duration $d(e)$ of edge $e$.
- Maximum approximation:

  $$\text{conf}(e) := \max_{t_b(e) \leq t < t_e(e)} p_t(i(e)|X)$$
Confidence Measures

Posterior probability of word \( w \) at position \( s \) in word graph \( L \):
- assign edges to positions via mapping \( g : E(L) \rightarrow \mathbb{N} \)

\[
p_n(w|X) := \sum_{e_1^N \in \pi(L)} p(e_1^N|X)
\]

\( \exists m : g(e_m) = n \land i(e_m) = w \)

\( p_n("this"|X) \)
Confidence Measures
Position-Wise Word Posteriors over Word Graphs

\[ p_{n}("\text{this}\)|X) \]

\[ W_1 = \text{like this meal} \quad p(W_1, X) = 0.2 \quad p(W_1|X) = 0.4 \]
\[ W_2 = \text{like this veal} \quad p(W_2, X) = 0.15 \quad p(W_2|X) = 0.3 \]
\[ W_3 = \text{alike his veal} \quad p(W_3, X) = 0.1 \quad p(W_3|X) = 0.2 \]
\[ W_4 = \text{alike this veal} \quad p(W_4, X) = 0.05 \quad p(W_4|X) = 0.1 \]
\[ p_{n=3}("\text{this}\)|X) = p(W_1|X) + p(W_2|X) + p(W_4|X) = 0.4 + 0.3 + 0.1 = 0.8 \]
Confidence Measures

Position-wise word posteriors over word graphs

Consider mapping \( g : E(L) \to \mathbb{N} \) to compute position-wise word posterior probabilities from graph \( L \) given \( X \):

- **Monotonicity:** for all graph paths \( e_1^N \) require:
  \[
  g(e_n) < g(e_{n+j}) \quad \text{for } j > 0
  \]

- Probability that word \( w \) is observed at position \( n \) given \( X \) reduces to sum over edge posterior probabilities (\( \rightarrow \) forward-backward)
  \[
  p_n(w|X) := \sum_{e_1^N \in \pi(L): \exists m: g(e_m) = n \land i(e_m) = w} p(e_1^N|X) = \sum_{e \in E(L): \rho(e) = n \land i(e) = w} p(e|X)
  \]

- Defines a probability distribution for each position, where
  \[
  p_n(\epsilon|X) := 1 - \sum_{w \in \Sigma} p_n(w|X)
  \]

- How to obtain \( g : E(L) \to \mathbb{N} \)?
  Will be discussed later, assumed to be given for now.
Confidence Measures

Confidence measures based on position-wise word posteriors:

- Use position given by \( g : E(L) \rightarrow \mathbb{N} \)

\[
\text{conf}(i(e), g(e)) := p_{g(e)}(i(e) | X)
\]

- **Note:** provides confidence for deletions
  - \( e_1^N \) is best scoring graph path.
  - confidence for deletions between \( e_m \) and \( e_{m+1} \):

\[
\text{conf}(\epsilon, n) := \begin{cases} 
  p_n(\epsilon | X) & \text{for } g(m) < n < g(m + 1) \\
  0 & \text{otherwise}
\end{cases}
\]

- **Compact representation of position-wise word posteriors is the confusion network.**
Confidence Measures

Confusion networks from word graphs

Confusion network (CN) derived from word graph $\mathbf{L}$:

- slot-wise posterior probability distribution

$$p_n(w|X) := \sum_{e_1^N \in \pi(L)} p(e_1^N|X), \quad p_n(\varepsilon|X) := 1 - \sum_{w \in \Sigma} p_n(w|X)$$
Confidence Measures

Confusion networks from word graphs

\[
p_1(\text{"I"}|X) = 0.7 \quad p_2(\text{"like"}|X) = 0.7 \quad p_3(\text{"this"}|X) = 0.8 \quad p_4(\text{"veal"}|X) = 0.6
\]

\[
p_1(\epsilon|X) = 0.3 \quad p_2(\text{"alike"}|X) = 0.3 \quad p_3(\text{"his"}|X) = 0.2 \quad p_4(\text{"meal"}|X) = 0.4
\]
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Bayes Risk Decoding

- Viterbi decoding minimizes sentence error, but:
- Standard evaluation metric is word error (defined by Levenshtein distance)
- Question: How to do better than Viterbi decoding?
- Bayes risk decoding with Levenshtein distance yields minimum expected word error
Bayes Risk Decoding

Motivation

| Word sequence $W$                  | $p(W|X)$ |
|-----------------------------------|----------|
| I like this meal                   | 0.4      |
| I like this veal                   | 0.3      |
| alike his veal                     | 0.2      |
| alike this veal                    | 0.1      |

$\hat{W} := \hat{w}_1^N \in [\Sigma \cup \{\epsilon\}]^N$ is hypothesis

- expected error for position $n$ is $(1 - p_n(\hat{w}_n|X))$
- expected error for $\hat{W}$ is $\sum_{n=1}^{N} (1 - p_n(\hat{w}_n|X))$
- expected error of Viterbi hypothesis “I like this meal”: $0.3 + 0.3 + 0.2 + 0.6 = 1.4$
- minimum expected error hypothesis “I like this veal”: $0.3 + 0.3 + 0.2 + 0.4 = 1.2$
Bayes Risk Decoding

Confusion network decoding

| W           | $p(W|X)$ |
|-------------|----------|
| I like this meal | 0.4     |
| I like this veal | 0.3     |
| alike his veal  | 0.2     |
| alike this veal | 0.1     |

Confusion network with position posterior probabilities $p_n(w|X)$:

- minimum expected error hypothesis: $\hat{W} = [\arg\max p_n(w|X)]_{n=1}^{N}$
- equals decoding of CN
- CN decoding aims at minimizing a word-based error
- Viterbi decoding aims at minimizing sentence error
- we are interested in minimized word error (defined by the Levenshtein distance)
Bayes Risk Decoding

Bayes risk decoding with loss function \( L : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R} \)

\[
\hat{W} := \arg \min_{W \in \Sigma^*} \sum_{V \in \Sigma^*} p(V|X) L(W, V)
\]

- \( \hat{W} \) has the minimum expected loss
- sentence error \( \rightarrow \) MAP (or Viterbi) decoding

\[
L(W, V) := 1 - \delta(W, V)
\]

- Levenshtein distance

\[
L(W, V) := \text{Lev}(W, V)
\]

- loss function of choice for speech recognition evaluation
- non-local dependencies
- exact computation prohibitive on graphs
- use approximation \( \rightarrow \) only local dependencies

- graph-based Bayes risk decoder
  - paths through graph instead of \( W \in \Sigma^* \)
Graph-Based Bayes Risk Decoding

Hypothesis and summation space

- summation space graph $S$ defines posterior distribution $p(W|X)$ for $W \in \Sigma^*$
- $S$ is given by word graph from HMM decoder
- hypothesis space graph $H$ defines set of possible decoding outcomes $\hat{W}$
- for unknown posterior distribution
  - $H$ usually a superset of $S$
  - $H$ depends on loss function
  - sentence error $\rightarrow H = S$

\[
\hat{W} := \arg \min_{e_1^N \in \pi(H)} \sum_{f_1^M \in \pi(S)} p(f_1^M|X) L(e_1^N, f_1^M)
\]

efficient computation, if $L(e_1^N, f_1^M)$ has only local dependencies
Local Loss Functions

- for $e_1^N \in \pi(H)$ and $f_1^M \in \pi(S)$

$$L(e_1^N, f_1^M) = \sum_{n=1}^{N} \sum_{m=1}^{M} L(e_n, f_m)$$

- word-wise loss computation
- only local dependencies
Bayes Risk Decoding with Local Loss Functions

Bayes decision rule with local loss function:

$$\hat{W} := \arg\min_{e_1^N \in \pi(H)} \sum_{f_1^M \in \pi(S)} p(f_1^M | X) \sum_{n=1}^{N} \sum_{m=1}^{M} L(e_n, f_m)$$

$$= \arg\min_{e_1^N \in \pi(H)} \sum_{n=1}^{N} \sum_{f \in E(S)} p(f | X) L(e_n, f)$$

$$:= c(e_n, S)$$

- efficient computation
  1. rescore each arc in $e \in E(H)$ with $c(e, S)$
  2. apply Viterbi decoder to $H$

- quadratic runtime: $O(|E(H)||E(S)|)$

What are good local loss functions?
Levenshtein Distance Approximations

- Local loss function shall approximate Levenshtein distance
- Frame error based loss functions
  - Count errors on a per-frame base
  - Normalize frame error on a per-word base
  - No alignment problem
- CN distance
  - CN defines alignment between any two paths in a graph
  - Compute distance based on the CN alignment
  - Levenshtein distance is lower bound for CN distance

For both types of loss functions:
Experiments show strong correlation between exact word error and word error approximation
Frame Error

**Idea:** compare words frame-wise using word boundaries from time-alignment.

**Example:**

<table>
<thead>
<tr>
<th>reference:</th>
<th>I</th>
<th>like</th>
<th>this</th>
<th>meal</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypothesis:</td>
<td>alike</td>
<td>this</td>
<td>meal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time frame:</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame errors</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Frame error (FE) definition:**

\[ L(e^N_1, f^M_1) := \sum_{t=1}^{T} (1 - \delta(i(e_t), i(f_t))) \]

- \( e_t, f_t \): edges (from \( e^N_1, f^M_1 \) which intersect with time frame \( t \), i.e.
  - \( e_t \in e^N_1 \mid t_b(e_t) \leq t < t_e(e_t) \)
Frame Error Based Loss Functions

Define:
- \( d(e) \): duration in number of time frames
- \( o(e, f) \): overlap in number of time frames
- \( h(e, f) \): overlap conditioned on label equality

Rewrite frame error:

\[
L(e_1^N, f_1^M) := \sum_{t=1}^{T} (1 - \delta(i(e_t), i(f_t)))
\]

\[
= \sum_{n=1}^{N} \left[ d(e_n) - \sum_{m=1}^{M} o(e_n, f_m) \cdot \delta(i(e_n), i(f_m)) \right] := h(e_n, f_m)
\]

\[
= \sum_{m=1}^{M} \left[ d(f_m) - \sum_{n=1}^{N} h(e_n, f_m) \right]
\]
Frame Error Based Loss Functions

Rescoring function for (position-wise) Bayes risk decoder:

\[ c(e, S) := \sum_{f \in E(S)} p(f|X) L(e, f) \]
\[ = d(e) - \sum_{f \in E(S)} p(f|X) h(e, f) \]
\[ = \sum_{t=t_b(e)}^{t_e(e)-1} [1 - p_t(i(e)|X)] \]

- simple rescoring with frame-wise word posterior probabilities
Frame Error Based Loss Functions

Rescoring function for (position-wise) Bayes risk decoder:

\[
c(e, S) = \sum_{t = t_b(e)}^{t_e(e)-1} \left[ 1 - p_t(i(e)|X) \right]
\]

Discrepancy between word and frame error:

- long words have higher impact in frame error based decision i.e. the corresponding frame error generally increases with word length
- idea: normalize frame error on a per-word basis
- open: normalize per hypothesis or per reference word?
Hypothesis-Side Normalized Frame Error

Rescoring function for Bayes risk decoder with frame error normalized by hypothesis duration:

\[
\hat{W} := \arg \min_{e_1^N \in H} \sum_{n=1}^{N} \left[ 1 - \sum_{f \in S} p(f|X) \frac{h(e_n, f)}{d(e_n)} \right]
\]

\[
= \arg \min_{e_1^N \in H} \sum_{n=1}^{N} \left[ 1 - \frac{1}{d(e_n)} \sum_{t=t_b(e)}^{t_e(e)-1} p_t(i(e)|X) \right]
\]

\[:= c(e_n, S)\]

- normalize FE for each edge \(e \in H\) by \(d(e)\)
- normalized sum of frame-wise word posterior probabilities
- high number of deletions
Reference-Side Normalized Frame Error

Rescoring function for Bayes risk decoder with frame error normalized by reference duration:

\[
\hat{W} := \arg \min_{e_1^N \in H} \sum_{f \in S} p(f|X) \left[ 1 - \sum_{n=1}^{N} \frac{h(e_n, f)}{d(f)} \right]
\]

\[
= \arg \min_{e_1^N \in H} \left[ \sum_{f \in S} p(f|X) - \sum_{f \in S} \sum_{n=1}^{N} \frac{h(e_n, f)}{d(f)} p(f|X) \right]
\]

\[
= \arg \min_{e_1^N \in H} \sum_{n=1}^{N} \left[ - \sum_{f \in S} p(f|X) \frac{h(e_n, f)}{d(f)} \right]
\]

\[
:= c(e_n, S)
\]

- normalize FE for each edge \( f \in S \) by \( d(f) \)
- normalization → cannot use frame-wise word posteriors
- high number of insertions
Bias in Single-Sided Frame Error Normalization

- hypothesis-side normalization does not penalize deletions
- reference-side normalization does not penalize insertions
- best approach in practice: average both normalizations

Example:

<table>
<thead>
<tr>
<th>reference:</th>
<th>I</th>
<th>like</th>
<th>satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypothesis:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>alike</td>
<td>that</td>
<td>infection</td>
</tr>
<tr>
<td>error normalized by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. reference</td>
<td>1.0</td>
<td>1.0</td>
<td>0.4</td>
</tr>
<tr>
<td>2. hypothesis</td>
<td>0.2</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>average of 1. + 2.</td>
<td>0.6</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Hypothesis Space for Minimum Frame Error Search

- Frame error computation requires word boundaries in $H$
- Derive $H$ from summation space graph $S$
  - Build time-conditioned graph from $S$: merge all states with equal time stamps
  - $\pi(H) \supseteq \pi(S)$

Example:

Original lattice:

```
0  3  10  14  24
I  like  this  meal
```

Time-conditioned form:

```
0  3  10  11  14  24
I  like  this  meal
```

Schütter/Ney: Advanced ASR 339 August 5, 2010
Confusion Network Distance

- recall: a confusion network $CN(L)$ is derived from graph $L$ by a function

$$g : E(L) \rightarrow \mathbb{N},$$

which maps each edge to a unique position, while keeping the order of each path contained: $g(e_n) < g(e_{n+1})$ for $e_1^N \in \pi(L)$

- $g(\cdot)$ maps each path in $L$ to a path in $CN(L)$

- each path in $CN(L)$ has equal length $N := \max_{e \in E(L)} g(e)$

- $CN(L)$ defines a distance between any two paths $e_1^N$ and $f_1^N$ through the confusion network mapping to positions:

$$L(e_1^N, f_1^N) := \sum_{n=1}^{N} [1 - \delta(i(e_n), i(f_n))]$$
Confusion Network Distance

Recall Bayes decision rule with local loss function:

\[
\hat{W} := \arg \min_{e_1^N \in \pi(H)} \sum_{f_1^M \in \pi(S)} p(f_1^M | X) \sum_{n=1}^N \sum_{m=1}^M L(e_n, f_m)
\]

- problem: confusion network in general does not allow to keep original path probabilities
- idea: define loss function between paths from confusion network and its source graph, where the latter contains the original path probabilities
- approach: use confusion network as Bayes risk decoding hypothesis space $H$ only, whereas for summation the original graph $S$ is used
- let $g(\cdot)$ be defined for two graphs $H$ and $S$
  \[ \Rightarrow g(\cdot) \text{ defines a loss function } L : \pi(H) \times \pi(S) \rightarrow \mathbb{R} \]
- note: loss function does not remain local in the above form, but does retain its efficiency.
Hypothesis Space for CN Decoding

- summation space $S$ given by word graph from HMM decoder
- let confusion network mapping $g : E(S) \to [1, N]$ be given
- construction of hypothesis space $H$:
  1. initialize $N + 1$ states; state 0 is the initial and $N$ the final state
  2. foreach state $s \in [1, N)$:
     - foreach $w \in \Sigma \cup \{\epsilon\}$:
       - connect state $s$ with state $s + 1$ with edge $e$, where $i(e) = w$ and $g(e) = s + 1$
- $H$ has CN-structure
  - any path $e_1^N \in \pi(H)$ has exactly $N$ arcs
  - a path $e_1^N \in \pi(H)$ has no gaps, i.e. $g(e_s) = s$
- $H$ contains all possible Bayes risk decoding results for the CN distance as loss function
- further investigations make use of special structure of $H$; minimum CN distance decoding with an arbitrary hypothesis space is left as an exercise
CN Distance for Graphs

CN distance between \( e_1^N \in H \) and \( f_1^M \in S \):

- **substitutions:**
  - hypothesis and reference word, but hypothesis word \( \neq \) reference word

\[
L_S(e_1^N, f_1^M) := \sum_{n=1}^{N} \sum_{m=1}^{M} \delta(g(e_n), g(f_m)) \cdot [1 - \delta(i(e_n), \epsilon)] \\
\cdot [1 - \delta(i(f_m), \epsilon)] \cdot [1 - \delta(i(e_n), i(f_m))]
\]

- **deletions:**
  - no hypothesis word, but reference word

\[
L_D(e_1^N, f_1^M) := \sum_{n=1}^{N} \sum_{m=1}^{M} \delta(g(e_n), g(f_m)) \cdot \delta(i(e_n), \epsilon) \\
\cdot [1 - \delta(i(f_m), \epsilon)]
\]
CN Distance for Graphs

CN distance between $e_1^N \in H$ and $f_1^M \in S$ (ctd.):

- insertions:
  - hypothesis word, but no reference word
  - problem: position labels for $f_1^M$ are not strictly consecutive, gaps!
  - indirect: $\# \text{ins} = \#(\text{hyp. words}) - \#(\text{cor and sub})$

\[
L_1(e_1^N, f_1^M) := \left[ \sum_{n=1}^{N} \left[ 1 - \delta(i(e_n), \epsilon) \right] \right] - \left[ \sum_{n=1}^{N} \sum_{m=1}^{M} \delta(g(e_n), g(f_m)) \right] \\
\cdot \left[ 1 - \delta(i(e_n), \epsilon) \right] \cdot \left[ 1 - \delta(i(f_m), \epsilon) \right]
\]
CN Distance for Graphs

CN distance: sum of four costs:

\[ L(e_1^N, f_1^M) := \sum_{n=1}^{N} \left[ 1 - \delta(i(e_n), \epsilon) \right] \]

\[ + \sum_{n=1}^{N} \sum_{m=1}^{M} \delta(g(e_n), g(f_m)) \cdot \]

\[ \quad \cdot \left\{ - \left[ 1 - \delta(i(e_n), \epsilon) \right] \cdot \left[ 1 - \delta(i(f_m), \epsilon) \right] \right. \]

\[ + \left[ 1 - \delta(i(e_n), \epsilon) \right] \cdot \left[ 1 - \delta(i(f_m), \epsilon) \right] \cdot \left[ 1 - \delta(i(e_n), i(f_m)) \right] \]

\[ + \delta(i(e_n), \epsilon) \cdot \left[ 1 - \delta(i(f_m), \epsilon) \right] \right\} \]

\[ =: l(e_n, f_m) \]

- consider local sum of Kronecker delta expressions \( l(e, f) \)
Simplify local sum of deltas:

\[
l(e, f) = - \left[1 - \delta(i(e), \epsilon)\right] \cdot \left[1 - \delta(i(f), \epsilon)\right] \\
+ \left[1 - \delta(i(e), \epsilon)\right] \cdot \left[1 - \delta(i(f), \epsilon)\right] \\
\cdot \left[1 - \delta(i(e), i(f))\right] \\
+ \delta(i(e), \epsilon) \cdot \left[1 - \delta(i(f), \epsilon)\right] \\
= - \left[1 - \delta(i(e), \epsilon)\right] \cdot \left[1 - \delta(i(f), \epsilon)\right] \cdot \delta(i(e), i(f)) \\
+ \delta(i(e), \epsilon) \cdot \left[1 - \delta(i(f), \epsilon)\right] \\
= - \left[1 - \delta(i(e), \epsilon)\right] \cdot \delta(i(e), i(f)) \\
+ \delta(i(e), \epsilon) \cdot \left[1 - \delta(i(f), \epsilon)\right]
\]
CN Distance for Graphs

Resubstitute local sum of deltas into CN distance:

\[
L(e_1^N, f_1^M) = \sum_{n=1}^{N} \left[ 1 - \delta(i(e_n), \epsilon) \right] + \sum_{m=1}^{M} \delta(g(e_n), g(f_m)) l(e_n, f_m) \]

\[
= \sum_{n=1}^{N} \left\{ L_1(e_n) + \sum_{m=1}^{M} L_2(e_n, f_m) \right\} \]

alternative local loss function

Corresponding decision rule:

\[
\hat{W} := \text{arg min}_{e_1^N \in \pi(H)} \sum_{f_1^M \in \pi(S)} p(f_1^M | X) \sum_{n=1}^{N} \left\{ L_1(e_n) + \sum_{m=1}^{M} L_2(e_n, f_m) \right\} \]

\[
= \text{arg min}_{e_1^N \in \pi(H)} \sum_{n=1}^{N} \left\{ L_1(e_n) + \sum_{f \in E(S)} p(f | X) L_2(e_n, f) \right\} \]

\[
= \text{arg min}_{e_1^N \in \pi(H)} R(e_1^N | X) \]
CN Decoding

\[ R(e_1^N | X) = \sum_{n=1}^{N} \left\{ L_1(e_n) + \sum_{f \in E(S)} p(f|X)L_2(e_n, f_m) \right\} \quad \text{(posterior risk)} \]

\[ = \sum_{n=1}^{N} \left\{ 1 - \delta(i(e_n), \epsilon) + \sum_{f \in E(S)} p(f|X) \delta(g(e_n), g(f)) \cdot \left[ \delta(i(e_n), \epsilon) \cdot [1 - \delta(i(f), \epsilon)] - [1 - \delta(i(e_n), \epsilon)] \cdot \delta(i(e_n), i(f)) \right] \right\} \]

\[ = \sum_{n=1}^{N} \left\{ 1 - \left[ 1 - \delta(i(e_n), \epsilon) \right] \sum_{f \in E(S)} p(f|X) \delta(g(e_n), g(f)) \cdot \delta(i(e_n), i(f)) \right. \]

\[ \left. - \delta(i(e_n), \epsilon) \left[ 1 - \sum_{f \in E(S)} p(f|X) \delta(g(e_n), g(f)) \left[ 1 - \delta(i(f), \epsilon) \right] \right] \right\} \]

\[ = \sum_{n=1}^{N} \left[ 1 - p_n(i(e_n)|X) \right] + \sum_{n=1}^{N} \left[ 1 - p_n(\epsilon|X) \right] = \sum_{n=1}^{N} \left[ 1 - p_n(i(e_n)|X) \right] \]
CN Decoding

Bayes risk decoding:

\[ \hat{W} = \arg \min_{e_1^N \in \pi(H)} R(e_1^N | X) \]

\[ = \arg \min_{e_1^N \in \pi(H)} \sum_{n=1}^{N} [1 - p_n(i(e_n)|X)] \]

\[ = \left[ \arg \max_{w \in \Sigma \cup \{\epsilon\}} p_n(w|X) \right]_{n=1}^{N} \]

with

\[ p_n(w|X) := \sum_{f \in E(S)} p(f|X) \cdot \delta(n, g(f)) \cdot \delta(w, i(f)) \]

\[ p_n(\epsilon|X) := 1 - \sum_{f \in E(S)} p(f|X) \cdot \delta(n, g(f)) \cdot [1 - \delta(i(f), \epsilon)] \]

\[ = 1 - \sum_{w \in \Sigma} p_n(w|X) \]
Bayes risk decoding:

\[ \hat{W} = \left \{ \arg \max_{w \in \Sigma \cup \{\epsilon\}} p_n(w|X) \right \}_{n=1}^{N} \]

- decoding only requires CN derived from S (valid only for the special construction of H)
- gaps in the position labels are handled by the derived posterior probabilities for the empty word \( p_n(\epsilon|X) \)
CN Distance vs. Levenshtein Distance

- CN alignments cannot always represent Levenshtein alignments
- CN distance is upper bound to Levenshtein distance

Example:

Given:
1. \texttt{a b a b a}
2. \texttt{b a b a c}
3. \texttt{a b a c a}

<table>
<thead>
<tr>
<th></th>
<th>\textbf{Levenshtein}</th>
<th>\textbf{CN}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\texttt{a b a b a}</td>
<td>\texttt{a b a b a}</td>
</tr>
<tr>
<td>2</td>
<td>\texttt{b a b a c}</td>
<td>\texttt{b a b a c}</td>
</tr>
<tr>
<td>3</td>
<td>\texttt{a b a c a}</td>
<td>\texttt{a b a c a}</td>
</tr>
</tbody>
</table>

Lev(1, 2) = 2  Lev(1, 3) = 1  Lev(2, 3) = 2
CN(1, 2) = 2  CN(1, 3) = 1  CN(2, 3) = 3
CN Construction

- CN construction for graphs: find $g : E(L) \rightarrow \mathbb{N}$
- $g(\cdot)$ induces the Bayes risk decoder

$$\text{CN}(X; g) := \arg \min_{e_1^N \in H} \sum_{f_1^M} p(f_1^M | X) L(e_1^N, f_1^M; g)$$

- goal: find $g(\cdot)$ such that the error on a training set $[\tilde{W}, X]_{r=1}^R$ is minimized

$$g_{\text{opt}} := \sum_{r=1}^R \text{Lev}(\tilde{W}_r, \text{CN}(X_r; g))$$

- other optimization criteria possible, e.g. choose $g(\cdot)$ such that the Bayes risk is minimized
- how to compute $g_{\text{opt}}$ for LVCSR graphs?
  - heuristic 1: $N$-best lists, align entries to Viterbi path (pivot element)
  - heuristic 2: cluster the edges in $S$
Edge Clustering Algorithm

An edge clustering algorithm

▶ idea: use a fixed set of clusters
  ▶ use set of pivot edges to initialize edge clusters
  ▶ add more pivot edges, if not all edges can be clustered
▶ all edges in a cluster must overlap in time
  ▶ sort clusters by average edge start time
  ▶ \( g(e_n) < g(e_{n+1}) \) holds for each \( e_1 \in \pi(S) \)
▶ define appropriate distance function between edge \( e \) and edge cluster \( E \); usually based on
  ▶ time overlap
  ▶ word equality
  ▶ pronunciation similarity
▶ weight distance for edge \( e \) with \( p(e|X) \)
  ⇒ clustering is driven by likely edges
Edge Clustering Algorithm

Algorithm outline:

initialize set of pivot edges \( P \leftarrow \emptyset \)
initialize remaining edge set \( R = E(S) \)
while \( R \) is not empty
  choose set of pivot edges \( P_R \) from \( R \)
  \( P \leftarrow P \cup P_R \)
reinitialize remaining edge set \( R \leftarrow \emptyset \)
initialize edge clusters from pivot edges
for \( e \in E(S) \setminus P \) sorted by weighted cluster distance:
  \( C \leftarrow \) nearest edge cluster of \( e \)
  if \( e \) and \( e' \) overlap
    foreach \( e' \in C \):
      assign \( e \) to \( C \)
  else:
    assign \( e \) to \( R \)

▷ open: how to find the pivot edges?
Edge Clustering Algorithm

- initialize clusters from
  - none overlapping edges
  - most likely edges

Pivot edge set algorithm outline:

```python
select_pivot_arcs(E):
    initialize pivot edge set $P \leftarrow \emptyset$
    for $e \in E$ sorted by $p(e|X)$:
        if foreach $e' \in P$: $e'$ and $e$ not overlap:
            assign $e$ to $P$
    return $P$
```
Edge Clustering Algorithm

Example:

1. initial pivot arcs: “eh:0-10” “hello:10-40”
2. assign “hello:0-32” to second cluster
3. new pivot pivot arc: “[si]:32-40” (since [si] arc does not overlap)
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
System combination

- observation: different ASR systems lead to different errors
- graphs provided by $J$ systems?
- question: how to minimize word error by combining the $J$ graphs?

- how to compute a single hypothesis from $J$ parallel word graphs?
System Combination

- idea: introduce system as hidden variable

\[ p(W|X) = \sum_{j=1}^{J} p(W|j, X)p(j), \]

assumption: system prior is independent of acoustic \( X \)

\( \Rightarrow \) summation space is given by the graph union

\[ S := \bigcup_{j=1}^{J} L_j \]

\( \Rightarrow \) path probability for a path through \( S \)

\[ p(e^N_1|X) := \sum_{j=1}^{J} p(j)p_j(e^N_1|X), \]

where \( p_j(e^N_1|X) = 0 \) if \( e^N_1 \notin \pi(L_j) \)
Can we reduce graph combination to a single graph problem?

- construct modified graph union $S'$
  1. introduce super initial state $s_0$
  2. foreach system $j$:
     - connect $s_0$ with initial state of $L_j$ with edge $e$,
       where $i(e) = \epsilon$ and $w(e) = p(j)/p_j(X)$
- for a path $e_1^N \in \pi(S')$ holds

\[ p(e_1^N, X) = p(j)p_j(e_1^N|X) = p(e_1^N|X) \]

graph combination $\Leftrightarrow$ Bayes risk decoding of $S'$
System Combination

Alternative approaches:

- Minimum frame error combination
- Confusion Network Combination
- ROVER
Minimum frame error combination

Rescoring function for Bayes risk decoder:

\[ c(e, S) := d(e) - \sum_{f \in E(S)} p(f|X) h(e, f) = \sum_{t=t_b(e)}^{t_e(e)-1} [1 - p_t(i(e)|X)] \]

Extension to system combination:

- compute frame-wise word posteriors as

\[ p_t(w|X) := \sum_{j=1}^{J} p(j)p_{j,t}(w|X) \]

- however: equivalent to computing \( p_t(w|X) \) from \( S' \)!
Recogniser Output Voting Error Reduction (ROVER):

- Levenshtein alignment of individual systems outputs:
  1. select supervisor system (usually with lowest expected WER)
  2. align other hypotheses to supervisor

- system-dependent hypothesis \( \hat{W}_j := \tilde{\omega}_{j,1} \) of jth system

- alignment of \( J \) system-dependent hypotheses

\[
A := (A_1, A_2, \ldots, A_S) \text{ with } A_k := (a_{k,1}, \ldots, a_{k,J})
\]

- interpretation: at alignment position \( s \) the words \( w_{1,a_{s,1}}, w_{2,a_{s,2}}, \ldots, w_{J,a_{s,J}} \) compete,
where \( a_{s,j} = 0 \) indicates the empty word, i.e. \( w_{s,0} := \epsilon \)
ROVER Decoding

ROVER (ctd.):
▶ use word-wise scores
  ▶ majority voting:

\[
c_j(s, w) := \begin{cases} 
  1, & \text{if } w_{j,a_s,j} = w \\
  0, & \text{otherwise}
\end{cases}
\]

▶ confidence voting:

\[
c_j(s, w) := \begin{cases} 
  p_{j,a_s,j}(w|X), & \text{if } a_{s,j} \neq 0 \text{ and } w = w_{j,a_s,j} \\
  p_0, & \text{if } a_{s,j} = 0 \text{ and } w = \epsilon \\
  0, & \text{otherwise}
\end{cases}
\]

where \( p_0 \) constant confidence for a deletion

▶ decode alignment position-wise

\[
\hat{W} = \left[ \arg \max_{w \in \Sigma \cup \{\epsilon\}} \sum_{j=1}^{J} c_j(s, w) \right]_{s=1}^{S}
\]
Confusion Network Combination (CNC)

Confusion Network Combination (CNC):

- Levenshtein alignment of system-dependent CNs
- ROVER vs. CNC:
  - information provided by $j$th system at position $n$

<table>
<thead>
<tr>
<th>word</th>
<th>ROVER</th>
<th>CNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>1</td>
<td>$p_{j,n}(w_1</td>
</tr>
<tr>
<td>$w_2$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

words sorted by $p_n(w|X)$ in decreasing order

- CN provides at each position prob. distribution over all words
- Levenshtein alignment of CNs with local cost defined between prob. distributions
Confusion Network Combination (CNC)

CNC (ctd.)

- alignment between two CNs

\[ A := (A_1, A_2, \ldots) \text{ with } A_s := (a_{s,1}, a_{s,2}) \]

- interpretation: combined position-wise word posterior probability distr. at alignment position \( s \)

\[ p_s(w|X) := \sum_{j=1}^{2} p(j)p_{j,a_{s,j}}(w|X) \]

where \( p_{j,0}(\epsilon|X) := 1 \)

- alignment result is a CN of length \( S \)

\[ c(A) = \sum_{s=1}^{S} \left[ 1 - \max_{w \in \Sigma \cup \{\epsilon\}} p_s(w|X; A) \right] \]

\( \Rightarrow \) minimizing \( c(A) \) \( \Rightarrow \) Bayes decision rule using CN distance
Confusion Network Combination (CNC)

Levensthein alignment of CNs

- align 1st CN and 2nd CN

\[ C(l, k) := \min \begin{cases} 
C(k - 1, l) + c(k, 0), \\
C(k - 1, l - 1) + c(k, l), \\
C(k, l - 1) + c(0, l)
\end{cases} \]

with local cost minimizing the position-wise/local word error:

\[ c(l, k) := 1 - \max_{w \in \Sigma \cup \{\epsilon\}} \left\{ p(1)p_{1,k}(w|X) + p(2)p_{2,l}(w|X) \right\} \]

- align 1+2nd CN and 3rd CN
- ...
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
Experiments

Chinese 230h Test System

- 230h training data (BN/BC)
- ML-trained speaker-normalized models, 1M Gaussians
- 60K vocabulary, 4-gram LM
- two-pass recognition (speaker adaptation, LM-rescoring)
- systems vary in the
  - acoustic front-ends: MFCC, PLP, or Gammatone features
  - phonetic decision tree: randomized CART estimation

Chinese Development and Test Sets

<table>
<thead>
<tr>
<th>name</th>
<th>speech</th>
<th>running words</th>
<th>running chars</th>
<th>vocabulary words</th>
<th>vocabulary chars</th>
</tr>
</thead>
<tbody>
<tr>
<td>gale-dev07</td>
<td>2.5h</td>
<td>27.1K</td>
<td>46.8K</td>
<td>5.2K</td>
<td>1.9K</td>
</tr>
<tr>
<td>gale-eval07</td>
<td>1.6h</td>
<td>-</td>
<td>28.1K</td>
<td>-</td>
<td>1.7K</td>
</tr>
<tr>
<td>gale-dev08</td>
<td>1.0h</td>
<td>10.5K</td>
<td>18.2K</td>
<td>2.9K</td>
<td>1.4K</td>
</tr>
</tbody>
</table>
Experiments

English TC-Star 2007 Evaluation System

- 92h training data (European parliament plenary sessions, EPPS)
- single, globally pooled, diagonal co-variance matrix
- MPE trained speaker normalized models, 1M Gaussians
- 50K vocabulary, 4-gram LM
- two-pass recognition (speaker adaptation, LM-rescoring)
- four systems that differ mainly in
  - system 1: MFCC front-end
  - system 2: Gammatone front-end
  - system 3: MFCC front-end + NN-based phoneme posteriors
  - system 4: MFCC front-end + 190h unsupervised EPPS training data

English EPPS Development and Test Sets

<table>
<thead>
<tr>
<th>name</th>
<th>speech</th>
<th>running words</th>
</tr>
</thead>
<tbody>
<tr>
<td>epps-dev06</td>
<td>3.2h</td>
<td>27.0K</td>
</tr>
<tr>
<td>epps-eval06</td>
<td>3.2h</td>
<td>30.0K</td>
</tr>
<tr>
<td>epps-eval07</td>
<td>2.9h</td>
<td>27.0K</td>
</tr>
</tbody>
</table>
Experiments

English TC-Star 2007 Cross-Site System Combination

- 92h training data (European parliament plenary sessions, EPPS)
- 190h unsupervised training data (EPPS, optional)
- Lattices provided by
  - FBK/IRST, Trento, Italy (IRST)
  - CNRS/LIMSI, Paris, France (LIMSI)
  - RWTH Aachen University, Germany (RWTH)
  - University of Karlsruhe, Germany (UKA)
- If a site does internal system combination, e.g. RWTH Aachen, lattices are from the best single system
Bayes Risk Decoding
Chinese 230h Test System

<table>
<thead>
<tr>
<th>System</th>
<th>Decoder</th>
<th>CER[%]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>dev07</td>
<td>eval07</td>
<td>dev08</td>
</tr>
<tr>
<td>s1</td>
<td>Viterbi</td>
<td>14.54</td>
<td>15.08</td>
<td>13.28</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>14.32</td>
<td>14.95</td>
<td>13.10</td>
</tr>
<tr>
<td>s2</td>
<td>Viterbi</td>
<td>14.82</td>
<td>15.02</td>
<td>13.54</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>14.52</td>
<td>14.74</td>
<td>13.35</td>
</tr>
<tr>
<td>s3</td>
<td>Viterbi</td>
<td>15.07</td>
<td>15.60</td>
<td>13.80</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>14.86</td>
<td>15.42</td>
<td>13.67</td>
</tr>
</tbody>
</table>

1 tuning set, CER: [Chinese] Character Error Rate
<table>
<thead>
<tr>
<th>System</th>
<th>Decoder</th>
<th>WER[%]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>dev06</td>
<td>eval06</td>
<td>eval07</td>
</tr>
<tr>
<td>s1</td>
<td>Viterbi</td>
<td>11.09</td>
<td>8.43</td>
<td>9.81</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>11.97</td>
<td>8.83</td>
<td>10.48</td>
</tr>
<tr>
<td>s2</td>
<td>Viterbi</td>
<td>11.89</td>
<td>8.70</td>
<td>10.07</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>11.87</td>
<td>9.31</td>
<td>11.57</td>
</tr>
<tr>
<td>s3</td>
<td>Viterbi</td>
<td>12.43</td>
<td>8.98</td>
<td>10.76</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>10.73</td>
<td>8.22</td>
<td>9.57</td>
</tr>
<tr>
<td>s4</td>
<td>Viterbi</td>
<td>12.06</td>
<td>9.44</td>
<td>11.73</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>11.42</td>
<td>8.61</td>
<td>9.78</td>
</tr>
</tbody>
</table>

1 tuning set
## Bayes Risk Decoding

### English TC-Star 2007 Cross-Site

<table>
<thead>
<tr>
<th>System</th>
<th>Decoder</th>
<th>WER [%] eval06</th>
<th>WER [%] eval07</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIMSI</td>
<td>Viterbi</td>
<td>8.16</td>
<td>9.13</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>8.07</td>
<td>8.96</td>
</tr>
<tr>
<td>RWTH</td>
<td>Viterbi</td>
<td>8.46</td>
<td>9.71</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>8.24</td>
<td>9.54</td>
</tr>
<tr>
<td>UKA</td>
<td>Viterbi</td>
<td>8.80</td>
<td>10.22</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>8.98</td>
<td>10.36</td>
</tr>
<tr>
<td>IRST</td>
<td>Viterbi</td>
<td>10.09</td>
<td>9.81</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>10.06</td>
<td>9.82</td>
</tr>
</tbody>
</table>

1 tuning set
## Bayes Risk Decoding

### Frame Error vs. CN-Distance

<table>
<thead>
<tr>
<th>System</th>
<th>loss function</th>
<th>frame norm.</th>
<th>CER[%] (del/ins) err</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>dev&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>Chinese</td>
<td>frame err.</td>
<td>hypothesis</td>
<td>(2.92/1.38) 14.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>symmetric</td>
<td>(3.01/0.75) 13.13</td>
</tr>
<tr>
<td></td>
<td>CN-dist.</td>
<td>n.a.</td>
<td>(2.79/1.45) 14.30</td>
</tr>
</tbody>
</table>

### WER[%] (del/ins) err

<table>
<thead>
<tr>
<th>System</th>
<th>loss function</th>
<th>frame norm.</th>
<th>WER[%] (del/ins) err</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>frame err.</td>
<td>hypothesis</td>
<td>(1.95/1.15) 8.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>symmetric</td>
<td>(2.22/0.99) 9.00</td>
</tr>
<tr>
<td></td>
<td>CN-dist.</td>
<td>n.a.</td>
<td>(1.65/1.33) 8.07</td>
</tr>
</tbody>
</table>

<sup>1</sup> tuning set, CER: [Chinese] Character Error Rate
### System Combination

#### Chinese 230h Test System

<table>
<thead>
<tr>
<th>System</th>
<th>Combination</th>
<th>CER [%]</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>dev07¹</td>
<td>eval07</td>
<td>dev08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1</td>
<td></td>
<td>14.54</td>
<td>15.08</td>
<td>13.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td></td>
<td>14.82</td>
<td>15.02</td>
<td>13.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td></td>
<td>15.07</td>
<td>15.60</td>
<td>13.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1+s2</td>
<td>union</td>
<td>13.54</td>
<td>13.96</td>
<td>12.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNC</td>
<td>13.55</td>
<td>13.95</td>
<td>12.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROVER</td>
<td>13.63</td>
<td>14.09</td>
<td>12.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1+s2+s3</td>
<td>union</td>
<td>13.15</td>
<td>13.65</td>
<td>12.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNC</td>
<td>13.16</td>
<td>13.74</td>
<td>12.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROVER</td>
<td>13.22</td>
<td>13.86</td>
<td>12.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>union CN</td>
<td>13.15</td>
<td>13.65</td>
<td>12.19</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹ tuning set, CER: [Chinese] Character Error Rate
## System Combination

### English TC-Star 2007 Evaluation System

<table>
<thead>
<tr>
<th>System</th>
<th>Combination</th>
<th>dev06</th>
<th>eval06(^1)</th>
<th>eval07</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td></td>
<td>11.09</td>
<td>8.43</td>
<td>9.81</td>
</tr>
<tr>
<td>s2</td>
<td></td>
<td>11.89</td>
<td>8.70</td>
<td>10.07</td>
</tr>
<tr>
<td>s3</td>
<td></td>
<td>12.43</td>
<td>8.98</td>
<td>10.76</td>
</tr>
<tr>
<td>s4</td>
<td></td>
<td>12.06</td>
<td>9.44</td>
<td>11.73</td>
</tr>
</tbody>
</table>

**s1+s2 union**
- CNC: 10.22, 7.82, 8.98
- ROVER: 10.54, 7.90, 9.11

**s1+s2+s3**
- CNC: 10.14, 7.70, 8.98
- ROVER: 10.42, 7.73, 9.17

**s1+s2+s3+s4**
- CNC: 10.33, 7.59, 8.94
- ROVER: 10.70, 7.67, 9.15

\(^1\) tuning set
## System Combination

### English TC-Star 2007 Cross-Site

<table>
<thead>
<tr>
<th>System</th>
<th>Combination</th>
<th>WER [%]</th>
<th>eval06 $^1$</th>
<th>eval07</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIMSI</td>
<td></td>
<td>8.16</td>
<td>9.13</td>
<td></td>
</tr>
<tr>
<td>RWTH</td>
<td></td>
<td>8.46</td>
<td>9.71</td>
<td></td>
</tr>
<tr>
<td>UKA</td>
<td></td>
<td>8.80</td>
<td>10.22</td>
<td></td>
</tr>
<tr>
<td>IRST</td>
<td></td>
<td>10.09</td>
<td>9.81</td>
<td></td>
</tr>
<tr>
<td>LIMSI+RWTH</td>
<td>union</td>
<td>6.46</td>
<td>7.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNC</td>
<td>6.38</td>
<td>7.51</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROVER</td>
<td>6.69</td>
<td>7.85</td>
<td></td>
</tr>
<tr>
<td>LIMSI+RWTH+UKA</td>
<td>union</td>
<td>6.38</td>
<td>7.63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNC</td>
<td>6.27</td>
<td>7.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROVER</td>
<td>6.32</td>
<td>7.77</td>
<td></td>
</tr>
<tr>
<td>LIMSI+RWTH+UKA+IRST</td>
<td>union</td>
<td>6.28</td>
<td>7.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNC</td>
<td>6.14</td>
<td>7.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROVER</td>
<td>6.21</td>
<td>7.26</td>
<td></td>
</tr>
</tbody>
</table>

$^1$ tuning set
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications
   4.1 Why Word Graphs?
   4.2 Word Graph Generation using Word Pair Approximation
   4.3 Word Graph Pruning
   4.4 Multi-Pass Search
   4.5 Language Model Rescoring
   4.6 Path probabilities over word graphs
   4.7 Confidence Measures
   4.8 Bayes Risk Decoding
   4.9 System Combination
   4.10 Experiments
   4.11 References

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training


References

Confidence Measures


References

Confidence Measures


References

Bayes Risk Decoding

References

Bayes Risk Decoding


► J. Xue and Y. Zhao. Improved confusion network algorithm and shortest path search from word lattice. ICASSP, 2005.


References

Model and System Combination

References
Model and System Combination


Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 M-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search: Idea

Idea

At every time $\tau$ a separate copy of the lexical prefix tree (time conditioned tree copy) with the hypothesis $Q_\tau(t, s)$ is started.

Possible advantage: using beam search only a few time conditioned tree copies are active (experiments will show 40-50).
Time Conditioned Search

Approach

- Time synchronous evaluation of the Bayes’ decision rule as before.
- Full sequence joint probability:

\[
Pr(w_1 \ldots w_n) \cdot Pr(x_1 \ldots x_T | w_1 \ldots w_N)
\]

- Definitions:
  - probability that word \( w \) generated the acoustic vectors \( x_{\tau+1}, \ldots, x_t \):

\[
h(w; \tau, t) := Pr(x_{\tau+1}^t | w)
\]

  - conditional probability for the generation of the acoustic vectors \( x_1, \ldots, x_t \) and a word sequence \( w_1^n \) that ends at time \( t \):

\[
G(w_1^n; t) := Pr(w_1^n) \cdot Pr(x_1^t | w_1^n)
\]
Time Conditioned Search

Approach

- Decomposition (recall word graph generation):

\[
\begin{align*}
\max_{\tau} \quad & G(w_1^{n-1};\tau) p(w_n|w_1^N) h(w_n;\tau, t) \\
= & G(w_1^n; t)
\end{align*}
\]

- Optimize over word boundary: \(\tau\)

\[
\begin{align*}
G(w_1^n; t) & = \max_{\tau} \{Pr(w_n|w_1^{n-1}) \cdot G(w_1^{n-1};\tau) \cdot h(w_n;\tau, t)\} \\
& = Pr(w_n|w_1^{n-1}) \cdot \max_{\tau} \{G(w_1^{n-1};\tau) \cdot h(w_n;\tau, t)\}
\end{align*}
\]

- So far: no optimization of the word boundaries: organization of word successor hypotheses as tree.
Time Conditioned Search

Approach

- So far: no optimization of the word boundaries: organization of word successor hypotheses as tree.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 M-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Consider the m-gram language model $p(u_m | u_{1}^{m-1})$.

- Distinguish word sequences by their last $(m - 1)$ words $u_{1}^{m-1}$
- Definition:
  - Conditional probability for an acoustic vector sequence $x_1 \ldots x_t$ and a word sequence that ends at time $t$ with a word subsequence $u_2^m$:
    \[
    H(u_2^m; t) = \max_{w_1^m} \left[ Pr(w_1^n) \cdot Pr(x_t^t | w_1^n) : w_{1-n+2}^n = u_2^m \right]
    \]
  - Optimize word ends (dynamic programming):
    \[
    H(u_2^m; t) = \max_{u_1} \left\{ p(u_m | u_{1}^{m-1}) \cdot \max_{\tau} \left[ H(u_{1}^{m-1}; \tau) \cdot h(u_m; \tau, t) \right] \right\}
    \]
  - Using bigram language model $p(w | v)$:
    \[
    H(w; t) = \max_{v} \left\{ p(w | v) \cdot \max_{\tau} \left[ H(v; \tau) \cdot h(w; \tau, t) \right] \right\}
    \]
Consider the hypothesis that word \( w \) ends at time \( t \).

Optimization over word boundary/word start time at word end!
0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 M-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search

Implementation

Principle:

- Time synchronous evaluation of the hypotheses (as before).
- Lexical search tree.
- Start a separate lexical tree at every time frame.
- Define: $Q_T(t, s) = \text{probability of the best partial path that generates the acoustic vectors } x_1 \ldots x_t \text{ up to a state } s \text{ at time } t \text{ of a lexical prefix tree with start time } \tau$. 
Time Conditioned Search

Implementation

Dynamic programming recursion for bigram language models:

- Initialization (best predecessor hypothesis):
  \[
  Q_{t-1}(t-1, s) = \begin{cases} 
  \max_u \{ H(u; t-1) \} & : s = 0 \\
  0.0 & : s > 0 
  \end{cases}
  \]

- Within words:
  \[
  Q_\tau(t, s) = \max_\sigma \{ p(x_t, s|\sigma) \cdot Q_\tau(t-1, \sigma) \}
  \]

- Word boundaries:
  \[
  H(w; t) = \max_v \{ p(w|v) \cdot \max_\tau \{ H(v; \tau) \cdot \frac{Q_\tau(t, S_w)}{\max_u H(u; \tau)} \} \}
  \]
  \[
  = \max_v \{ p(w|v) \cdot \max_\tau \{ H(v; \tau) \cdot h(w; \tau, t) \} \}
  \]
Properties of the time conditioned search:

▶ Time synchronous.
▶ Store the score $H(v; \tau)$ for corrections, modifications of word boundary optimization.
▶ Separate word boundary optimization over start time and predecessor word.
▶ No backpointers.
**Time Conditioned Search: Implementation**

**proceed over time** $t$ **from left to right**

<table>
<thead>
<tr>
<th>ACOUSTIC LEVEL: process hypotheses $Q_\tau(t, s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>- initialization:</td>
</tr>
</tbody>
</table>
| $Q_{t-1}(t-1, s) = \begin{cases} 
\max_u H(u; t-1) & \text{if } s = 0 \\
0.0 & \text{if } s > 0 
\end{cases}$ |
| - time alignment: $Q_\tau(t, s)$ using DP recursion |
| - prune unlikely hypotheses |
| - purge bookkeeping lists |

<table>
<thead>
<tr>
<th>WORD PAIR LEVEL: process hypotheses $Q_\tau(t, S_w)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>for each pair $(w; t)$ do</td>
</tr>
<tr>
<td>- word score: $h(w; \tau, t) = Q_\tau(t, S_w)/\max_u H(u; \tau)$</td>
</tr>
<tr>
<td>- best predecessor and best boundary:</td>
</tr>
<tr>
<td>$[v_0, \tau_0](w, t) := \arg \max_{v, \tau} { p(w</td>
</tr>
<tr>
<td>- store $[v_0, \tau_0](w, t)$</td>
</tr>
<tr>
<td>- tree start-up score:</td>
</tr>
<tr>
<td>$H(w; t) = p(w</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. **Time Conditioned Search**
   5.1 Idea
   5.2 $M$-Gram Language Models
   5.3 Implementation
   5.4 **Extension to Trigram Language Models**
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search

Extension to Trigram Language Models

- Trigram language model \( p(w|uv) \):

\[
H(v, w; t) = \max_u \left\{ p(w|uv) \cdot \max_{\tau} \left[ H(u, v; \tau) \cdot h(w; \tau, t) \right] \right\}
\]

- Dynamic programming recursion initialization (best word end hypothesis):

\[
Q_{t-1}(t-1, s) = \begin{cases} 
\max_{(u,v)} \{ H(u, v; t-1) \} & : s = 0 \\
0.0 & : s > 0
\end{cases}
\]

- Within words:

\[
Q_\tau(t, s) = \max_{\sigma} \{ p(x_t, s|\sigma) \cdot Q_\tau(t-1, \sigma) \}
\]

- Word boundaries:

\[
H(v, w; t) = \max_u \left\{ p(w|uv) \cdot \max_{\tau} \left[ H(u, v; t) \cdot \frac{Q_\tau(t, S_w)}{\max_{(u',v')} H(u', v'; \tau)} \right] \right\}
\]
Time Conditioned Search: Experimental Results

Test: NAB’94 – H1 Development

- vocabulary: 20000 words
- 10 male and 10 female speakers
- total of 310 sentences = 7387 spoken words
- unknown words: 199 (out of vocabulary rate: 2.7 %)
- test set perplexities:
  - bigram: $PP_{bi} = 198.1$
  - trigram: $PP_{tri} = 130.2$

Acoustic Modelling:

- trained on: WSJ0 and WSJ1 (284 speakers; 80 h of speech)
- phoneme models: 4688 (43+1 monophones, 557 diphones, 4087 triphones)
- tied states: 4623 (6-state HMM)
- densities: 290 000 per gender
## Time Conditioned Search

**Experimental Results: Word Conditioned Tree Search vs. Time Conditioned Tree Search**

Bigram language model, PP=198:

<table>
<thead>
<tr>
<th>Search</th>
<th>States</th>
<th>Arcs</th>
<th>Trees</th>
<th>Words</th>
<th>LM</th>
<th>Rec</th>
<th>del - ins</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>word conditioned</strong></td>
<td>5022</td>
<td>1416</td>
<td>20</td>
<td>86</td>
<td>86</td>
<td></td>
<td>2.5 - 2.7</td>
<td>17.1</td>
</tr>
<tr>
<td></td>
<td>9335</td>
<td>2593</td>
<td>28</td>
<td>156</td>
<td>156</td>
<td></td>
<td>2.5 - 2.7</td>
<td>16.8</td>
</tr>
<tr>
<td></td>
<td>26493</td>
<td>7222</td>
<td>45</td>
<td>540</td>
<td>540</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td>39146</td>
<td>10528</td>
<td>53</td>
<td>610</td>
<td>610</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>52786</td>
<td>13968</td>
<td>60</td>
<td>560</td>
<td>560</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.4</td>
</tr>
<tr>
<td><strong>time conditioned</strong></td>
<td>6175</td>
<td>1781</td>
<td>27</td>
<td>117</td>
<td>1539</td>
<td></td>
<td>2.4 - 2.9</td>
<td>17.3</td>
</tr>
<tr>
<td></td>
<td>11280</td>
<td>3101</td>
<td>31</td>
<td>197</td>
<td>2976</td>
<td></td>
<td>2.4 - 2.7</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>30019</td>
<td>7998</td>
<td>37</td>
<td>520</td>
<td>8761</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td>44289</td>
<td>11679</td>
<td>41</td>
<td>777</td>
<td>13162</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td>60692</td>
<td>15877</td>
<td>43</td>
<td>1075</td>
<td>18081</td>
<td></td>
<td>2.4 - 2.6</td>
<td>16.4</td>
</tr>
</tbody>
</table>
### Time Conditioned Search

#### Experimental Results: Word Conditioned Tree Search vs. Time Conditioned Tree Search

Trigram language model, PP=122:

<table>
<thead>
<tr>
<th>LM</th>
<th>States</th>
<th>Arcs</th>
<th>Trees</th>
<th>Words</th>
<th>LM</th>
<th>Rec</th>
<th>del - ins</th>
<th>WER[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>word conditioned</td>
<td>4714</td>
<td>1392</td>
<td>29</td>
<td>80</td>
<td>80</td>
<td></td>
<td>1.7 - 2.9</td>
<td>14.8</td>
</tr>
<tr>
<td></td>
<td>18734</td>
<td>5430</td>
<td>70</td>
<td>294</td>
<td>294</td>
<td></td>
<td>1.6 - 2.8</td>
<td>14.2</td>
</tr>
<tr>
<td></td>
<td>48940</td>
<td>13877</td>
<td>112</td>
<td>702</td>
<td>702</td>
<td></td>
<td>1.6 - 2.8</td>
<td>14.1</td>
</tr>
<tr>
<td></td>
<td>67541</td>
<td>18764</td>
<td>130</td>
<td>953</td>
<td>953</td>
<td></td>
<td>1.6 - 2.7</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>86772</td>
<td>23688</td>
<td>145</td>
<td>1223</td>
<td>1223</td>
<td></td>
<td>1.6 - 2.7</td>
<td>13.9</td>
</tr>
<tr>
<td>time conditioned</td>
<td>8573</td>
<td>2459</td>
<td>28</td>
<td>136</td>
<td>1477</td>
<td></td>
<td>1.6 - 2.9</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>15103</td>
<td>4194</td>
<td>31</td>
<td>234</td>
<td>2806</td>
<td></td>
<td>1.6 - 2.8</td>
<td>14.1</td>
</tr>
<tr>
<td></td>
<td>24914</td>
<td>6780</td>
<td>34</td>
<td>388</td>
<td>4858</td>
<td></td>
<td>1.6 - 2.7</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>38323</td>
<td>10299</td>
<td>38</td>
<td>606</td>
<td>7663</td>
<td></td>
<td>1.6 - 2.7</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>71757</td>
<td>18953</td>
<td>42</td>
<td>1202</td>
<td>14293</td>
<td></td>
<td>1.6 - 2.7</td>
<td>14.0</td>
</tr>
</tbody>
</table>

Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 M-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search

Equivalence of Word Conditioned and Time Conditioned Search Organization

For time conditioned search define:

\[ Q_{\tau}(t, s) := \max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^t} \left\{ Pr(x_1^t, s_1^t | w_1^{n-1}, \ldots) \right\} : s_{\tau} = S_{w_{n-1}}; s_t = s \right\} \]

Review:

Word conditioned tree search for m-gram language models \( p(v_m | v_1^{m-1}) \):

\[ Q_{v_1^{m-1}}(t, s) := \max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^t} \left\{ Pr(x_1^t, s_1^t | w_1^{n-1}) \right\} : w_{n-m+1}^{n-1} = v_1^{m-1}; s_t = s \right\} \]
Time Conditioned Search

Equivalence of Word Conditioned and Time Conditioned Search Organization

time conditioned search

\[ \tau \rightarrow t \]

word conditioned search

\[ v_1^{m-1} \rightarrow t \]
The path with the history $v_1^{m-1}$ that ends at time $t$ in state $s$, must have reached state $S_{v_{m-1}}$ once at time $\tau$: $s_\tau := S_{v_{m-1}}$.

This path is split into two partial paths at time $\tau$, were time $\tau$ has to be optimized:

$$Q_{v_1^{m-1}}(t, s)$$

$$= \max_{\tau} \left[ \max_{w_1^{n-1}} \left\{ Pr(w_1^{n-1}) \cdot \max_{s_1^{\tau}} \left\{ Pr(x_1^{\tau}, s_1^{\tau}|w_1^{n-1}) : w_{n-m+1}^{n-1} = v_1^{m-1}; s_\tau = S_{v_{m-1}} \right\} \right\} \right]$$

$$\cdot \max_{s_{\tau+1}^{t}} \left\{ Pr(x_{\tau+1}^{t}, s_{\tau+1}^{t}|--) : s_t = s \right\}$$

$$= \max_{\tau} \left[ H(v_1^{m-1}; \tau) \cdot \max_{s_{\tau+1}^{t}} \left\{ Pr(x_{\tau+1}^{t}, s_{\tau+1}^{t}|--) : s_t = s \right\} \right]$$

$$= \max_{\tau} \left[ H(v_1^{m-1}; \tau) \cdot \frac{Q_\tau(t, s)}{\max_{u_1^{m-1}} H(u_1^{m-1}, \tau)} \right]$$
Time Conditioned Search

Equivalence of Word Conditioned and Time Conditioned Search Organization

\[
Q_{v_1^{m-1}}(t, s) = \max_{\tau} \left[ \frac{H(v_1^{m-1}; \tau)}{\max_{u_1^{m-1}} H(u_1^{m-1}, \tau)} \cdot Q_\tau(t, s) \right] \quad \forall s \text{ with } s \neq 0.
\]

Especially for a word boundary \( s = S_{v_m} \) we get:

\[
Q_\tau(t, S_{v_m}) = \max_{u_1^{m-1}} H(u_1^{m-1}, \tau) \cdot h(v_m; \tau, t)
\]

Substitute:

\[
Q_{v_1^{m-1}}(t, S_{v_m}) = \max_{\tau} \left[ H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \right]
\]
Time Conditioned Search

Equivalence of Word Conditioned and Time Conditioned Search Organization

Word conditioned search with recombination for $s = 0$:

\[ Q_{v_2^m}(t, s = 0) = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m}) \right\} \]

\[ = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot \max_{\tau} \{ H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \} \right\} \]

\[ = H(v_2^m; t). \]

- It is possible to convert scores of the time conditioned search into scores of the word conditioned search.
- The opposite conversion is usually impossible because of the optimization over $\tau$ which is done implicitly in word conditioned search.
- time conditioned search is “less compact” without further optimization/pruning.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 M-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search

Across-Word Contexts

- Across-word contexts lead to multiple tree roots, depending on left phoneme context of words to be started.
- **Problem**: time conditioned search assumes unique tree root, relative to which word probabilities can be computed.
- **Observation**: coarticulated root states are recombined after first generation.
- **Idea**: partition tree into word-conditioned root network, followed by time-conditioned subtrees.
- Subtrees are compatible with time conditioned search structure, i.e. have unique tree roots.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search
   5.1 Idea
   5.2 $M$-Gram Language Models
   5.3 Implementation
   5.4 Extension to Trigram Language Models
   5.5 Equivalence of Word Conditioned and Time Conditioned Search Organization
   5.6 Across-Word Contexts
   5.7 Pruning

6. Normalization and Adaptation

7. Discriminative Training
Time Conditioned Search

Pruning

Standard pruning:

- As for word conditioned search, acoustic and language model pruning are applied.
- Acoustic pruning:
  - Acoustic pruning threshold:
    \[ Q_{AC}(t) := \max_{\tau, s} Q_{\tau}(t, s). \]
  - Discard all hypotheses \((\tau, s)\) at time \(t\) for which
    \[ Q_{\tau}(t, s) < f_{AC} \cdot Q_{AC}(t). \]
  - Note: Implicitly, pruning affects all word end hypotheses attached to a search tree.
Time Conditioned Search

Pruning

- Language model pruning:
  - Language model recombination/word boundary optimization:

\[
H(v^m_2; t) = \max_{v_1} \left\{ p(v_m|v^{m-1}_1) \cdot \max_{\tau} \frac{H(v^{m-1}_1; \tau) \cdot Q_\tau(t, S_{v_m})}{\max_{u^{m-1}_1} H(u^{m-1}_1; \tau)} \right\}
\]

\[
Q_t(t, 0) = \max_{v^m_2} H(v^m_2; t).
\]

- Pruning threshold:

\[
Q_{LM}(t) := \max_{u^M_2} H(u^M_2; t).
\]

- Define set \(E(t)\) of all word end hypotheses surviving language model pruning at a given time frame:

\[
E(t) := \{(v^{m-1}_1, q) | q = H(v^{m-1}_1; t), q > f_{LM} \cdot Q_{LM}(t)\}
\]

- Attach \(E(t)\) to the corresponding search tree started at time \(t\).
Time Conditioned Search

Pruning

- **LM look-ahead:**
  - only unigram LM look-ahead is efficient.
  - LM look-ahead with context dependency (bigram or higher) requires maximization over all contexts, information gain does not pay off w.r.t. efficiency.

- **Efficiency problems:**
  - Only single best ending word hypotheses is used to start tree at a given time frame.
  - Word end hypotheses with different probabilities attached to search tree.
    \[\Rightarrow\] Some of the word end hypotheses might already be below the pruning thresholds.
  - LM recombination builds on all word end hypotheses with varying start times.
  - LM pruning is applied afterwards.
    \[\Rightarrow\] Word boundary optimization delayed.
Time Conditioned Search

Pruning

Early word end pruning:
▶ To prevent too much language model recombinations, apply acoustic pruning to expanded word end hypotheses before computing the LM probability (cf. word conditioned search).
▶ Word boundary optimization:

\[
Q_{v_1^{m-1}}(t) := \max_{\tau} \frac{H(v_1^{m-1}; \tau) \cdot Q_{\tau}(t, S_{v_m})}{\max_{u_1^{m-1}} H(u_1^{m-1}; \tau)}
\]

▶ Acoustic pruning on expanded word end hypotheses:
▶ Discard all word end hypotheses \((v_1^{m-1}; t)\) with

\[
Q_{v_1^{m-1}}(t) < f_{AC} \cdot Q_{AC}(t)
\]

▶ LM recombination on remaining hypotheses:

\[
H(v_2^m; t) = \max_{v_1} \{ p(v_m|v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t) \}
\]

\[
Q_t(t, 0) = \max_{v_2^m} H(v_2^m; t).
\]

▶ Anticipated LM pruning: use/update current best hypothesis.
Time Conditioned Search

Pruning

Context Recombination

- In time conditioned search word boundaries are recombined only at the word ends, whereas in word conditioned search, word boundaries are recombined implicitly early on.
- Search trees for adjacent start times can be expected to expand to the same state hypotheses (and therefore also the same word end hypotheses).
- Idea: discard word end hypothesis $v^m_{1-1}$ from specific time frame, if all expanded state hypotheses with respect to this word end hypothesis within one tree are lower than the equivalent state hypotheses within another tree.

A hypothesis $(v^m_{1-1}, q) \in E(\tau)$ ending in the word sequence $v^m_{1-1}$ with probability $q$ can be removed from $E(\tau)$ if:

$$q \cdot \frac{Q_\tau(t, s)}{Q_\tau(\tau, 0)} < \max_{\tau', q' : (h, q') \in E(\tau')} q' \cdot \frac{Q_{\tau'}(t, s)}{Q_{\tau'}(\tau', 0)}$$
Time Conditioned Search

Pruning

- Complexity of context recombination is problematic: all active contexts $v_1^{m-1}$, and states $s$ have to be visited.
- Approximation: perform context recombination not in every time frame, but at certain intervals.

Further speed-up of time conditioned search:

- Using an HMM with skip transition, a word end hypothesis still only is generated from the last state of a word, even though a skip from the second to last state also would leave the words HMM.
- Nevertheless, this approximation does not have a significant effect on the search result.
- In the same way it can be expected that the effect would be small, if word starts also would not be allowed at every time frame.
- Approach: only allow word starts (i.e. tree starts) at certain intervals.
Time Conditioned Search

Pruning: Results

Search space comparison:

- Average number of different active word end state hypotheses per time frame, independent of context and start time.
- Average number of word end hypotheses per time frame.

Results on a small subset of the NAB’94 H1 dev corpus with 20k word vocabulary, with relaxed pruning constraints:

<table>
<thead>
<tr>
<th>search trees conditioned on:</th>
<th>word context [k]</th>
<th>start time [k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word end states</td>
<td>5</td>
<td>5.5</td>
</tr>
<tr>
<td>word ends:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o early pruning</td>
<td>5</td>
<td>1150</td>
</tr>
<tr>
<td>with early pruning</td>
<td></td>
<td>11.3</td>
</tr>
<tr>
<td>after pruning</td>
<td>1.6</td>
<td>8.3</td>
</tr>
<tr>
<td>after recombination</td>
<td>0.77</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Experiments on European Parliament Plenary Sessions (EPPS) English:

- Vocabulary: 60k words.
- Across-word models.
- Word conditioned search (WCS):
  - optional bigram language model look-ahead (Bigram LAH)
- Time Conditioned Search (TCS):
  - Context recombination every 5th time frame (CR5).
  - Tree start every 2nd time frame (Interval 2).
- Measure word error rate (WER) vs. real time factor (RTF).
- Comparison of time and word conditioned search.
Time Conditioned Search

Pruning: Results

EPPS Eval06, 60k words

WER [%]

RTF

TCS
TCS, CR5
TCS Interval 2
TCS Interval 2, CR5
Time Conditioned Search
Pruning: Results

![Graph showing WER (%) vs. RTF for different models: WCS, WCS, Bigram LAH, TCS, TCS Interval 2, CR5.
EPPS Eval06, 60k words.

**WER [%]**
- WCS
- WCS, Bigram LAH
- TCS
- TCS Interval 2, CR5

**RTF**
Time Conditioned Search

Conclusions

Pruning experiments show:

▶ Context recombination leads to moderate improvements.
▶ A tree start interval of 2 frames gives significant speed-up.
▶ Context recombination gives negligible effect on top of tree start interval.
▶ Tree start interval larger than 2 gives degradations.

Overall:

▶ With optimized pruning, time conditioned search outperforms word conditioned search.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Normalization and Adaptation

Motivation

Objective: try to fit acoustic model to speaker/condition.
- Need new acoustic adaptation data from the current speaker or condition.

Distinctions:
- Model based – Feature based
  - Modify model to better fit the features ⇒ adaptation.
  - Transform features to better fit model ⇒ normalization.
- Supervised – Unsupervised
  - Transcriptions available for adaptation data ⇒ supervised.
  - No transcriptions available ⇒ unsupervised.
- Recognition side – Also in training
  - Normalization also in training is straightforward.
  - Possible also for adaptation ⇒ speaker adaptive training (SAT).
Normalization and Adaptation

Supervised Adaptation

Supervised adaptation = constrained acoustic model reestimation.

- Need manual transcriptions of adaptation data.
- Theoretically similar to acoustic model training – maximum likelihood.

Model based: $M' = f(M, \phi)$

$$\hat{\phi} = \arg \max_{\phi} P(x_1^T | w_1^T, M').$$

Feature based: $x'_1 = f(x_1^T, \theta)$ (note Jacobi determinant)

$$\hat{\theta} = \arg \max_{\theta} \left| \frac{df(x_1^T, \theta)}{dx_1^T} \right| P(x'_1 | w_1^T, M).$$

Like in AM training: discriminative training possible and useful.
Simultaneous generation of adaptation word sequence and estimation of adaptation parameters.

- Model based: $M' = f(M, \phi)$

$$\{\hat{w}_1^T, \hat{\phi}\} = \arg \max_{\{w_1^T, \phi\}} P(w_1^T)P(x_1^T | w_1^T, M').$$

- Feature based method can be defined analogously.

- Above method infeasible in practice.

- Approximations always used.
Normalization and Adaptation

Unsupervised Adaptation - Approximations

**Word graph** based approximation

- Perform **first pass** recognition generating word graph.
- Posterior confidence measures from word graph.
- Weighted accumulation.

**First-best** approximation

- Only take first best output instead of word graph.
- Estimation is performed exactly as in the supervised case, but use first pass output as transcription.
- This method is used most often in practice.
Normalization and Adaptation

Estimation Criteria

Different estimation criteria are possible

- **Supervised adaptation**
  - Discriminative criteria can be applied (see Chapter 7).
  - No principal difference to acoustic model training.

- **Unsupervised adaptation**
  - Currently no significant improvement using discriminative criteria compared to maximum likelihood estimation.
  - Appropriate criteria are still an open research issue.
Normalization and Adaptation

Use Cases: Transcription System

- **Objective**: generate transcription from untranscribed audio data.

- Offline system – multiple passes over data allowed.

- Several different speakers and conditions, this requires:
  - Segmentation, to partition audio in homogeneous portions: single speaker and single condition per segment.
  - Clustering of segments into same (similar) speakers and/or conditions.

- Speakers not previously known/no transcribed adaptation data available: unsupervised adaptation.
Normalization and Adaptation

Use Cases: Dictation System

- **Objective**: Transcribe speech from known speaker in real time.
- Typically only one speaker uses each system (at a time).
- User corrects errors in the ASR output ⇒
  - Corrected transcripts usable for supervised adaptation.
- Amount of adaptation data continually increases ⇒
  - Complexity of adaptation method should also increase.
  - Finally complete retraining of acoustic model might be used.
Normalization and Adaptation

Use Cases: On-line Interactive System

- **Objective:** Transcribe speech from different *unknown* speakers in real time.
- Used for dialog as well as command and control systems.
- Unsupervised adaptation.
- Requires **fast** adaptation, i.e. adaptation using very little data (some seconds).
- Can benefit from **incremental** adaptation, i.e. continually updating adaptation for new data.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Audio Segmentation

Objective: split audio stream in homogeneous regions according to:

- speaker identity,
- recording condition (e.g. background noise, telephone channel),
- signal type (e.g. speech, music, noise, silence), and
- spoken word sequence.

Segmentation affects speech recognition performance:

- speaker adaptation and speaker clustering: assume one speaker per segment,
- language model assumes sentence boundaries at segment end,
- non-speech regions cause insertion errors,
- overlapping speech is not recognized correctly, causes errors at surrounding regions,
- words must not be split.
Audio Segmentation

Methods

- **Metric based**
  - Compute distance between adjacent regions.
  - Segment at maxima of the distances.
  - Distances: KL distance, Bayesian information criterion.

- **Model based**
  - Classify regions using precomputed models for music, speech, etc.
  - Segment at changes in acoustic class.

- **Decoder guided**
  - Apply speech recognition to input audio stream.
  - Segment at silence regions.
  - Other information produced by the decoder can be used.
Bayesian Information Criterion (BIC):

- Likelihood criterion for a model $\theta$, given data set $x_1^N$:

$$BIC(\theta, x_1^N) = \log p(x_1^N|\theta) - \frac{\lambda}{2} \cdot d(\theta) \cdot \log(N),$$

$d(\theta)$: number of parameters in $\theta$, 
$\lambda$: penalty weight for model complexity.

- used for model selection: choose model maximizing the BIC.
Audio Segmentation

Metric-based using BIC (ctd.)

Change point detection using BIC:

- Input stream is modeled as Gaussian process in the cepstral space.
- Feature vectors of one segment: drawn from multivariate Gaussian:
  \[ x_i \ldots x_j \sim \mathcal{N}(\mu, \Sigma). \]
- For hypothesized segment boundary \( t \) in \( x^T \) decide between
  \[ x_1 \ldots x_T \sim \mathcal{N}(\mu, \Sigma), \text{ and} \]
  \[ x_1 \ldots x_t \sim \mathcal{N}(\mu_1, \Sigma_1) \quad x_{t+1} \ldots x_T \sim \mathcal{N}(\mu_2, \Sigma_2). \]
- Use difference of BIC values:
  \[
  \Delta \text{BIC}(t) = T \log |\Sigma| - t |\Sigma_1| - (T - t) |\Sigma_2| - \lambda P
  \]
  \[
  P = \frac{1}{2} (D + \frac{1}{2} D(D + 1)) \log T
  \]

  \( D \): dimensionality of feature vectors.
- Change point if \( \Delta \text{BIC}(t) > 0 \).
Audio Segmentation
Metric-based using BIC (ctd.)

- Detecting single change point in $x_1^T$:

$$\hat{t} = \arg\max_t \{ \Delta \text{BIC}(t) \}$$

- Detecting multiple change points:
  - Apply change point detection to a sliding, size-growing window.
  - Start with a window of initial size.
  - Enlarge window until change point detected, or maximum window size reached.
  - Restart with initial window size behind previous change point.
Literature


Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Speaker Clustering

Introduction

- **Objective:** Group speech segments into clusters for adaptation.
- **Segments from same or similar speakers should be grouped.**
- **Requirement:** Clustering should give lowest possible adaptation WER.

**BIC Method**

- Uses **acoustic** features only.
- Greedy, bottom up clustering.
- BIC used to control number of clusters.
Speaker Clustering

BIC Clustering

- Greedy, bottom up, BIC clustering method [Chen et al. @IBM 1998].
- Each cluster is modeled using single Gaussian, full covariance.
- BIC criterion, $C_k$ is clustering with $K$ clusters, $x_1^T$ input data points, and feature dimension $D$:

$$BIC(C_k, x_1^T) = \sum_{k=1}^{K} \left( -\frac{1}{2} n_k \log |\Sigma_k| \right) - \frac{1}{2} K(D + \frac{1}{2} D(D + 1)) \log T.$$ 

- Algorithm:
  0. Start with one cluster for each segment.
  1. Try all possible pairwise cluster merges.
  2. Merge the pair that gives the largest increase in $BIC(C_k, x_1^T)$.
  3. Iterate from 1, until $BIC(C_k, x_1^T)$ starts to decrease.
In practice modifications to the BIC criterion can be used [Zhou+ @ Uni. Colorado 2000].

Criterion takes form: likelihood term + complexity term.

As in segmentation section, multiply complexity term with a penalty weight $\lambda$.

Tune $\lambda$ by optimizing WER on a development set.

In a practical implementation, a $K \times K$ matrix is used to cache change in BIC for each merge pair, between iterations.

Also possible to use BIC only as stop criterion, with distance measure to choose pair-wise merging of clusters.
Speaker Clustering

References

▶ S. S. Chen and P. S. Gopalakrishnan, Clustering Via the Bayesian Information Criterion With Applications in Speech Recognition, ICASSP 1998.

Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Introduction

What is the vocal tract?

- Dimensions of vocal tract vary between speakers.
- Different vocal tract lengths give different formant positions
  ⇒ systematic differences in features between speakers.
- **Goal**: warp frequency spectrum to match spectra from different speakers for the same phoneme.
Vocal Tract Length Normalization (VTLN)

Introduction

![Formants Graph](image-url)
Vocal Tract Length Normalization (VTLN)

Parameterization

- VTLN warping performed as part of acoustic feature extraction.
- Warping is performed on spectrum, before Mel-warping.
- Typically: warp filter bank centers/widths (as Mel-warping).

Linear warping

- Single (varying) parameter $\alpha$, termed warping factor.
- To get bijective mapping, knee parameter $\omega_0$ often used $\Rightarrow$ piecewise linear warping.
Vocal Tract Length Normalization (VTLN)

Estimation

Estimation from formant position

- Compute mean position of (some) formant for speaker.
- Derive warping factor from difference with global formant position.
- In [Eide et al. 96], third formant used, c.f. the paper for details.

Maximum likelihood (ML) estimation [Lee et al. AT&T 1996]

\[
\hat{\alpha} = \arg \max_{\alpha} P(x^{(\alpha)}_1 | \alpha, w_N^1)
\]

- Grid search over warping factors, i.e. [0.8:0.1:1.2], maximization problem.

- In practice: Use alignment (or other state distribution) obtained by existing acoustic model.

- Stable due to single parameter, can be used speaker-wise or segment-wise.
Vocal Tract Length Normalization (VTLN)

Variants

**Fast VTLN** [Welling@RWTH 1999]
- ML estimation requires transcription.
- For use in recognition ⇒ Two pass unsupervised.
- **Goal:** eliminate first recognition pass:
  - Train (GMM) warping factor classifier on acoustic training set.
  - Classifier uses only acoustic features (similar to clustering).
  - In recognition, estimate warping factor using classifier.

**Effect of Jacobi Determinant**
- Varying transformation (=warping) in optimization.
- Thus, should include Jacobi determinant.
- **Solution:** VTLN as linear transform, see [Pitz@RWTH 200] and [Umesh@Indian Inst. Tech. 2005].
Vocal Tract Length Normalization (VTLN)

References

▶ L. Lee, R. C. Rose, Speaker normalization using efficient frequency warping procedures, ICASSP 1996.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 AudioSegmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Maximum Likelihood Linear Regression (MLLR)

Introduction

- **Goal**: Approximate speaker specific acoustic model using affine transforms of speaker independent model Gaussian parameters [Leggetter+ @Cambridge 1995].

- **Parameterizations** ($r$ is speaker/condition)
  - Mean adaptation: $\mu'_{rs} = A_{rs}\mu_s + b_{rs}$.
  - Covariance adaptation: $\Sigma'_{rs} = H_{rs}\Sigma_s H_{rs}^T$.
  - Often only mean adaptation used – $\Sigma$-adaptation not discussed here.

- Typically estimated using maximum likelihood criterion.

- State tying: $A_{rs}$, $b_{rs}$, and $H_{rs}$ are either global or depend on sets of phonetically related states.
Maximum Likelihood Linear Regression (MLLR) Estimation

- Rewrite MLLR formulation:
  \[ \xi_s = [1 \mu_s^T]^T, \quad W_{rs} = [b_{rs} A_{rs}], \mu'_rs = W_{rs} \xi_s. \]

- Assume adaptation data \((x_1^T, w_1^N)\) from single speaker, drop index \(r\).

- Depending on the amount of adaptation data, **tying** is applied.

- For the derivation, we assume a single pooled adaptation matrix \(W\), i.e. we assume complete tying.

- All other cases of tying can be derived from the pooled case by dividing the adaptation data according to the Viterbi alignment.

- Parameter estimation using maximum likelihood criterion:
  \[ \hat{W} = \arg \max_W P(x_1^T | w_1^N, \theta, W). \]

- Optimization using expectation maximization (EM) gives:
  \[ W_{ML} = \arg \min_W \left\{ \sum_s \sum_{t=1}^T \gamma_s(t)(x_t - W \xi_s)^T \Sigma_s^{-1}(x_t - W \xi_s) \right\}. \]
Maximum Likelihood Linear Regression (MLLR)

Estimation: General Case - Full Covariance

Take derivative w.r.t. adaptation matrix component $W$ and set to zero:

$$\frac{\partial}{\partial W} \left\{ \sum_s \sum_{t=1}^T \gamma_s(t)(x_t - W \xi_s)^T \Sigma_s^{-1}(x_t - W \xi_s) \right\}$$

$$\propto \sum_s \sum_{t=1}^T \gamma_s(t) \Sigma_s^{-1} \left[ x_t \xi_s^T - W \xi_s \xi_s^T \right] \overset{!}{=} 0$$

$$\Leftrightarrow \sum_s \sum_{t=1}^T \gamma_s(t) \Sigma_s^{-1} W \xi_s \xi_s^T = \sum_s \sum_{t=1}^T \gamma_s(t) \Sigma_s^{-1} x_t \xi_s^T$$
Maximum Likelihood Linear Regression (MLLR)

Estimation: Case with Diagonal Covariances

Now assume diagonal covariances $\Sigma_{s,ij} = \sigma^2_{s,i} \delta_{ij}$.

Component-wise representation and rearrangement:

\[
\sum_{s} \sum_{t=1}^{T} \gamma_s(t) \Sigma^{-1}_s \xi_s \xi^T_s = \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \Sigma^{-1}_s x_t \xi^T_s
\]

\[\Leftrightarrow \sum_{n} W_{in} \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \frac{1}{\sigma^2_{s,i}} \xi_{s,n} \xi_{s,j} \quad = \quad \sum_{s} \frac{1}{\sigma^2_{s,i}} \sum_{t=1}^{T} \gamma_s(t) x_{t,i} \xi_{s,j} \quad \forall \ i, j\]

\[\Leftrightarrow \left[ W \cdot G^{(i)} \right] = Z_{ij} \quad \forall \ i, j \quad \mid \cdot (G^{(i)-1})_{jk} \quad \forall \ k, \quad \sum_{j}\]

\[\Leftrightarrow W_{ik} = \left( Z \cdot G^{(i)-1} \right)_{ik} \quad \forall \ i, k\]
Maximum Likelihood Linear Regression (MLLR)

Estimation

Therefore, assuming diagonal covariances, the adaptation matrix can be obtained row-wise, i.e. for each row $i$ separately:

$$W_{ij} = (Z_s G_s^{(i) -1})_{ij} \quad \forall \ i, j$$

with accumulators

$$G^{(i)} = \sum_s \frac{1}{\sigma_{s,i}^2} \xi_s \xi_s^T \sum_{t=1}^T \gamma_s(t),$$

$$Z = \sum_s \sum_{t=1}^T \gamma_s(t) \Sigma_s^{-1} x_t \xi_s^T \quad \text{with} \ \Sigma_s = \text{diag}(\sigma_{s,1}^2, \ldots, \sigma_{s,D}^2).$$

- Closed form solution for adaptation matrix.
- Accumulator storage requirements: $\mathcal{O}(D^3)$ (for $G^{(i)}$).
Maximum Likelihood Linear Regression (MLLR)

Tree Based State Tying

- Described in [Leggetter^{+}@Cambridge 1995].
- Basis: Phonetic classification tree, i.e. from AM state tying.
- Each node represents a regression class, a set of states.
- Typically silence has a separate state.
- Dynamically adjusted regression classes – Algorithm:
  - Accumulation of statistics in each leaf node.
  - Additionally accumulate observation count accumulator.
  - Combine (=add) accumulators from all child nodes – store in parent node.
  - Continue merging until root node is reached.
  - For each leaf node:
    1. If observation count larger than threshold: use accumulator.
    2. Else go to parent node and repeat from 1.

- Algorithm allows regression classes with too few observations to still be adapted, by combining data from other classes.
Maximum Likelihood Linear Regression (MLLR) 

Variants

- Standard MLLR uses same number of matrices \( A \) and offsets \( b \).
- But also possible to have many more offsets than matrices [Digitakis@John Hopkins Workshop 1999].
- Allows finer granularity in adjusting number of parameters.
- When no transformation matrices, only offsets – sometimes known as shift MLLR or bias MLLR.
- Estimation is very similar to standard MLLR, but \( A \) and \( b \) computed separately.
- Method for \( A \) is formally the same as for matrix \( W \) in MLLR.
- Offset \( b_c \) given by the following formula - depends on corresponding matrix \( A_{c^{'}}(c) \).

\[
b_c = \frac{1}{T} \sum_{t=1}^{T} \sum_{s_t \in c} \left( x_t - A_{c^{'}}(c) \mu_{s_t} \right).
\]
Maximum Likelihood Linear Regression (MLLR)

References

- C. J. Leggetter and P. C. Woodland, Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models, CSL April 1995
- C. J. Leggetter and P. C. Woodland, Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression, ARPA SLT Workshop 1995
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Feature Space MLLR (fMLLR)

Introduction

- **Goal:** Normalize features to better fit speaker [Digalakis+@SRI 1995].
- **Affine transform of features:** \( x'_t = A_{rs}x_t + b_{rs} \).
- **Note:** transformation of probability density, Jacobian determinant needed:

\[
P(x'_t) = \mathcal{N}(x'_t|\mu_s, \Sigma_s) \Leftrightarrow P(x_t) = |A_{rs}| \mathcal{N}(A_{rs}x_t + b_{rs}|\mu_s, \Sigma_s).
\]

- **Advantages:**
  - Pure feature transform \( \Rightarrow \) no changes to core speech recognition decoder needed.
  - Speaker adaptive training easy to implement.
- **State tying:** single global matrix per speaker.
- **But:** Can also be seen as model transform \( \Rightarrow \) regression classes possible with changes to decoder
MLLR with same matrix for mean and covariance transform:

\[
P(x_t \mid A_{rs}, b_{rs}, \mu_s, \Sigma_s) = \frac{1}{\sqrt{\det(2\pi A_{rs} \Sigma_s A_{rs}^T)}} e^{-\frac{1}{2} (x_t - A_{rs} \mu_s - b_{rs})^T (A_{rs} \Sigma_s A_{rs}^T)^{-1} (x_t - A_{rs} \mu_s - b_{rs})} \\
= \frac{1}{\det(A_{rs}) \sqrt{\det(2\pi \Sigma)}} e^{-\frac{1}{2} (A'_{rs} x + b'_{rs} - \mu)^T \Sigma^{-1} (A'_{rs} x + b'_{rs} - \mu)} \\
= \det(A'_{rs}) \mathcal{N}(A'_{rs} x \mid \mu, \Sigma).
\]

where \( A'_{rs} = A_{rs}^{-1} \) and \( b'_{rs} = -A_{rs}^{-1} b_{rs} \).

- This is exactly the formula for fMLLR:
  - Gaussian distributed random variable, affine transform
    \( \Rightarrow \) Gaussian also in new coordinate system.
  - Well known fact from mathematical statistics.
Feature Space MLLR (fMLLR)

Maximum Likelihood Estimation

As for MLLR, rewrite in linear form, drop speaker index \( r \), and assume complete tying:

\[
\chi_t' = W \xi_t \quad \text{with} \quad \xi_t = [1 \ x_t^T]^T \quad \text{and} \quad W = [b \ A].
\]

Maximum likelihood EM equation:

\[
W_{\text{ML}} = \arg \max_W \left\{ T \log \det A - \frac{1}{2} \sum_s \sum_{t=1}^T \gamma_s(t) (W \xi_t - \mu_s)^T \Sigma_s^{-1} (W \xi_t - \mu_s) \right\}
\]

\[
= \arg \max_W \left\{ T \log \det A + \sum_s \sum_{t=1}^T \gamma_s(t) \text{Trace} \left( W \xi_t \mu_s^T \Sigma_s^{-1} \right) - \frac{1}{2} \sum_s \sum_{t=1}^T \gamma_s(t) \text{Trace} \left( \Sigma_s^{-1} W \xi_t \xi_t^T W^T \right) \right\}.
\]
Feature Space MLLR (fMLLR)

Maximum Likelihood Estimation

Rewrite in terms of sufficient statistics:

\[
W_{ML} = \arg \max_{W} \left\{ T \log \det A + \text{Trace} \left( W \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \xi_t \mu_s^T \Sigma_s^{-1} \right) \right\} \\
- \frac{1}{2} \sum_{i,j=1}^{D} \sum_{k,l=1}^{D+1} W_{jk} W_{il} \cdot \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \frac{1}{\sigma^2_{s,i}} \xi_t, k \xi_t, l
\]

\[
\text{diagonal covar.} = \arg \max_{W} \left\{ T \log \det A + \sum_{i=1}^{D} \sum_{k=1}^{D+1} W_{ik} \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \xi_t, k \mu_s, i \frac{1}{\sigma^2_{s,i}} \right\} \\
- \frac{1}{2} \sum_{i=1}^{D} \sum_{k,l=1}^{D+1} W_{ik} W_{il} \cdot \sum_{s} \sum_{t=1}^{T} \gamma_s(t) \frac{1}{\sigma^2_{s,i}} \xi_t, k \xi_t, l
\]

\[
= Z_{ki} (3^{rd} \text{ord. tensor/matrix } \forall i)
\]

\[
=: G_{kl}^{(i)} (3^{rd} \text{ord. tensor/matrix } \forall i)
\]
Feature Space MLLR (fMLLR)

Maximum Likelihood Estimation

Case of diagonal covariances $\Sigma_{s,ij} = \sigma_{s,i}^2 \delta_{ij}$:

$$W_{\text{ML}} = \arg \max_W \left\{ T \log \det A + \sum_{i=1}^{D} \sum_{k=1}^{D+1} W_{ik} Z_{ki} - \frac{1}{2} \sum_{i=1}^{D} \sum_{k,l=1}^{D+1} W_{ik} W_{il} G_{kl}^{(i)} \right\} ,$$

with sufficient statistics:

$$G^{(i)} = \sum_{s}^{T} \sum_{t=1}^{T} \frac{\gamma_s(t)}{\sigma^2_{s,i}} \xi_t \xi_t^T$$

$$Z = \sum_{s}^{T} \sum_{t=1}^{T} \gamma_s(t) \xi_t \mu_s^T \Sigma_s^{-1}$$

- No closed form solution for $W$, iterative methods needed.
- Row-wise iterative solution in [Gales+@Cambridge 1998].
Feature Space MLLR (fMLLR)  
Dimension Reducing Adaptation

**Dimension reducing fMLLR [Lööf et al. @RWTH 2007]**

- **Goal:** Utilize more context information from features.
- Replace LDA matrix (instead of applying fMLLR after LDA).
- Non quadratic matrix \( \Rightarrow \) Derivation as for fMLLR not possible.
- Instead of ML, use ratio of acoustic model likelihood and a global **competing model** likelihood.
  - Competing model: single Gaussian trained on adaptation data,
    - Parameters: \( \mu', \Sigma' \) (in transformed feature space).

**In essence:** How much better do the transformed features fit the target acoustic model, than it fits a global single Gaussian?
Feature Space MLLR (fMLLR)

Dimension Reducing fMLLR

Consider adaptation data likelihood for global single Gaussian:

$$\log P(W_rx_1^T | \mu', \Sigma') = \sum_{t=1}^{T} \log \mathcal{N}(W_rx_t | \mu', \Sigma')$$

$$= \sum_{t=1}^{T} \left[ -\frac{1}{2} \log \det(2\pi \Sigma') - \frac{1}{2} (W_rx_t - \mu')^T \Sigma'^{-1} (W_rx_t - \mu') \right]$$

$$= -\frac{T}{2} \log \left[ (2\pi)^D \det(\Sigma') \right]$$

$$- \frac{1}{2} \text{Trace} \left[ \sum_{t=1}^{T} (W_rx_t - \mu')(W_rx_t - \mu')^T \cdot \Sigma'^{-1} \right]$$

$$= \text{Trace}(T \Sigma' \Sigma'^{-1}) = T \cdot \text{Trace} I_D = TD$$

$$= -\frac{T}{2} \log \det \Sigma' - \frac{TD}{2} \log 2\pi - \frac{TD}{2}$$
Feature Space MLLR (fMLLR)

Dimension Reducing fMLLR

Maximize likelihood ratio:

\[ \hat{W}_r = \arg \max_{W_r} \left\{ - \log P(W_r x_1^T | \mu', \Sigma') + \log P(W_r x_1^T | M, w_1^N) \right\} \]

with

\[ \log P(W_r x_1^T | \mu', \Sigma') = -\frac{T}{2} \log \det \Sigma' - \frac{TD}{2} \log 2\pi - \frac{TD}{2} \]

\[ = -\frac{T}{2} \log \det [W_r \Sigma W_r^T] + \text{const}(W_r) \]

with covariance \( \Sigma \) of untransformed features: \( \Sigma' = W_r \Sigma W_r^T \).

Substitute into maximum likelihood ratio:

\[ \hat{W}_r = \arg \max_{W_r} \left\{ \frac{T}{2} \log \det(W \Sigma W^T) + \log P(W_r x_1^T | M, w_1^N) \right\} \]

▷ Further derivation similar to regular fMLLR, details in [Lööf+@RWTH 2007]
Feature Space MLLR (fMLLR)

References

- M. J. F. Gales, Maximum Likelihood Linear Transformations for HMM-based Speech Recognition, CSL Apr. 1998.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Speaker Adaptive Training (SAT)

Introduction

- Adaptation compensates for speaker differences in recognition.
- But: We also have speaker differences in training corpus.
- Question: How can we compensate for both these differences?

Speaker-Adaptive Normalization

- Apply transform on training data also.
- Model training using transformed acoustic features.

Speaker-Adaptive Adaptation

- Interaction between model and transform requires simultaneous model and transform parameter training.
- Cannot simply retrain model
  → modified acoustic model training necessary.
Speaker Adaptive Training

VTLN and fMLLR

Training Procedure

1. Estimate speaker independent acoustic model $M_{SI}$ as normal.
2. Decide on / estimate a target model $M_T$ for adaptation estimation.
3. Use $M_T$ to estimate VTLN or fMLLR adaptation supervised for each speaker in training.
4. Transform features using the estimated adaptation.
5. Train speaker adaptive model $M_{SAT}$ on transformed features, starting from $M_{SI}$.

Recognition Procedure

1. First pass recognition using speaker independent acoustic model $M_{SI}$.
2. Estimate VTLN or fMLLR adaptation unsupervised using target model $M_T$.
3. Transform features using the estimated adaptation.
4. Second pass recognition using speaker adaptive model $M_{SAT}$. 
Original SAT was estimated iteratively, where in each iteration both acoustic model and adaptation parameters were reestimated [Anastasakos\textsuperscript{+}@BBN 1996].

**Problem:** interaction between acoustic model complexity and adaptation.

A simple, single Gaussian target model is known to lower WER both for VTLN [Welling\textsuperscript{+}@RWTH 1999] and fMLLR [Stemmer\textsuperscript{+}@FBK 2005].
Speaker Adaptive Training

MLLR

- Iterative re-estimation of MLLR, Gaussian means $\mu_s$, and Gaussian covariances $\Sigma_s$ [Anastasakos\textsuperscript{+} @ BBN 1996].
- MLLR parameters $A_r$ and $b_r$ estimated as usual on best mode.
- Re-estimation for Gaussian parameters using maximum likelihood:

$$
\begin{align*}
\mu_s &= \left( \sum_{t} \gamma_s(t) A_{r(t)}^T \Sigma^{-1}_s A_{r(t)} \right)^{-1} \left( \sum_{t} \gamma_s(t) A_{r(t)}^T \Sigma^{-1}_s (x_t - b_r(t)) \right) \\
\Sigma_s &= \text{Same as for standard ML but using SAT mean}
\end{align*}
$$

- **Drawback**: Require $D \times D$ accumulator for each mean $\mu_s$.
- 800k Gaussians, $D=45$, 64 bit float $\Rightarrow$ 12 GB accumulator.
Speaker Adaptive Training

References

- L. Welling and S. Kanthak and H. Ney, Improved Methods for Vocal Tract Normalization, ICASSP 1999
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Examples and Results

Experimental setup

All examples using RWTH TC-STAR 2006 English ASR systems:

- Training corpus: 88h.
- Development and Test sets: 3.2h each.
- MFCC + voicedness feature.
- One pass, Fast VTLN.
- LDA dimension: $45 \times 153$.
- ML trained acoustic models, approx. 900k Gaussians.
- Two pass system: fMLLR and MLLR unsupervised using first best output.
- **Note:** some differences between systems – direct WER comparison between slides not always possible.
Examples and Results
Basic Methods – Progressive Improvement

<table>
<thead>
<tr>
<th>Method</th>
<th>Dev</th>
<th>Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>18.5</td>
<td>-</td>
</tr>
<tr>
<td>+VTLN+voice</td>
<td>17.2</td>
<td>14.4</td>
</tr>
<tr>
<td>+fMLLR</td>
<td>15.7</td>
<td>-</td>
</tr>
<tr>
<td>+MLLR</td>
<td>14.0</td>
<td>11.8</td>
</tr>
</tbody>
</table>

- Different adaptation methods can be combined.
- **Here**: Each method gives an additional improvement – not always the case.
**Examples and Results**

Different combinations

<table>
<thead>
<tr>
<th>fMLLR</th>
<th>MLLR</th>
<th>SAT</th>
<th>Proj.</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dev</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>16.4</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>15.1</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td></td>
<td></td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>14.4</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>14.4</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>13.8</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>13.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>13.3</td>
</tr>
</tbody>
</table>

- **Here:** Baseline includes VTLN.
- fMLLR Improvement: 8 – 12% relative.
- Additional improvement MLLR: 7 – 8% relative (from MLLR alone more).
- Improvement from SAT: 5 – 8% relative.
- **Total** improvement (over VTLN): 19 – 23% rel.
Examples and Results
Use of Confidence Measures

- Baseline: fMLLR SAT using dimension reducing transforms.
- Confidence measure for adaptation: instead of using best recognized transcription from first pass, use all word graph hypotheses weighted by confidence measure.
- Goal: Evaluate different model adaptation schemes, with and without confidence measures.

<table>
<thead>
<tr>
<th></th>
<th>Dev</th>
<th>Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Conf</td>
</tr>
<tr>
<td>1st pass</td>
<td>16.3</td>
<td>–</td>
</tr>
<tr>
<td>fMLLRP SAT</td>
<td>14.2</td>
<td>–</td>
</tr>
<tr>
<td>MAP</td>
<td>14.1</td>
<td>13.0</td>
</tr>
<tr>
<td>MLLR</td>
<td>13.2</td>
<td>12.9</td>
</tr>
<tr>
<td>Shift-MLLR</td>
<td>13.2</td>
<td>12.7</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation
   6.1 Introduction
   6.2 Audio Segmentation
   6.3 Speaker Clustering
   6.4 Vocal Tract Length Normalization (VTLN)
   6.5 Maximum Likelihood Linear Regression (MLLR)
   6.6 Feature Space MLLR (fMLLR)
   6.7 Speaker Adaptive Training
   6.8 Examples and Results
   6.9 Summary

7. Discriminative Training
Summary

▶ Speaker adaptation important to obtain good performance in speaker independent speech recognition.
▶ Approach and parameterization usually depend on:
  ▶ supervision (yes/no),
  ▶ recognition mode (on-/offline),
  ▶ amount of adaptation data.
▶ Feature and model based approaches usually are additive.
▶ Unsupervised adaptation: 20% relative word error rate improvement possible.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Introduction

Motivation

**Aim** of discriminative methods: improve class separation

- standard maximum likelihood (ML) training: maximize reference class conditional $p_\theta(x|c)$
- maximum mutual information (MMI) training: maximize reference class posterior $p_\theta(c|x) = \frac{p(c) \cdot p_\theta(x|c)}{\sum_{c'} p(c') \cdot p_\theta(x|c')}$

**Where’s the difference?**

- **Ideally**: (almost) no difference! In case of infinite training data and correct model assumptions, the true probabilities are obtained in both cases. They lead to equal decisions, provided the class prior $p(c)$ is known. (Proof: model free optimization.)
- **ML training**: classes are handled independently, therefore decision boundaries are not considered explicitly in training.
- in **MMI training** and generally in discriminative training, the reference class directly competes against all other classes, decision boundaries become relevant in training.
In practice, model assumptions are incorrect, and training data is limited. Here discriminative training can be beneficial.

**Example:** a two class problem (with pooled covariance matrix)

Clearly, in case of ML training, the outlier deteriorates the decision boundary, whereas MMI training registers the minor importance of the outlier.

MMI captures decision boundary, although model assumption does not fit in second case (pooled covariance).
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups

Schlüter/Ney: Advanced ASR
August 5, 2010
Overview

Questions:
- Which discriminative criterion to take?
- Relation to decision rule and evaluation measure?
- How to optimize criterion?
- Efficiency?
- Influence of modeling?
- Uniqueness of solution?
- Generalization?

Bottomline:
- How to utilize available training material to obtain optimum recognition performance?
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Training Criteria

Notation

\( X_r \)
- sequence \( x_{r,1}, x_{r,2}, \ldots, x_{r,T_r} \) acoustic observation vectors

\( W_r \)
- spoken word sequence \( w_{r,1}, w_{r,2}, \ldots, w_{r,N_r} \) in training utterance \( r \)

\( W \)
- any word sequence

\( p(W) \)
- language model probability, supposed to be given

\( p_\theta(X_r|W) \)
- acoustic emission probability/acoustic model

\( \theta \)
- set of all parameters of the acoustic model

\( M_r \)
- set of competing word sequences to be considered

\( f \)
- smoothing function
Training Criteria

General Approach

- input: training data and stochastic model $p_\theta(X, W)$ with free model parameters $\theta$

- output: “optimal” model parameters $\hat{\theta}$

- optimality defined via training criterion

\[
\hat{\theta} := \arg \max_\theta \{ F(\theta) \}
\]

Unified training criterion [Macherey$^+$ 2005]

\[
F(\theta) = \sum_{r=1}^{R} f \left( \log \left( \frac{\sum_W p(W) p_\theta(X_r|W) \cdot A(W, W_r)}{\sum_{W \in M_r} p(W) p_\theta(X_r|W)} \right) \right)
\]

- covers maximum mutual information (MMI), minimum classification error (MCE), minimum phone/word error (MPE/MWE)

- control set $M_r$ of competing hypotheses, cost function, smoothing function, scaling of models (not shown)
Training Criteria
Probabilistic Training Criteria

Objective

▶ find good estimate of probability distribution
▶ optimality regarding error via Bayes’ decoding
  (asymptotic w.r.t. amount of training data)
Training Criteria

Maximum Likelihood (ML)

- optimization of joint probability

\[
\arg \max_{\theta} \sum_r \log (p(W_r) p_{\theta}(X|W_r)) = \arg \max_{\theta} \sum_r \log p_{\theta}(X_r|W_r)
\]

- Tutorial on HMM [Rabiner 1989].
- Maximization of probability of reference word sequences (classes).
- Model correctness important.
- HMM: maximization for each class separately.
- Neglects competing classes.
- Expectation-maximization: local convergence guaranteed.
- Estimation efficient, easily parallelizable.
Training Criteria

Maximum Mutual Information (MMI)

- Optimization of conditional probability

\[
\arg \max_{\theta} \sum_r \log p_{\theta}(W_r|X_r) = \arg \max_{\theta} \sum_r \log \frac{p(W_r)p_{\theta}(X_r|W_r)}{\sum_V p(V)p_{\theta}(X_r|V)}
\]

- Considers competing classes and therefore decision boundaries
- Necessitates set of competing classes on training data.
- Optimization for standard modeling (HMMs, mixture distributions): only gradient descent or similar.
- Optimization using log-linear modeling: convex problem
- 1st application of MMI for ASR using discrete HMMs [Bahl+ 1986]:
  - 2000 isolated words, 18% rel. improvement in word error rate.
- MMI: discrete and continuous probability densities [Brown 1987]:
  - isolated E-set letters, 18% rel. improvement in recognition rate.
- MMI: discrete & continuous probability densities [Normandin 1991]:
  - digit strings, up to 50% rel. improvement in string error rate.
Training Criteria

Error-Based Training Criteria

Objective: Optimize some error measure directly, e.g.:

- Empirical recognition error on training data
  - Advantage: direct relation to decision rule
  - Problem: non-differentiable training criterion, use of differentiable approximations in practice
  - Problem: ASR classes (words/word sequences) difficult to handle

- Model-based expected error on training data
  - Advantage: word or phoneme error easy to handle
  - Usually, approximated word/phoneme error, but correct edit distance also is viable [Heigold+ 2005]
  - Relation to decision rule less straight-forward.
  - Over-training and generalization becomes an issue (→ regularization, margin)
Training Criteria
Minimum classification error (MCE)

- For ASR: minimization of smoothed empirical sentence error [Juang & Katagiri 1992, Chou\(^+\) 1992].

\[
\arg \min_{\theta} \frac{1}{R} \sum_{r=1}^{R} \frac{1}{1 + \left[ \frac{\sum_{W \neq W_r} p_\theta^\alpha(X_r | W) \cdot p^\alpha(W)}{p_\theta^\alpha(X_r | W_r) \cdot p^\alpha(W_r)} \right]^{2\varrho}}^2
\]

- Smoothing parameters \(\alpha\) and \(\varrho\).
- Upper bound to Bayes’ error rate for any acoustic model [Schlüter\(^+\) 2001]
- Lesser effect of incorrect model assumptions.
Training Criteria

Minimum word/phone error (MWE/MPE)

- minimization of model-based expected word/phone error on training data [Povey & Woodland 2002]

\[
\arg \max_\theta \sum_{r=1}^R \frac{\sum_W A(W, W_r)p(W)p_\theta(X_r|W)}{\sum_W p(W)p_\theta(X_r|W)}
\]

- Criterion: maximum expected accuracy \(A(W, W_r)\).
- Accuracy usually approximate, but exact case based on edit (Levenshtein) distance also possible [Heigold 2005].
- Regularization (e.g. I-smoothing [Povey & Woodland 2002]) necessary due to overtraining.
- Usually better than MMI and MCE.
Training Criteria

Practical Issues

- Importance of language model in **training of acoustic model**.
- Relative and absolute scaling of language and acoustic model in training.
- Necessity for recognition of training data.
- Efficient calculation of discriminative training statistics using word lattices.
Training Criteria

Language Models for Discriminative Training

Potential Importance of Language Model Choice:

▶ language model for recognition of alternative word sequences
▶ language model dependence of discriminative training criterion itself
▶ interaction of language model of acoustic model parameters

Correlation hypothesis:

Only those acoustic models need optimization, which even together with a language model do not sufficiently discriminate.

→ language model choice would correlate for training and recognition.

Masking hypothesis:

Language model usually largely improves recognition accuracy and might mask deficiencies of the acoustic models.

→ suboptimal language models for training would give better performance.
Discriminative training includes language model. In training, unigram language model usually leads to the best word error rates [Schütler et al. 1999] (WSJ 5k):

<table>
<thead>
<tr>
<th>language models</th>
<th>criterion</th>
<th>word error rates [%]</th>
<th>dev</th>
<th>eval</th>
<th>dev &amp; eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>bi</td>
<td>ML</td>
<td>6.91</td>
<td>6.78</td>
<td>6.86</td>
<td></td>
</tr>
<tr>
<td>zero</td>
<td>MMI</td>
<td>6.71</td>
<td>6.03</td>
<td>6.41</td>
<td></td>
</tr>
<tr>
<td>uni</td>
<td></td>
<td>6.59</td>
<td>6.00</td>
<td>6.33</td>
<td></td>
</tr>
<tr>
<td>bi</td>
<td></td>
<td>6.71</td>
<td>6.20</td>
<td>6.48</td>
<td></td>
</tr>
<tr>
<td>tri</td>
<td></td>
<td>6.87</td>
<td>6.54</td>
<td>6.72</td>
<td></td>
</tr>
<tr>
<td>tri</td>
<td>ML</td>
<td>4.82</td>
<td>4.11</td>
<td>4.51</td>
<td></td>
</tr>
<tr>
<td>zero</td>
<td>MMI</td>
<td>4.63</td>
<td>4.05</td>
<td>4.38</td>
<td></td>
</tr>
<tr>
<td>uni</td>
<td></td>
<td>4.30</td>
<td>3.64</td>
<td>4.01</td>
<td></td>
</tr>
<tr>
<td>bi</td>
<td></td>
<td>4.48</td>
<td>3.94</td>
<td>4.24</td>
<td></td>
</tr>
<tr>
<td>tri</td>
<td></td>
<td>4.58</td>
<td>4.00</td>
<td>4.33</td>
<td></td>
</tr>
</tbody>
</table>

-8%  -11%
Training Criteria
Scaling of likelihoods

- recognition: absolute scaling of likelihoods irrelevant (language model scale vs. acoustic model scale)
- absolute scaling does have impact on word posterior calculation [Wessel 1998, Woodland & Povey 2000]
- use language model scale $\beta$ also in training:

$$p(X, W) = p(W)^\beta p_\theta(X|W)$$

- replace $p(X, W)$ with:

$$p(X, W)^\gamma = p(W)^{\beta\gamma} p_\theta(X|W)^\gamma \quad \text{for} \quad \gamma \in [0, 1]$$

- optimum approx. for $\gamma = \frac{1}{\beta}$, i.e. use

$$p(X, W)^{\frac{1}{\beta}} = p(W)p_\theta(X|W)^{\frac{1}{\beta}}$$

- For simplicity here usually omitted in equations.
Training Criteria
Competing Word Sequences

- Problem: Exponential number of competing word sequences.
- Competing word sequences need to be estimated:
  - Hypothesis-generation on training data using recognizer.
  - Initial lattice generation using recognizer sufficient.
  - Later acoustic model rescoring constrained to lattice.
- Representation and processing of competing word sequences.
  - Efficient algorithms to process word lattices.
  - Generic implementation: weighted finite state transducers.
Training Criteria

Competing Word Sequences

History:

- best recognized word sequence for MMI (Corrective Training) [Normandin 1991]:
  - considers incorrectly recognized training sentences only
- best *incorrectly* recognized word sequence for MCE [Juang & Katagiri 1992]:
  - interpretation of smoothed sentence error still valid
- $N$-best recognized word sequences for MMI [Chow 1990]:
  - continuous speech recognition, 1000 words
  - only minor improvements in word error rate
- word graphs from recognition for MMI training [Valtchev 1997]:
  - large vocabulary, 64k words
  - efficient implementation
  - 5-10% relative improvement in word error rate
Training Criteria

Comparative Experimental Results

<table>
<thead>
<tr>
<th>Crit.</th>
<th>SieTill Test</th>
<th>WSJ 5k</th>
<th>EPPS English</th>
<th>Mandarin BN/BC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dev</td>
<td>Evl</td>
<td>Dev06</td>
</tr>
<tr>
<td>ML</td>
<td>1.81</td>
<td>4.55</td>
<td>3.74</td>
<td>14.4</td>
</tr>
<tr>
<td>MMI</td>
<td>1.79</td>
<td>4.07</td>
<td>3.53</td>
<td>13.8</td>
</tr>
<tr>
<td>MCE</td>
<td>1.69</td>
<td>4.02</td>
<td>3.47</td>
<td>13.8</td>
</tr>
<tr>
<td>MWE</td>
<td>1.69</td>
<td>3.98</td>
<td>3.44</td>
<td>13.8</td>
</tr>
<tr>
<td>MPE</td>
<td>1.69</td>
<td>4.17</td>
<td>3.62</td>
<td>13.4</td>
</tr>
</tbody>
</table>

- SieTill [Schlüter 2000]
- WSJ 5k [Macherey 2010]
- EPPS/broadcasts [Heigold 2010]
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. **Discriminative Training**
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Parameter Optimization

Motivation

**Goal:** optimization method for discriminative training criteria $F(\theta)$ w.r.t. set of parameters $\theta$ which provides *reasonable* convergence.

- Various approaches, e.g.:
  - extended *Baum-Welch* (EBW) [Normandin 1991]
  - gradient descent, study: e.g. [Valtchev 1995]
  - MMI with log-linear models: generalized iterative scaling (GIS)
  - generalization of GIS to log-linear models with hidden variables and further criteria like MPE and MCE [Heigold+ 2008a]

- Problems:
  - robust setting of step sizes/iteration constants (EBW and gradient descent),
  - convergence speed (especially GIS).
Parameter Optimization
Extended Baum-Welch

- Motivated by a growth transformation [Gopalakrishanan+ 1991]
- Widely used for discriminative training of Gaussian mixture HMMs, e.g. [Normandin 1991, Valtchev+ 1997, Schlüter 2000, Woodland & Povey 2002]
- Highly optimized heuristics for finding right order of magnitude for iteration constants.
- Training of Gaussian mixture HMMs: require positive variances to obtain estimate for iteration constants.
Parameter Optimization

Gradient descent

Follow gradient to optimize parameter:

\[ \hat{\theta} = \theta + \gamma \nabla_{\theta} F_{\theta} \]

Step sizes:
- heuristic, e.g. for MCE [Chou\textsuperscript{+} 1992])
- by comparison to EBW [Schlüter 2000]

Convergence:
- local optimum
- better convergence: general purpose approaches, e.g. Qprop, Rprop, or L-BFGS, for experimental comparisons see [McDermott & Katagiri 2005, McDermott\textsuperscript{+} 2007, Gunawardana\textsuperscript{+} 2005, Mahajan\textsuperscript{+} 2006]
Parameter Optimization

Rprop [Riedmiller & Braun 1993]

General purpose gradient based optimization:

- assume iteration $n$
- parameter update:

$$
\theta_i^{(n+1)} = \theta_i^{(n)} + \gamma_i^{(n)} \text{sign} \left( \frac{\partial F(\theta^{(n)})}{\partial \theta_i} \right)
$$

- update of step sizes $\gamma_i^{(n)}$:

$$
\gamma_i^{(n+1)} = \begin{cases} 
\min \{ \gamma_i^{(n)} \cdot \eta^+, \gamma_{\text{max}} \} & \text{if } \frac{\partial F(\theta^{(n)})}{\partial \theta_i} \cdot \frac{\partial F(\theta^{(n-1)})}{\partial \theta_i} > 0 \\
\max \{ \gamma_i^{(n)} \cdot \eta^-, \gamma_{\text{min}} \} & \text{if } \frac{\partial F(\theta^{(n)})}{\partial \theta_i} \cdot \frac{\partial F(\theta^{(n-1)})}{\partial \theta_i} < 0 \\
\gamma_i^{(n)} & \text{otherwise}
\end{cases}
$$

- $\eta^+ \in (1, \infty)$, $\eta^- \in (0, 1)$
Parameter Optimization

Formal gradient of MMI

- notation:
  - $W$: word sequence $w_1, \ldots, w_N$
  - $r$: index of training segment/utterance given by $(X_r, W_r)$
  - $X_r$: acoustic observation vector sequence $x_{r1}, \ldots, x_{rT}$
  - $W_r$: reference/spoken word sequence $w_{r1}, \ldots, w_{rN}$
  - $s^T_1$: HMM state sequence $s_1, \ldots, s_T$

- MMI training criterion:

$$F_{\text{MMI}}(\theta) = \sum_r \log \left( \frac{p(W_r)p_\theta(X_r|W_r)}{\sum_W p(W)p_\theta(X_r|W)} \right)$$

$$= \sum_r \left( \log p(W_r)p_\theta(X_r|W_r) - \log \sum_W p(W)p_\theta(X_r|W) \right)$$

- acoustic model (HMM):

$$p_\theta(X_r, W) = \sum_{s^T_1} \prod_{t=1}^{T_r} p(s_t|s_{t-1})p_\theta(x_{rt}|s_t)$$
Parameter Optimization

Derivative of MMI w.r.t. Parameter

Gradient of MMI criterion:

\[
\nabla_{\theta} F_{\text{MMI}}(\theta) = \sum_{r} \left( \nabla_{\theta} \log p_{\theta}(X_r | W_r) \right.
\]

\[
- \sum_{W} p(W)p_{\theta}(X_r | W) \nabla_{\theta} \log p_{\theta}(X_r | W)
\]

\[
\left. \right) \frac{\sum_{W'} p(W')p_{\theta}(X_r | W')} \sum_{W'} p(W')p_{\theta}(X_r | W') \right)
\]

For efficient evaluation, consider derivative of acoustic model, \( \nabla_{\theta} \log p_{\theta}(X_r | W) \).
Parameter Optimization

Derivative of Acoustic Model

\[ \nabla_{\theta} \log p_{\theta}(X_r, W) = \nabla_{\theta} \log \sum_{s_t \colon W}^{T_r} \prod_{t=1}^{T_r} p_{\theta}(x_{rt} \mid s_t) p(s_t \mid s_{t-1}) \]

\[ = \sum_{t=1}^{T_r} \sum_{s \colon 1}^{T_r} \left( \nabla_{\theta} \log p_{\theta}(x_{rt} \mid s_t) \right) \cdot \frac{\sum_{s_t \colon W}^{T_r} p_{\theta}(x_{rt} \mid s_t) p(s_t \mid s_{t-1})}{\sum_{s_t \colon W}^{T_r} \prod_{t=1}^{T_r} p_{\theta}(x_{rT} \mid s_T) p(s_T \mid s_{T-1})} \]

\[ = \sum_{t=1}^{T_r} \sum_{s} \left( \nabla_{\theta} \log p_{\theta}(x_{rt} \mid s) \right) \cdot \left( \sum_{s}^{T_r} p_{\theta}(X_r, s^{T_r} \mid W) \right) \cdot \nabla_{\theta} \log p_{\theta}(x_{rt} \mid s) \]

\[ = \sum_{t=1}^{T_r} \sum_{s} \gamma_{rt}(s \mid W) \cdot \nabla_{\theta} \log p_{\theta}(x_{rt} \mid s) \]

with the word sequence conditioned state posterior (occupancy):

\[ \gamma_{rt}(s \mid W) = \frac{\sum_{s}^{T_r} p_{\theta}(X_r, s^{T_r} \mid W)}{p_{\theta}(X_r \mid W)} = p_{\theta, t}(s \mid X_r, W) \]
resubstitute derivative of acoustic model into derivative of MMI criterion:

\[
\nabla_\theta F_{\text{MMI}}(\theta) = \sum_r \sum_{t=1}^{T_r} \sum_s \left( \nabla_\theta \log p_\theta(x_{rt}|s) \right) \cdot \left( \gamma_{rt}(s|W_r) - \frac{\sum_W p(W)p_\theta(X_r|W)\gamma_{rt}(s|W)}{\sum_{W'} p(W')p_\theta(X_r|W')} \right)
\]

\[
= \sum_r \sum_{t=1}^{T_r} \sum_s \left( \nabla_\theta \log p_\theta(x_{rt}|s) \right) \cdot \left( \gamma_{rt}(s|W_r) - \gamma_{rt}(s) \right)
\]

with the general state posterior (occupancy):

\[
\gamma_{rt}(s) = \frac{\sum_W p(W)p_\theta(X_r|W)\gamma_{rt}(s|W)}{\sum_{W'} p(W')p_\theta(X_r|W')} = p_{\theta,t}(s|X_r)
\]
Parameter Optimization

Efficient Calculation of State Occupancies

- Efficient calculation of spoken word sequence conditional state occupancy $\gamma_{rt}(s|W_r)$: forward-backward state probabilities on trellis of word sequence.
- Efficient calculation of general state occupancy $\gamma_{rt}(s)$: forward-backward probabilities on trellis of word lattice.

**Viterbi approximation:**

- $\gamma_{rt}(s|W) = \delta_{s,s_{rt}(W)}$ with forced alignment $S_r(W) = s_{r1}(W), \ldots, s_{rT_r}(W)$ of spoken word sequence
- Assume a (word) lattice $\mathcal{M}_r$ for utterance $r$, with edges $\omega$ representing a word $w(\omega)$ (in context) with start time $t_{s}(\omega)$ and end time $t_{e}(\omega)$, and a corresponding forced alignment $s_{t_{s}}^{t_{e}}(\omega)$. An edge sequence $\mathcal{W} \in \mathcal{M}_r$ then corresponds to the word sequence $W(\mathcal{W})$. Consequently, the language model and acoustic model can also be defined for an edge sequence, which then might specify word boundaries, phonetic and language model context.
Parameter Optimization

Word Posterior Probabilities

For the general state occupancy in Viterbi approximation we obtain:

$$\gamma_{rt}(s) = \frac{\sum_{W} p(W) p_{\theta}(X_r|W) \delta_{s,s_{rt}}(W)}{p_{\theta}(X_r)}$$

$$= \sum_{\omega} \delta_{s,s_{rt}}(\omega) \frac{\sum_{W: \omega \in W} p(W) p_{\theta}(X_r|W)}{p_{\theta}(X_r)}$$

$$= \sum_{\omega} \delta_{s,s_{rt}}(\omega) p(\omega|X_r)$$

with the edge (or word in context) posterior

$$p(\omega|X_r) = \sum_{W: \omega \in W} \frac{p(W) p_{\theta}(X_r|W)}{p_{\theta}(X_r)}$$

A forward-backward algorithm is used to efficiently compute edge (word in context) posterior probabilities using word lattices.
Parameter Optimization

Formal gradient of MPE

- \( A(W, W_r) \): accuracy (negated error) between string \( W \) and \( W_r \)
- Example (MPE): approximate phone accuracy [Povey & Woodland 2002]
- Expectation of accuracy:

\[
E_\theta[A(\cdot, W_r)] := \sum_{W} A(W, W_r) \cdot \frac{p(W)p_\theta(X_r|W)}{\sum_{W'} p(W')p_\theta(X_r|W')}
\]

- MPE training criterion:

\[
F_{\text{MPE}}(\theta) = \sum_{r} E_\theta[A(\cdot, W_r)]
\]
Parameter Optimization

Derivative of MPE w.r.t. Parameter

Derivative of MPE criterion:

$$\nabla_\theta p_\theta(X_r|W) = p_\theta(X_r|W) \cdot (\nabla_\theta \log p_\theta(X_r|W))$$

$$\nabla_\theta F_{\text{MPE}}(\theta) = \sum_r \sum_W (A(W, W_r) - E_\theta[A(\cdot, W_r)]) \cdot (\nabla_\theta \log p_\theta(X_r|W)) \cdot \frac{p(W)p_\theta(X_r|W)}{\sum_{W'} p(W')p_\theta(X_r|W')}$$

For efficient evaluation, consider derivative of acoustic model:

$$\nabla_\theta \log p_\theta(X_r|W) = \sum_{t=1}^{T_r} \sum_s \left( \nabla_\theta \log p_\theta(x_{rt}|s) \right) \cdot \frac{\sum_{s_{1:T_r} : s_t = s} p_\theta(X_r, s_{1:T_r}|W)}{p_\theta(X_r|W)}$$
resubstitute derivative of acoustic model into derivative of MPE criterion:

$$\nabla_\theta F_{\text{MPE}}(\theta) = \sum_r \sum_{t=1}^{T_r} \sum_s (\nabla_\theta \log p_\theta(x_{rt}|s)) \cdot \tilde{\gamma}_{rt}(s)$$

with the general state accuracy:

$$\tilde{\gamma}_{rt}(s) = \sum_W \left( A(W, W_r) - E_\theta[A(\cdot, W_r)] \right) \cdot \frac{\sum_{s_1^{Tr} : s_t = s} p(W)p_\theta(X_r, s_1^{Tr}|W)}{\sum_{W'} p(W')p_\theta(X_r|W')}$$

which can be computed efficiently, similar to the case of general state occupancies.
In general:

- **assumption:** \( A(W, W_r) = \sum_{t=1}^{T_r} A(s_{rt}(W), s_{rt}(W_r)) \)
- **example:** approximate phone accuracy [Povey & Woodland 2002]
- **efficient calculation of general state accuracy** \( \tilde{\gamma}_{rt}(s) \): forward-backward accuracies on trellis of word lattice [Povey & Woodland 2002]
Parameter Optimization

Word Posterior Accuracies

For the general state accuracy we in Viterbi approximation obtain:

$$\tilde{\gamma}_{rt}(s) = \frac{\sum_{W} \left( A(W, W_r - E_\theta[A(\cdot, W_r)]) \right) \cdot p(W)p_\theta(X_r | W)\delta_{s,s_{rt}}(W)}{p_\theta(X_r)}$$

$$= \sum_\omega \delta_{s,s_{rt}}(\omega) \frac{\sum_{W:\omega \in W} \left( A(W, W_r - E_\theta[A(\cdot, W_r)]) \right) \cdot p(W)p_\theta(X_r | W)}{p_\theta(X_r)}$$

$$= \sum_\omega \delta_{s,s_{rt}}(\omega) \tilde{p}(\omega | X_r)$$

with the edge (or word in context) posterior accuracies

$$\tilde{p}(\omega | X_r) = \sum_{W:\omega \in W} \frac{\left( A(W, W_r - E_\theta[A(\cdot, W_r)]) \right) \cdot p(W)p_\theta(X_r | W)}{p_\theta(X_r)}$$

Later, an efficient way of computing edge (word in context) posterior accuracies using word lattices will be presented.
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Efficient Calculation of Discriminative Statistics

Forward/Backward Probabilities on Word Lattices

Let $\omega_s(\mathcal{W})$ and $\omega_e(\mathcal{W})$ be the first and last edge of a continuous edge sequence $\mathcal{W}$ on a word lattice.

Assume that the lattice fully encodes the language model context:

$$p(\mathcal{W}(\mathcal{W})) = p(\mathcal{W} = \omega_1^N) = \prod_{n=1}^{N} p(\omega_n|\omega_{n-1})$$

Let $\omega_{ri}$ and $\omega_{rf}$ be the initial and final edges of a word lattice for utterance $r$. Then define the following forward ($\Phi$) and backward ($\Psi$) probabilities on initial and final partial edge sequences on the word lattice respectively:

$$\Phi(\omega) = \sum_{\mathcal{W}: \omega_s(\mathcal{W})=\omega_{ri}} p(\mathcal{W})p_{\theta}(x_{r1}^{te(\mathcal{W})}|\mathcal{W})$$

$$\Psi(\omega) = \sum_{\mathcal{W}: \omega_s(\mathcal{W})=\omega} p(\mathcal{W})p_{\theta}(x_{r}^{Tr}_{ts(\mathcal{W})}|\mathcal{W})$$
Efficient Calculation of Discriminative Statistics

Forward/Backward Probabilities on Word Lattices

For the forward probability a recursion formulae can be derived by separating the last edge from the edge sequence in the summation and \( \prec \) denoting direct predecessor edges:

\[
\Phi(\omega) = \sum_{\mathcal{W}: \omega_s(\mathcal{W}) = \omega_{ri}} \sum_{\omega_e(\mathcal{W}) = \omega} p(\mathcal{W}) p_{\theta}(x_{r1}^{te(\mathcal{W})} | \mathcal{W})
\]

\[
= \sum_{\omega' \prec \omega} \sum_{\mathcal{W}': \omega_s(\mathcal{W}') = \omega_{ri}} \sum_{\omega_e(\mathcal{W}') = \omega'} p(\mathcal{W}') p(\omega | \omega') p_{\theta}(x_{r1}^{te(\mathcal{W}')} | \mathcal{W}') p_{\theta}(x_{rt}^{te(\omega)} | \omega)
\]

\[
= \sum_{\omega' \prec \omega} \Phi(\omega') p(\omega | \omega') p_{\theta}(x_{rt}^{te(\omega)} | \omega).
\]

Using this recursion formula, the forward probabilities can be calculated efficiently on word lattices.
Similar to the forward probabilities, a recursion formula can be derived for efficient calculation of the backward probabilities and \( \succ \) denoting direct successor edges:

\[
\Psi(\omega) = \sum_{\mathcal{W}: \omega_s(\mathcal{W}) \succ \omega} p(\mathcal{W}) p_\theta(x_r^{T_r}_{ts}(\omega)|\omega \mathcal{W})
\]

\[
= \sum_{\omega' \succ \omega} \sum_{\mathcal{W}': \omega_s(\mathcal{W}') \succ \omega'} p(\omega' | \omega) p(\mathcal{W'}) p_\theta(x_r^{t_e}_{ts}(\omega) | \omega) p_\theta(x_r^{T_r}_{ts}(\omega') | \omega' \mathcal{W}')
\]

\[
= \sum_{\omega' \succ \omega} p_\theta(x_r^{t_e}_{ts}(\omega) | \omega) p(\omega' | \omega) \Psi(\omega')
\]
Using the forward and backward probabilities, the edge/word posterior on a word lattice can be written as

\[
p(\omega | X_r) = \frac{\Phi(\omega) \sum_{\omega' \succ \omega} p(\omega' | \omega) \Psi(\omega')}{\Phi(\omega_{rf})}
\]

with \( p^\theta(X_r) = \Phi(\omega_{rf}) = \Psi(\omega_{ri}). \)

Word posterior probabilities follow naturally from MPE and similar discriminative training criteria. They also are the basis for confidence measures, which are used for unsupervised training, adaptation, or dialog systems. They are also part of approximate approaches to Bayes’ decision rule with word error cost, like confusion networks [Mangu+ 1999], or minimum frame word error [Wessel+ 2001a, Hoffmeister+ 2006].
Efficient Calculation of Discriminative Statistics
FB Probabilities: Generalization to WFSTs

Replace word lattice with WFST

- edge label: word with pronunciation
- weight of edge $\omega$: $p \leftarrow p(\omega | \omega') \cdot p_\theta(x_{t_e(\omega)} t_{s(\omega)} | \omega)$
- semiring: substitute arithmetic operations (multiplication, addition, inversion) with operations of probability semiring

<table>
<thead>
<tr>
<th>Semiring</th>
<th>$\mathbb{I}K$</th>
<th>$p \oplus p'$</th>
<th>$p \otimes p'$</th>
<th>0</th>
<th>1</th>
<th>$\text{inv}(p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>$\mathbb{I}R^+$</td>
<td>$p + p'$</td>
<td>$p \cdot p'$</td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{p}$</td>
</tr>
</tbody>
</table>

Example WFST from SieTill, $\forall r =$”drei sechs neun” (in red)
Efficient Calculation of Discriminative Statistics

FB Probabilities: Generalization to WFSTs

Forward probabilities ($pre(\omega) \in \mathcal{W}$ such that $pre(\omega) \prec \omega$)

$$\Phi(\omega) := \bigoplus_{\mathcal{W}: \omega_s(\mathcal{W}) = \omega_i, \omega \in \mathcal{W}} \bigotimes_{\omega_e(\mathcal{W}) = \omega} p(\omega | pre(\omega)) \otimes p_{\theta}(x_r^{te(\omega)} | \omega)$$

$$= \bigoplus_{\omega' \prec \omega} \Phi(\omega') \otimes p(\omega | \omega') \otimes p_{\theta}(x_r^{te(\omega)} | \omega)$$

Backward probabilities: similar

Using the forward and backward probabilities, the edge posterior on a WFST $\mathcal{X}_r$ can be written as

$$p(\omega | \mathcal{X}_r) = \Phi(\omega) \otimes \left( \bigoplus_{\omega' \succ \omega} p(\omega' | \omega) \otimes \Psi(\omega') \right) \otimes \text{inv}(\Phi(\omega_{rf}))$$
Efficient Calculation of Discriminative Statistics

Expectation Semiring

vector weight \((p, v)\) of edge \(\omega\) with

\[
\begin{align*}
\triangleright p &\leftarrow p(\omega|\omega') \cdot p_\theta(x_{r t_s}(\omega)|\omega) \\
\triangleright v &\leftarrow A(\omega) \cdot p \\
\end{align*}
\]

- accuracy of edge \(\omega\) such that \(\otimes_{\omega \in W} A(\omega) = A(W, W_r)\)
- approximate phone accuracy [Povey & Woodland 2002] can be decomposed in this way
- such a decomposition not possible in general

expectation semiring [Eisner 2001]:
vector semiring whose first component is a probability semiring

<table>
<thead>
<tr>
<th>Semiring</th>
<th>(\mathbb{R}\times\mathbb{R})</th>
<th>((p, v) \oplus (p', v'))</th>
<th>((p, v) \otimes (p', v'))</th>
<th>(0)</th>
<th>(\mathbb{R}^+)</th>
<th>inv((p, v))</th>
</tr>
</thead>
<tbody>
<tr>
<td>expectation</td>
<td>(\mathbb{R}\times\mathbb{R})</td>
<td>((p + p', v + v'))</td>
<td>((p \cdot p', p \cdot v' + p' \cdot v))</td>
<td>((0, 0))</td>
<td>((1, 0))</td>
<td>(\left(\frac{1}{p}, -\frac{v}{p^2}\right))</td>
</tr>
</tbody>
</table>
Efficient Calculation of Discriminative Statistics

Edge Posteriors & Expectation Semiring

probability semiring

▶ word posterior probabilities (see MMI derivative) identical to edge posteriors using probability semiring

\[ p(\omega | X_r) = p_{\text{probability}}(\omega | X_r) \]

▶ intuitive and classical result [Rabiner 1989]

expectation semiring

▶ word posterior accuracies (see MPE derivative) identical to \( \nu \)-component of edge posteriors using expectation semiring [Heigold+ 2008b]

\[ \tilde{p}(\omega | X_r) = p_{\text{expectation},\nu}(\omega | X_r) \]

▶ also use this identity to efficiently calculate
  ▶ derivative of unified training criterion
  ▶ covariance between two random additive variables (related to MPE derivative)
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics

7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Generalisation to Log-Linear Modeling

Transformation: Gaussian into Log-Linear Model

Definition of Models

assume feature vector \( x \in \mathbb{R}^D \) and class \( c \in \{1, \ldots, C\} \)

**Gaussian model** \( \mathcal{N}(x|\mu_c, \Sigma_c) \) with

- means \( \mu_c \in \mathbb{R}^D \)
- positive-definite covariance matrices \( \Sigma_c \in \mathbb{R}^{D\times D} \)

induces posterior \( p_\theta(c|x) \)

\[
\frac{p(c)\mathcal{N}(x|\mu_c, \Sigma_c)}{\sum_{c'} p(c')\mathcal{N}(x|\mu_{c'}, \Sigma_{c'})}
\]

- include priors \( p(c) \in \mathbb{R}^+ \)

**Log-linear model** with unconstrained parameters

\( \lambda_{c0} \in \mathbb{R} \)
\( \lambda_{c1} \in \mathbb{R}^D \)
\( \lambda_{c2} \in \mathbb{R}^{D\times D} \)

\[
\frac{\exp (x^T \lambda_{c2} x + \lambda_{c1}^T x + \lambda_{c0})}{\sum_{c'} \exp (x^T \lambda_{c'2} x + \lambda_{c'1}^T x + \lambda_{c'0})}
\]
Generalisation to Log-Linear Modeling

Transformation: Gaussian into Log-Linear Model

Comparison of terms quadratic, linear, and constant in observations $x$ leads to the transformation rules [Saul & Lee 2002, Gunawardana$^+$ 2005]:

1. $\lambda_{c2} = -\frac{1}{2} \Sigma_c^{-1}$
2. $\lambda_{c1} = \Sigma_c^{-1} \mu_c$
3. $\lambda_{c0} = -\frac{1}{2} (\mu_c^\top \Sigma_c^{-1} \mu_c + \log |2\pi \Sigma_c|) + \log p(c)$
Generalisation to Log-Linear Modeling

Transformation: Log-Linear into Gaussian Model

- invert transformation from Gaussian to log-linear model

1. $\Sigma_c = -\frac{1}{2} \lambda_{c2}^{-1}$
2. $\mu_c = \Sigma_c^{-1} \lambda_{c1}$
3. $p(c) = \exp \left( \lambda_{c0} + \frac{1}{2} (\mu_c^T \Sigma_c^{-1} \mu_c + \log |2\pi \Sigma_c|) \right)$

- problem: parameter constraints not satisfied in general
  - covariance matrices $\Sigma_c$ must be positive-definite
  - priors $p(c)$ must be normalized
- solution: model parameters for posterior are ambiguous
  e.g. for $\Delta \lambda_2 \in \mathbb{R}^{D \times D}$, $\Delta \lambda_0 \in \mathbb{R}$

$$\frac{\exp \left( x^T (\lambda_{c2} + \Delta \lambda_2) x + \lambda_{c1}^T x + (\lambda_{c0} + \Delta \lambda_0) \right)}{\sum_{c'} \exp \left( x^T (\lambda_{c'2} + \Delta \lambda_2) x + \lambda_{c'1}^T x + (\lambda_{c'0} + \Delta \lambda_0) \right)} = \frac{\exp \left( x^T \lambda_{c2} x + \lambda_{c1}^T x + \lambda_{c0} \right)}{\sum_{c'} \exp \left( x^T \lambda_{c'2} x + \lambda_{c'1}^T x + \lambda_{c'0} \right)}$$
Generalisation to Log-Linear Modeling

Transformation: Log-Linear into Gaussian Model

invert transformation rules for transformed log-linear model

1. $\Sigma_c = -\frac{1}{2}(\lambda_{c2} + \Delta\lambda_2)^{-1}$
2. $\mu_c = \Sigma_c^{-1}\lambda_{c1}$
3. $p(c) = \exp \left( (\lambda_{c0} + \Delta\lambda_0) + \frac{1}{2} (\mu_c^\top \Sigma_c^{-1} \mu_c + \log |2\pi \Sigma_c|) \right)$

use additional degrees of freedom to impose parameter constraints

- choose $\Delta\lambda_2 \in \mathbb{R}^{D \times D}$ such that $\lambda_{c2} + \Delta\lambda_2$ are negative-definite
- choose $\Delta\lambda_0$ such that $p(c)$ is normalized, i.e.,

$$\Delta\lambda_0 := -\log \sum_c \exp \left( \lambda_{c0} + \frac{1}{2} (\mu_c^\top \Sigma_c^{-1} \mu_c + \log |2\pi \Sigma_c|) \right)$$
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Convex Optimization

Motivation

Conventional approach:

- depends on initialization and choice of optimization algorithm
- spurious local optima (non-convex training criterion)
- many heuristics required
- i.e., involves much engineering work

“Fool-proof” approach:

- unique optimum (independent of initialization)
- accessibility of global optimum (convex training criterion)
- joint optimization of all model parameters, no parameters to be tuned
**Assumptions to cast HCRF into CRF**

- log-linear parameterization, e.g.
  \[ p(x|s) = \exp(x^\top \lambda_s x + \lambda_{s1} x + \lambda_{s0}) \] and \[ p(s'|s) = \exp(\alpha_{s's}) \]
- MMI-like training criterion
- alignment represents spoken sequence
- alignment of spoken sequence known and kept fixed
- use single densities with augmented features instead of mixtures
- exact normalization constant
Convex Optimization

Lattice-Based MMI

\[
F_{\text{lattice}}(\lambda) = \sum_r \frac{\sum_{s_1^T \in \mathcal{N}_r} p_\lambda(x_1^T r, s_1^T r)}{\sum_{s_1^T \in \mathcal{D}_r} p_\lambda(x_1^T r, s_1^T r)}
\]

- numerator word lattice \( \mathcal{N}_r \): state sequences \( s_1^T \) representing correct hypothesis
- denominator word lattice \( \mathcal{D}_r \): correct and competing state sequences, use word pair approximation and pruning
- non-convex

Word lattice
Fool-Proof MMI

\[
F_{\text{fool}}(\lambda) = \sum_r \frac{p_\lambda(\mathbf{x}_1^{T_r}, \hat{s}_1^{T_r})}{\sum_{s_1^{T_r} \in S_r} p_\lambda(\mathbf{x}_1^{T_r}, s_1^{T_r})}
\]

▷ consider only best state sequence \(\hat{s}_1^T\) in numerator, kept fixed
▷ sum over full state sequence network in denominator
▷ convex

Part of (pruned) HMM state network
Convex Optimization

Frame-Based MMI

\[ F_{\text{frame}}(\lambda) = \sum_{r} \sum_{t=1}^{T_r} \frac{p_\lambda(x_t, \hat{s}_t)}{\sum_{s=1}^{S} p_\lambda(x_t, s)} \]

- frame discrimination, cf. hybrid approach
- assume alignment for numerator \( s_1^T \), kept fixed
- summation over all HMM states \( s \in \{1, \ldots, S\} \) in denominator
- convex
These refinements do not break convexity:

- $\ell_2$-regularization
- margin term
Convex Optimization

Experimental Results

Initialization

Analyze effect of initial parameters on training.

- vary initialization for different training criteria
- experiments: digit strings (SieTill, German, telephone)
Convex Optimization

Read Speech (WSJ)

- 5k-vocabulary, trigram language model
- phone-based HMMs, 1,500 CART-tied triphones
- audio data: 15h (training), 0.4h (test)

- log-linear model with kernel-like features $f(x)$
  - first ($f_d(x) = x_d$) and second ($f_{dd'}(x) = x_d \cdot x_{d'}$) order features
  - cluster features: assume GMM of marginal distribution,
    $$p(x) = \sum_l p(x, l)$$

    $$f_i(x) = \begin{cases} p(l|x) & \text{if } p(l|x) \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

- starting from scratch (model) and linear segmentation
- frame-based MMI, with re-alignments
- details: [Wiesler 2009]
Convex Optimization

Read Speech (WSJ)

<table>
<thead>
<tr>
<th>Feature setup</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>First order features, monophones + second order features + $2^{10}$ cluster features + temporal context of size 9 + 1,500 CART-tied HMM states (triphones) + realignment</td>
<td>22.7</td>
</tr>
<tr>
<td>GHMM (ML) (MMI)</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Incorporation of Margin Concept

Motivation

Goal: incorporation of margin term into conventional training criteria

- replace likelihoods $p(W)p(X|W)$ with margin-likelihoods $p(W)p(X|W)\exp(-\rho A(W, W_r))$
- $A(W, W_r)$: accuracy between hypothesis $W$ and reference $W_r$
- interpretation (boosting): emphasize incorrect hypotheses by up-weighting
- interpretation (large margin): next slides

Margin

- low complexity task
- different loss functions
- convergence
- different parameterization
- local optima
- optimization

Margin in training is promising.

Individual contribution of margin in LVCSR training?
Incorporation of Margin Concept
Support Vector Machines (Hinge Loss)

Optimization problem for SVMs

\[
SVM(\lambda) = -\frac{C}{2} \| \lambda \|^2 - \sum_{r=1}^{R} l(W_r, d_r; \rho)
\]

- feature functions \( f(X, W) \), model parameters \( \lambda \)
- distance \( d_{rW} := \lambda^\top (f(X_r, W_r) - f(X_r, W)) \)
- hinge loss function \( l^{(\text{hinge})}(W_r, d_r; \rho) := \max_{W \neq W_r} \{ \max \{ -d_{rW} + \rho(A(W_r, W_r) - A(W, W_r)), 0 \} \} \)
- \( \ell_2 \)-regularization with constant \( C > 0 \)

Altun 2003, Taskar 2003
Margin-based/modified MMI (M-MMI)

\[ F_{\text{M-MMI}, \gamma}(\lambda) = -\frac{C}{2} \| \lambda \|^2 \]

\[ + \sum_{r=1}^{R} \frac{1}{\gamma} \log \left( \frac{\exp(\gamma (\lambda^T f(X_r, W_r) - \rho A(W_r, W_r)))}{\sum_W \exp(\gamma (\lambda^T f(X_r, W) - \rho A(W, W_r)))} \right) \]

**Lemma:** \( F_{\text{M-MMI}, \gamma} \xrightarrow{\gamma \to \infty} \text{SVM}^{\text{hinge}} \) (pointwise convergence).

\( \text{d}^+ \ 2008b \)
Incorporation of Margin Concept

Proof

\[ \Delta A(W, W_r) := A(W_r, W_r) - A(W, W_r) \]

\[-\frac{1}{\gamma} \log \left( \frac{\exp(\gamma(\lambda^T f(X_r, W_r) - \rho A(W_r, W_r)))}{\sum_W \exp(\gamma(\lambda^T f(X_r, W) - \rho A(W, W_r)))} \right) \]

\[ = \frac{1}{\gamma} \log \left( 1 + \sum_{W \neq W_r} \exp(\gamma(-d_{rW} + \rho \Delta A(W, W_r))) \right) \]

\[\gamma \to \infty \left\{ \begin{array}{ll}
\max_{W \neq W_r} \{-d_{rW} + \rho \Delta A(W, W_r)\} & \text{if } \exists W \neq W_r : d_{rW} < \rho \Delta A(W, W_r) \\
0 & \text{otherwise} \end{array} \right.\]

\[= \max_{W \neq W_r} \{ \max \{-d_{rW} + \rho \Delta A(W, W_r), 0\} \} \]

\[=: f^{(hinge)}(W_r, d_r; \rho). \]
Incorporation of Margin Concept
Support Vector Machines (Margin Error)

Optimization problem for SVMs

\[ SVM(\lambda) = -\frac{C}{2} \|\lambda\|^2 - \sum_{r=1}^{R} l(W_r, d_r; \rho) \]

- feature functions \( f(X, W) \), model parameters \( \lambda \)
- distance \( d_{rW} := \lambda^\top (f(X_r, W_r) - f(X_r, W)) \)
- margin error loss function
  \[ l^{(error)}(W_r, d_r; \rho) := E(A(\arg\min_W [d_{rW} + \rho A(W, W_r)], W_r)) \]
- \( \ell_2 \)-regularization with constant \( C > 0 \)

Heigold+2008b
Incorporation of Margin Concept
Smooth Approximation to SVM: Margin-MPE

Margin-based/modified MPE (M-MPE)

\[ F_{\text{M-MPE}, \gamma}(\lambda) = -\frac{C}{2} \|\lambda\|^2 + \sum_{r=1}^{R} \sum_{W} E(W, W_r) \left( \frac{\exp(\gamma(\lambda^T f(X_r, W_r) - \rho A(W_r, W_r)))}{\sum_{V} \exp(\gamma(\lambda^T f(X_r, V) - \rho A(V, W_r)))} \right) \]

Lemma: \( F_{\text{M-MPE}, \gamma} \xrightarrow{\gamma \to \infty} \text{SVM error} \).

d+ 2008b
Incorporation of Margin Concept

Experimental Evaluation of Margin

Digit strings (SieTill, German, telephone)

<table>
<thead>
<tr>
<th>dns/state</th>
<th>feature orders</th>
<th># param,</th>
<th>criterion</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>first</td>
<td>11k</td>
<td>ML</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MMI</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M-MMI</td>
<td>2.7</td>
</tr>
<tr>
<td>64</td>
<td>first</td>
<td>690k</td>
<td>ML</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MMI</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M-MMI</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>first, second, and third</td>
<td>1,409k</td>
<td>Frame</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MMI</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>M-MMI</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Incorporation of Margin Concept

Experimental Evaluation of Margin

European parliament plenary sessions in English (EPPS) and Mandarin broadcasts

<table>
<thead>
<tr>
<th>Criterion</th>
<th>EPPS En 90h</th>
<th>Mandarin BN/BC 230h</th>
<th>Mandarin BN/BC 1500h</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>12.0</td>
<td>21.9</td>
<td>17.9</td>
</tr>
<tr>
<td>MMI</td>
<td>20.8</td>
<td>20.6</td>
<td></td>
</tr>
<tr>
<td>M-MMI</td>
<td>20.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPE</td>
<td>11.5</td>
<td>20.6</td>
<td>16.5</td>
</tr>
<tr>
<td>M-MPE</td>
<td>11.3</td>
<td>20.3</td>
<td>16.3</td>
</tr>
</tbody>
</table>
Incorporation of Margin Concept
Experimental Evaluation of Margin

Handwriting Recognition (IFN/ENIT)

- isolated town names, handwritten
- choose slice features to use 1D HMM
- details: see [Dreuw+ 2009]

<table>
<thead>
<tr>
<th>Criterion</th>
<th>abc-d</th>
<th>abd-c</th>
<th>acd-b</th>
<th>bcd-a</th>
<th>abcd-e</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>7.8</td>
<td>8.7</td>
<td>7.8</td>
<td>8.7</td>
<td>16.8</td>
</tr>
<tr>
<td>MMI</td>
<td>7.4</td>
<td>8.2</td>
<td>7.6</td>
<td>8.4</td>
<td>16.4</td>
</tr>
<tr>
<td>M-MMI</td>
<td>6.1</td>
<td>6.8</td>
<td>6.1</td>
<td>7.0</td>
<td>15.4</td>
</tr>
</tbody>
</table>
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
Conclusions

Effective Discriminative Training

- **Discriminative Criteria**
  - fit decision rule: minimize training error
  - limit overfitting: include regularization and margin to exploit remaining degrees of freedom of the parameters

- **Optimization Methods**
  - general purpose methods give robust estimates
  - in convex case gradient descent still faster than growth transform (GIS)

- **Log-Linear Modeling**
  - convex (w/o hidden variables)
  - covers *Gaussians* completely, w/o constraints on e.g. variance
  - opens modeling up to arbitrary features
  - initialization: from scratch or from *Gaussians*

- **Estimation of Statistics**
  - efficiency: use word lattice to represent competing word sequences
  - implementation: generic approach using WFSTs, covers class of criteria
Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
References


References


References


References


References


References


References


References


Outline

0. Lehrstuhl für Informatik 6

1. Large Vocabulary Speech Recognition

2. Search using Lexical Pronunciation Tree

3. Across-Word Models

4. Word graphs and Applications

5. Time Conditioned Search

6. Normalization and Adaptation

7. Discriminative Training
   7.1 Introduction
   7.2 Overview
   7.3 Training Criteria
   7.4 Parameter Optimization
   7.5 Efficient Calculation of Discriminative Statistics
   7.6 Generalisation to Log-Linear Modeling
   7.7 Convex Optimization
   7.8 Incorporation of Margin Concept
   7.9 Conclusions
   7.10 References
   7.11 Speech Tasks: Corpus Statistics & Setups
## Speech Tasks: Corpus Statistics & Setups

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Train/Test [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SieTill</td>
<td>German digit strings</td>
<td>11/11 (Test)</td>
</tr>
<tr>
<td>EPPS En</td>
<td>English European Parliament plenary speech</td>
<td>92/2.9 (Evl07)</td>
</tr>
<tr>
<td>BNBC Cn 230h</td>
<td>Mandarin broadcasts</td>
<td>230/2.2 (Evl06)</td>
</tr>
<tr>
<td>BNBC Cn 1500h</td>
<td>Mandarin broadcasts</td>
<td>1,500/2.2 (Evl06)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>#States/#Dns</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>SieTill</td>
<td>430/27k</td>
<td>25 LDA(MFCC)</td>
</tr>
<tr>
<td>EPPS En</td>
<td>4,500/830k</td>
<td>45 LDA(MFCC+voicing) +VTLN+SAT/CMLLR</td>
</tr>
<tr>
<td>BNBC Cn 230h</td>
<td>4,500/1,100k</td>
<td>45 LDA(MFCC)+3 tones +VTLN+SAT/CMLLR</td>
</tr>
<tr>
<td>BNBC Cn 1500h</td>
<td>4,500/1,200k</td>
<td>45 SAT/CMLLR(PLP+voicing +3 tones+32 NN)+VTLN</td>
</tr>
</tbody>
</table>
Handwriting Tasks: Corpus Statistics & Setups

IFN/ENIT:
- isolated Tunisian town names
- 4 training folds + 1 additional fold for testing
- simple appearance-based image slice features
- each fold comprises approximately 500,000 frames

<table>
<thead>
<tr>
<th>Corpus</th>
<th>#Observations [k]</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFN/ENIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>6.5</td>
<td>452</td>
</tr>
<tr>
<td>b</td>
<td>6.7</td>
<td>459</td>
</tr>
<tr>
<td>c</td>
<td>6.5</td>
<td>452</td>
</tr>
<tr>
<td>d</td>
<td>6.7</td>
<td>451</td>
</tr>
<tr>
<td>e</td>
<td>6.0</td>
<td>404</td>
</tr>
</tbody>
</table>