Learning a Language Model from Continuous Speech

Graham Neubig, Masato Mimura, Shinsuke Mori, Tatsuya Kawahara

School of Informatics, Kyoto University, Japan
1. Outline
Training of a Speech Recognition System

Text Corpus
this is the song that never ends it just goes on and on my friends and if you started singing it not knowing what it was you'll just keep singing it forever just because this is the song that never ends it just goes on and on my friends and if you started singing it not knowing what it was you'll just keep singing it forever just because this is the song that never ends it just goes on and on my friends and if...

Speech

Transcription

Language Model

Decoder

Acoustic Model

Training

Training
Learning a Language Model from Continuous Speech

Training of a Speech Recognition System

Text Corpus

Training

Language Model

Decoder

Speech

Transcription

this is the song that never ends it just goes on and on my friends and if you started singing it not knowing what it was you'll just keep singing it forever just because this is the song that never ends it just goes on and on my friends and if...
Learning a Language Model from Continuous Speech

Training of a Speech Recognition System

- **Speech**
- **Training**
- **Language Model**
- **Decoder**
- **Acoustic Model**

Audio Example:

```
this is the song that never ends it just goes on and on my friends and if you started singing it not knowing what it was you'll just keep singing it forever just because this is the song that never ends it just goes on and on my friends and if...
```
Why Learn a Language Model from Speech?

- A straightforward way to handle spoken language
  - Fillers, colloquial expressions, and pronunciation variants are included in the model
- A way to learn models for resource-poor languages
  - LMs can be learned even for languages with no digitized text
  - Use with language-independent acoustic models? [Schultz & Waibel 01]
- Semi-supervised Learning
  - Learn a model from newspaper text, update it with spoken expressions or new vocabulary from speech
Our Research

- **Goal:** Learn a LM using **no text**
- **Two problems:**
  - Word boundaries are not clear → use unsupervised word segmentation
  - Acoustic ambiguity → Use a phoneme lattice to absorb acoustic model errors
- **Method:** Apply a Bayesian word segmentation method [Mochihashi+ 09] to phoneme lattices
  - Implementation using weighted finite state transducers (WFST)
- **Result:** An LM learned from continuous speech was able to significantly reduce the ASR phoneme error rate on test data
Previous Research

- **Learning words from speech**
  - Using audio/visual data and techniques such as MMI or MDL, learn grounded words [Roy+ 02, Taguchi+ 09]
  - Find similar audio segments using dynamic time warping and acoustic similarity scores [Park+ 08]

- **Learning language models from speech**
  - Use standard LM learning techniques on 1-best AM results [de Marcken 95, Gorin+ 99]
  - Multigram model from acoustic lattices [Driesen+ 08]
  - No research learning n-gram LMs with acoustic uncertainty
  - Most work handles small vocabulary (infant directed speech, digit recognition)
2. Unsupervised word segmentation
**LM-based Supervised Word Segmentation**

- **Training**: Use corpus $W$ that is annotated with word boundaries to train model $G$

- **Decoding**: for character sequence $x$, treat all word sequences $w$ as possible candidates
  - The probability of a candidate is proportional to its LM probability

$$x = \text{i am}$$

Language Model $G$

- $P(w=\text{iam}; G)$
- $P(w=\text{i am}; G)$
- $P(w=\text{ia m}; G)$
- $P(w=\text{i a m}; G)$
**LM-Based Unsupervised Word Segmentation**

- Estimate an unobserved word sequence $W$ of unsegmented corpus $X$, train language model $G$ over $W$.
- We desire a model that is highly expressive, but simple.
  - Likelihood $P(W|G)$ prefers expressive (complex) models.
  - Add a prior $P(G)$ that prefers simple models.
  - Find a model with high joint probability $P(G,W)=P(G)P(W|G)$.

---

**Simple Model**
- $P(G)$ high
- $P(W|G)$ low
- $P(G)P(W|G)$ low

**Ideal Model**
- $P(G)$ mid
- $P(W|G)$ mid
- $P(G)P(W|G)$ mid

**Complex Model**
- $P(G)$ low
- $P(W|G)$ high
- $P(G)P(W|G)$ low
Hierarchical Pitman-Yor Language Model (HPYLM) [Teh 06]

- An n-gram language model based on non-parametric Bayesian statistics
- Has a number of attractive traits
  - Language model smoothing is realized through prior $P(G)$
  - Parameters can be learned using Gibbs sampling

\[
PY(H_a', d_3', \Theta_3) \sim \begin{array}{ccc} H_{ba} & H_{ca} & \ldots \\ \end{array} \quad \begin{array}{ccc} H_{ab} & H_{db} & \ldots \\ \end{array} \sim PY(H_b', d_3', \Theta_3) \\
\sim PY(H_a, d_3, \Theta_3) \\
\sim PY(H_b, d_3, \Theta_3) \\
\sim PY(H_\varepsilon, d_2, \Theta_2) \\
\sim PY(H_{base}, d_1, \Theta_1)
\]
Unsupervised Word Segmentation using HPYLMs [Mochihashi+ 09]

- The model G is separated into a word-based language model LM and a character-based spelling model SM.
- Words and spellings are connected in a probabilistic framework (unknown words can be modeled).
- It is possible to sample word boundaries using a technique called forward-filtering/backward-sampling.
- Can be used with any (non-cyclic) finite-state automaton.
- Very similar to the forward-backward algorithm for HMMs.

\[
P_{LM}(i|s) \cdot P_{LM}(am|i) \cdot P_{LM}(in|am) \cdot P_{LM}(\text{unk}|in) \cdot P_{LM}(\text{now}|\text{unk}) \cdot P_{LM}(</s>|\text{now})
\]
\[
\cdot P_{SM}(c|s) \cdot P_{SM}(h|c) \cdot P_{SM}(i|h) \cdot P_{SM}(b|i) \cdot P_{SM}(a|b) \cdot P_{SM}(</s>|a)
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm

![Diagram showing forward filtering](image)

Forward filtering
add forward probabilities in order
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm

\[
\begin{align*}
\text{Forward filtering} & \quad \text{add forward probabilities in order} \\
\text{add forward probabilities in order} & \quad \text{add forward probabilities in order}
\end{align*}
\]

\[
f(s_0) = 1
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm

\[
egin{align*}
  f(s_0) &= 1 \\
  f(s_1) &= p(s_1|s_0) \times f(s_0) \\
  f(s_2) &= p(s_2|s_0) \times f(s_0) \\
  f(s_3) &= p(s_3|s_1) \times f(s_1) \\
  f(s_4) &= p(s_4|s_2) \times f(s_2) \\
  f(s_5) &= p(s_5|s_3) \times f(s_3) + p(s_5|s_4) \times f(s_4)
\end{align*}
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm

\[
\begin{align*}
  & f(s_0) = 1 \\
  & f(s_1) = p(s_1 | s_0) * f(s_0) \\
  & f(s_2) = p(s_2 | s_0) * f(s_0)
\end{align*}
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm.

\[
\begin{align*}
    f(s_0) &= 1 \\
    f(s_1) &= p(s_1|s_0) * f(s_0) \\
    f(s_2) &= p(s_2|s_0) * f(s_0) \\
    f(s_3) &= p(s_3|s_1) * f(s_1) + p(s_3|s_2) * f(s_2) \\
    f(s_4) &= p(s_4|s_2) * f(s_2) \\
    f(s_5) &= p(s_5|s_3) * f(s_3) + p(s_5|s_4) * f(s_4)
\end{align*}
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm.

\[
\begin{align*}
    f(s_0) &= 1 \\
    f(s_1) &= p(s_1|s_0)f(s_0) \\
    f(s_2) &= p(s_2|s_0)f(s_0) \\
    f(s_3) &= p(s_3|s_1)f(s_1) + p(s_3|s_2)f(s_2) \\
    f(s_4) &= p(s_4|s_1)f(s_1) + p(s_4|s_2)f(s_2) \\
    f(s_5) &= p(s_5|s_3)f(s_3) + p(s_5|s_4)f(s_4)
\end{align*}
\]
Forward Filtering

- **Forward filtering** is identical to the forward step in the forward-backward algorithm.

\[
\begin{align*}
  f(s_0) &= 1 \\
  f(s_1) &= p(s_1|s_0)f(s_0) \\
  f(s_2) &= p(s_2|s_0)f(s_0) \\
  f(s_3) &= p(s_3|s_1)f(s_1) + p(s_3|s_2)f(s_2) \\
  f(s_4) &= p(s_4|s_1)f(s_1) + p(s_4|s_2)f(s_2) \\
  f(s_5) &= p(s_5|s_3)f(s_3) + p(s_5|s_4)f(s_4)
\end{align*}
\]
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

\[
\begin{align*}
\text{backward sampling} & \quad \text{sample edges from the final state} \\
\text{sample} & \quad \text{path, starting at} \\
\text{final state, using} & \quad \text{edge and} \\
\text{forward probabilities} & \quad \text{probabilities}
\end{align*}
\]

\[
\begin{align*}
e(s_5 \rightarrow x) \\
p(x = s_3) & \propto p(s_5 | s_3) \cdot f(s_3) \\
p(x = s_4) & \propto p(s_5 | s_4) \cdot f(s_4)
\end{align*}
\]
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

\[
\begin{align*}
p(s_3|s_1) \\
p(s_1|s_0) \\
p(s_2|s_0) \\
p(s_3|s_2) & \quad p(s_3|s_1) \\
p(s_4|s_1) & \quad p(s_4|s_2) \\
p(s_5|s_3) & \quad p(s_5|s_4) \\
\end{align*}
\]

**backward sampling**

**sample edges from the final state**

\[
\begin{align*}
e(s_5 \rightarrow x) \\
p(x=s_3) & \propto p(s_5|s_3)f(s_3) \\
p(x=s_4) & \propto p(s_5|s_4)f(s_4)
\end{align*}
\]
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

\[
\begin{align*}
p(s_3|s_1) &= p(x=s_1) \propto p(s_3|s_1) * f(s_1) \\
p(x=s_2) &= p(s_2|s_0) \propto p(s_3|s_2) * f(s_2)
\end{align*}
\]
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

```
p(s_3|s_1)
p(s_5|s_3)
p(s_3|s_2)
p(s_4|s_1)
p(s_5|s_4)
p(s_2|s_0)
p(s_1|s_0)
```

**backward sampling**

**sample edges from the final state**
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

\[ p(s_1|s_0) \]
\[ p(s_2|s_0) \]
\[ p(s_2|s_0) \]
\[ p(s_3|s_1) \]
\[ p(s_3|s_1) \]
\[ p(s_4|s_2) \]
\[ p(s_5|s_3) \]
\[ p(s_5|s_3) \]
\[ p(s_5|s_4) \]

**backward sampling**

**sample edges from the final state**
Backward Sampling

- **Backward sampling** samples a path, starting at the final state, using the edge and forward probabilities.

backward sampling
sample edges from the final state
3. WFST Implementation and Learning from Speech
Generating Word Segmentation Candidates with WFSTs

- We propose a simple way to generate word segmentation candidates using WFSTs
- The WFSTs are quite similar to those used in ASR
A Language Model WFST for Word Segmentation

- Express both the Language Model LM and Spelling Model SM as a single WFST
Learning a Language Model from Continuous Speech

A Language Model WFST for Word Segmentation

- Express both the Language Model LM and Spelling Model SM as a single WFST

The key is the weighted edges connecting the two models.
Word Segmentation Candidates as a WFST

- Vocabulary “i, a, am”, unigram model
Word Segmentation Candidates as a WFST

- Vocabulary “i, a, am”, unigram model

![Diagram of a WFST with transitions and labels for word segmentation candidates]
Word Segmentation Candidates as a WFST

- Vocabulary “i, a, am”, unigram model
Word Segmentation
Candidates as a WFST

- Vocabulary “i, a, am”, unigram model
Adaptation to Speech

- When using WFSTs, adaptation to speech is simple
- Replace input $X$ with a HMM-based acoustic model
  - Forward-filtering $=$ creation of a recognition lattice
- However, full expansion using HMMs is impossible
  - Instead, we use a trimmed phoneme lattice with acoustic model scores
## Learning from Text, Learning from Speech

<table>
<thead>
<tr>
<th>Input</th>
<th>Text</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Character String</td>
<td>Phoneme Lattice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Text</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WFST Composition, Sampling</td>
<td>WFST Composition, Sampling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probability</th>
<th>Text</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P(W</td>
<td>G)P(G) ) (LM Likelihood, Prior)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Samples</th>
<th>Text</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Segmentation, LM</td>
<td>Phoneme String for Each Utterance, Segmentation, LM</td>
</tr>
</tbody>
</table>
4. Evaluation
Experimental Setting

• **Target**: Speech from meetings of the Japanese diet
  - Fluent, large-vocabulary speech
  - Actual vocabulary size is 2858 words

• **Data preparation**: triphone acoustic model
  - PER: one-best 34.2%, oracle 8.1%
  - Used *syllable lattices*, not phoneme lattices (due to requirements of the decoder)
  - 8-117 minutes of training data, 27 minutes of test data

• **Evaluation standard**:
  - Phoneme error rate over the test data using language model learned from training speech
An LM learned from continuous speech reduced the PER by 8.92%.

3-gram is better than 1-gram: learned contextual info.
Other Training Methods

1-best & syllable 3-gram
1-best & unsupervised seg.
lattice & unsupervised seg (proposed method)
manual transcription, segmentation
5. Conclusion
Conclusion

- We demonstrated that it is possible to learn a language model from continuous speech

  Released open source
  http://www.phontron.com/latticelm

- A number of potential applications
  - Learning language models and dictionaries for resource-poor languages
  - Elegant handling of spoken language
  - Semi-supervised learning
Thank You
Extra Slides
Vocabulary/Model Complexity

<table>
<thead>
<tr>
<th></th>
<th>1-gram</th>
<th>2-gram</th>
<th>3-gram</th>
<th>Gold Standard 3-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vocabulary</td>
<td>4480</td>
<td>1351</td>
<td>708</td>
<td>2858</td>
</tr>
<tr>
<td>Average Word Length (Syl.)</td>
<td>2.03</td>
<td>1.37</td>
<td>1.18</td>
<td>1.73</td>
</tr>
<tr>
<td>Language Model States</td>
<td>4480</td>
<td>16150</td>
<td>38759</td>
<td>34073</td>
</tr>
<tr>
<td>Spelling Model States</td>
<td>9624</td>
<td>3869</td>
<td>2426</td>
<td>8378</td>
</tr>
</tbody>
</table>
# Words learned

## Particles

<table>
<thead>
<tr>
<th>word</th>
<th>English</th>
<th># (rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>possessive</td>
<td>1052 (1)</td>
</tr>
<tr>
<td>ni</td>
<td>positional</td>
<td>830 (2)</td>
</tr>
<tr>
<td>to</td>
<td>and</td>
<td>685 (5)</td>
</tr>
</tbody>
</table>

## Subwords

<table>
<thead>
<tr>
<th>word</th>
<th>English</th>
<th># (rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ka</td>
<td>particle, subword</td>
<td>713 (3)</td>
</tr>
<tr>
<td>to:</td>
<td>subword</td>
<td>204 (27)</td>
</tr>
<tr>
<td>sai</td>
<td>subword</td>
<td>94 (65)</td>
</tr>
</tbody>
</table>

## Colloquial Expressions

<table>
<thead>
<tr>
<th>word</th>
<th>English</th>
<th># (rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yu:</td>
<td>say (colloq)</td>
<td>324 (19)</td>
</tr>
<tr>
<td>e:</td>
<td>filler</td>
<td>202 (28)</td>
</tr>
<tr>
<td>desune</td>
<td>discourse marker</td>
<td>94 (65)</td>
</tr>
</tbody>
</table>

## Content/Function Words

<table>
<thead>
<tr>
<th>word</th>
<th>English</th>
<th># (rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>koto</td>
<td>thing</td>
<td>189 (32)</td>
</tr>
<tr>
<td>omo</td>
<td>think (stem)</td>
<td>56 (109)</td>
</tr>
<tr>
<td>hanashi</td>
<td>speak</td>
<td>23 (242)</td>
</tr>
</tbody>
</table>

rimasukeredomo, mo:shiage, yu:fu:ni

jo:kyo:, kangae, chi:ki, toki, shiteki
Experimental Setup (2)

- **Training:**
  - 8-117 minutes of *continuous speech* as training data
  - 0.5-20 second utterances
  - Flat priors on hyperparameters, little influence
  - 20 samples burn-in, 50 LM samples

- **Testing:**
  - 27 minutes of speech separate from the training data
  - Lattice rescoring (not speech recognition)
  - Viterbi phoneme strings for each LM sample *combined* using ROVER
Interesting Pronunciation Variants

- **nippon** (Japan) → **nippo:n**
  - Learned with a long vowel not in the transcription
  - Extra emphasis is put on the name of the country, particularly when using nippon instead of nihon
- **shiteorimasu** (is doing) → **shitorimasu**
  - There are many places where the speakers skip vowels
- **N** → **nothing**
  - Many word-final Ns are not recognized by the AM
- Perhaps taking these into account would **improve AM training**?
Entropy Evaluation

- Gain over 1-best is much lower, why?
  - Different pronunciations than the transcription
    \textit{shiteorimasu} → \textit{shitorimasu}
  - Large effect on entropy, small on PER

- Diagram showing entropy and PER over minutes of training data
  - 1-best & syllable 3-gram
  - 1-best & unsupervised seg.
  - Lattice & unsupervised seg (proposed method)
  - Manual transcription, segmentation
Future Work: Grounding

• The model learns a segmented phoneme string

• For transcription, use actual text
  • Grounding with a grapheme string without pronunciations (subtitles?)
  • In semi-supervised learning, phonetic pronunciations of unknown words is often sufficient

• For dialog, use semantic grounding
  • Use a robot with cameras, match images to words
Future Work: Integration with HMM

- Currently working on lattices, **direct integration with HMM will give better results** (for both training, testing)
Future Work: Implementation

**Speed**
- Expanding FST lattice and forward filtering take a fair amount of time
  - 0.5-1 times real time

**Several ways for improvement**
- Perform *beam-search trimming* during forward filtering
- Parallel sampling

**Open-source**
- Will be made open-source pending code clean-up
- Goal: mid-September
Formal Modeling

- For text word segmentation, \( P(X|W) = 1 \), but for speech this is not the case.
- Our new objective is the joint probability of the model and acoustic features.
  \[
P(X,W,G) = P(X|W)P(W|G)P(G)
  \]
- Use an acoustic model scaling factor.
  \[
P(X,W,G) = \lambda P(X|W)P(W|G)P(G)
  \]
- Set to .2 (values between .1-.2 produced similar results).
Weighted Finite State Transducers (WFSTs)

- Finite state automata with input/output/weight

  ![Diagram A]

  ![Diagram B]

- Define weighted relations over strings
  - If weights are probabilities, probabilistic relations
  - Transducers combined through composition

  ![Diagram A⊙B]
Connecting Edges in Detail

- To the SM from the base state
  - Equal to the probability of generating a symbol from the base distribution

- In HPYLM, n-grams with an unknown word as \( w_{i-1} \) are equal to base probabilities*
  \[ P(w_i|w_{i-2}, \text{UNK}) = P(w_i|\text{UNK}) = P(w_i) \]

- OK to make edges from the SM only to base state

* technically not true if the same word appears twice in a single sentence
## Difference from Mochihashi's Method

<table>
<thead>
<tr>
<th></th>
<th>Mochihashi</th>
<th>Neubig</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spelling Model</strong></td>
<td>$\infty$-gram+ Poisson Distribution Explicit Length Limit</td>
<td>Character 3-gram No Length Limit</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Algorithmic Faster?</td>
<td>WFST-Based Simpler?, Lattice Possible</td>
</tr>
<tr>
<td><strong>Worst-Case Complexity</strong></td>
<td>$O(ML^n)$ ( M=\text{Sentence length} ) ( L=\text{Max word length} ) ( n=\text{n-gram length} )</td>
<td>$O(M^{n+1})$</td>
</tr>
<tr>
<td><strong>Expected Complexity</strong></td>
<td>$O(ML^n)$</td>
<td>$O(kM+E)$ ( E=\text{Number of existing word n-grams} ) ( k=\text{Spelling model} )</td>
</tr>
</tbody>
</table>