Automatic Speech Recognition (CS753)
Lecture 9: RNN-based architectures for ASR

Instructor: Preethi Jyothi
Aug 21, 2017
Recap: Hybrid DNN-HMM Systems

- Instead of GMMs, use scaled DNN posteriors as the HMM observation probabilities
- DNN trained using triphone labels derived from a forced alignment “Viterbi” step.
  - **Forced alignment**: Given a training utterance \( \{O,W\} \), find the most likely sequence of states (and hence triphone state labels) using a set of trained triphone HMM models, \( M \). Here \( M \) is constrained by the triphones in \( W \).
Recap: Tandem DNN-HMM Systems

- Neural network outputs are used as “features” to train HMM-GMM models
- Use a low-dimensional bottleneck layer representation to extract features from the bottleneck layer
Feedforward DNNs we’ve seen so far...

- Assume independence among the training instances

  - Independent decision made about classifying each individual speech frame

  - Network state is completely reset after each speech frame is processed

- This independence assumption fails for data like speech which has temporal and sequential structure
Recurrent Neural Networks

• Recurrent Neural Networks (RNNs) work naturally with sequential data and process it one element at a time

• HMMs also similarly attempt to model time dependencies. How’s it different?

  • HMMs are limited by the size of the state space. Inference becomes intractable if the state space grows very large!

• What about RNNs?
Two main equations govern RNNs:

\[ h_t = H(Wx_t + Vh_{t-1} + b^{(h)}) \]

\[ y_t = O(Uh_t + b^{(y)}) \]

where \( W, V, U \) are matrices of input-hidden weights, hidden-hidden weights and hidden-output weights resp; \( b^{(h)} \) and \( b^{(y)} \) are bias vectors
Recurrent Neural Networks

• Recurrent Neural Networks (RNNs) work naturally with sequential data and process it one element at a time

• HMMs also similarly attempt to model time dependencies. How’s it different?

• HMMs are limited by the size of the state space. Inference becomes intractable if the state space grows very large!

• What about RNNs? RNNs are designed to capture long-range dependencies unlike HMMs: Network state is exponential in the number of nodes in a hidden layer
Training RNNs

- An unrolled RNN is just a very deep feedforward network
- For a given input sequence:
  - create the unrolled network
  - add a loss function node to the network
  - then, use backpropagation to compute the gradients
Backpropagation

\[ \frac{\partial L}{\partial u} = \sum_{v \in \Gamma(u)} \frac{\partial L}{\partial v} \cdot \frac{\partial v}{\partial u} \]

**Backpropagation**

**Base case:** \( \frac{\partial L}{\partial L} = 1 \)

**For each** \( u \) (top to bottom):

- **For each** \( v \in \Gamma(u) \):
  - Inductively, have computed \( \frac{\partial L}{\partial v} \)
  - Directly compute \( \frac{\partial v}{\partial u} \)

**Compute** \( \frac{\partial L}{\partial u} \)

**Compute** \( \frac{\partial L}{\partial w} \)

where \( \frac{\partial L}{\partial w} = \frac{\partial L}{\partial u} \cdot \frac{\partial u}{\partial w} \)

**Forward Pass**

First, in a forward pass, compute values of all nodes given an input (The values of each node will be needed during backprop)

Where values computed in the forward pass may be needed
Training RNNs

- An unrolled RNN is just a very deep feedforward network

- For a given input sequence:
  - create the unrolled network
  - add a loss function node to the network
  - then, use backpropagation to compute the gradients

- This algorithm is known as backpropagation through time (BPTT)
Deep RNNs

- RNNs can be stacked in layers to form deep RNNs
- Empirically shown to perform better than shallow RNNs on ASR [G13]

---

Vanilla RNN Model

\[ h_t = H(Wx_t + Vh_{t-1} + b^{(h)}) \]

\[ y_t = O(Uh_t + b^{(y)}) \]

- **H**: Element wise application of the sigmoid or tanh function
- **O**: The softmax function

Run into problems of exploding and vanishing gradients.
Exploding/Vanishing Gradients

• In deep networks, gradients in early layers is computed as the product of terms from all the later layers

• This leads to unstable gradients:
  
  • If the terms in later layers are large enough, gradients in early layers (which is the product of these terms) can grow exponentially large: Exploding gradients
  
  • If the terms are in later layers are small, gradients in early layers will tend to exponentially decrease: Vanishing gradients

• To address this problem in RNNs, Long Short Term Memory (LSTM) units were proposed [HS97]

Long Short Term Memory Cells

- Memory cell: Neuron that stores information over long time periods
- Forget gate: When on, memory cell retains previous contents. Otherwise, memory cell forgets contents.
- When input gate is on, write into memory cell
- When output gate is on, read from the memory cell
Bidirectional RNNs

- BiRNNs process the data in both directions with two separate hidden layers.
- Outputs from both hidden layers are concatenated at each position.
RNN-based ASR system
ASR with RNNs

• We have seen how neural networks can be used for acoustic models in ASR systems

  • Main limitation: Frame-level training targets derived from HMM-based alignments

• Goal: Single RNN model that addresses these issues and replaces as much of the speech pipeline as possible [G14]

---

RNN Architecture

- $H$ was implemented using LSTMs in [G14]. Input: Acoustic feature vectors, one per frame; Output: Characters + space
- Deep bidirectional LSTM networks were used

---

Connectionist Temporal Classification (CTC)

• Neural networks in ASR are trained at the frame-level and typically require alignments between the acoustics and the word sequence during training telling you which label (e.g. triphone state) should be output at each timestep

• CTC tries to get around this!

• This is an objective function that allows RNN training without this explicit alignment step: CTC considers all possible alignments
CTC: Pre-requisites

• Augment the output vocabulary with an additional “blank” (\_\_) label

• For a given label sequence, there can be multiple alignments: (x, y, z) could correspond to (x, \_, y, \_, \_, z) or (\_, x, x, \_, y, z)

• Define a 2-step operator $B$ that reduces a label sequence by first, removing repeating labels and second, removing blanks. $B(“x, \_, y, \_, \_, z”) = B(“\_, x, x, \_, y, z”) = “x, y, z”$
CTC Objective Function

- CTC objective function is the probability of an output label sequence $y$ given an utterance $x$

$$CTC(x, y) = \Pr(y|x) = \sum_{a \in B^{-1}(y)} \Pr(a|x)$$

- Here, we sum over all possible alignments for $y$, enumerated by $B^{-1}(y)$

- CTC assumes that $\Pr(a|x)$ can be computed as $\prod_{t=1}^{T} \Pr(a_t|x)$

- i.e. CTC assumes that outputs at each time-step are conditionally independent given the input

- Efficient dynamic programming algorithm to compute this loss function and its gradients [GJ14]

[GJ14] Towards End-to-End Speech Recognition with Recurrent Neural Networks, ICML 14
Decoding

- First approximation: For a given test input sequence $x$, pick the most probable output at each time step

$$\arg \max_y \Pr(y|x) \approx B(\arg \max_a \Pr(a|x))$$

- More accurate decoding uses a search algorithm that also makes use of a dictionary and a language model. (Decoding search algorithms will be discussed in detail in later lectures.)
## WER results

<table>
<thead>
<tr>
<th>System</th>
<th>LM</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN-CTC</td>
<td>Dictionary only</td>
<td>24.0</td>
</tr>
<tr>
<td>RNN-CTC</td>
<td>Bigram</td>
<td>10.4</td>
</tr>
<tr>
<td>RNN-CTC</td>
<td>Trigram</td>
<td>8.7</td>
</tr>
<tr>
<td>Baseline</td>
<td>Bigram</td>
<td>9.4</td>
</tr>
<tr>
<td>Baseline</td>
<td>Trigram</td>
<td>7.8</td>
</tr>
</tbody>
</table>

Some erroneous examples produced by the end-to-end RNN

Target: “There’s unrest but we’re not going to lose them to Dukakis”
Output: “There’s unrest but we’re not going to lose them to Dekakis”

Target: “T. W. A. also plans to hang its boutique shingle in airports at Lambert Saint”
Output: “T. W. A. also plans tohing its bootik single in airports at Lambert Saint”
Another end-to-end system

- Decoding is still at the word level. Out-of-vocabulary (OOV) words cannot be handled.

- Build a system that is trained and decoded entirely at the character-level.

  - This would enable the transcription of OOV words, disfluencies, etc.

- [M et al.]: Shows results on the Switchboard task. Matches a GMM-HMM baseline system but underperforms compared to an HMM-DNN baseline.

[M et al.]: Maas et al., “Lexicon Free Conversational Speech Recognition with Neural Networks”, NAACL 15
Model Specifics

- Approach consists of two neural models:
  - A deep bidirectional RNN (DBRNN) mapping acoustic features to character sequences (Trained using CTC.)
  - A neural network character language model

![Diagram of DBRNN](image)

Image from Maas et al., “Lexicon Free Conversational Speech Recognition with Neural Networks”, NAACL 15
Decoding

• Simplest form: Decode without any language model

• Beam Search decoding:
  • Combine DBRNN outputs with a char-level language model
  • Char-level language model applied at every time step (unlike word models)
  • Circumvents the issue of handling OOV words during decoding
Experimental Results

<table>
<thead>
<tr>
<th>Method</th>
<th>CER</th>
<th>EV</th>
<th>CH</th>
<th>SWBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM-GMM</td>
<td>23.0</td>
<td>29.0</td>
<td>36.1</td>
<td>21.7</td>
</tr>
<tr>
<td>HMM-DNN</td>
<td>17.6</td>
<td>21.2</td>
<td>27.1</td>
<td>15.1</td>
</tr>
<tr>
<td>HMM-SHF</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
<td>12.4</td>
</tr>
<tr>
<td>CTC no LM</td>
<td>27.7</td>
<td>47.1</td>
<td>56.1</td>
<td>38.0</td>
</tr>
<tr>
<td>CTC+5-gram</td>
<td>25.7</td>
<td>39.0</td>
<td>47.0</td>
<td>30.8</td>
</tr>
<tr>
<td>CTC+7-gram</td>
<td>24.7</td>
<td>35.9</td>
<td>43.8</td>
<td>27.8</td>
</tr>
<tr>
<td>CTC+NN-1</td>
<td>24.5</td>
<td>32.3</td>
<td>41.1</td>
<td>23.4</td>
</tr>
<tr>
<td>CTC+NN-3</td>
<td>24.0</td>
<td>30.9</td>
<td>39.9</td>
<td>21.8</td>
</tr>
<tr>
<td>CTC+RNN</td>
<td>24.9</td>
<td>33.0</td>
<td>41.7</td>
<td>24.2</td>
</tr>
<tr>
<td>CTC+RNN-3</td>
<td>24.7</td>
<td>30.8</td>
<td>40.2</td>
<td>21.4</td>
</tr>
</tbody>
</table>

Table 1: Character error rate (CER) and word error rate results on the Eval2000 test set. We report word error rates on the full test set (EV) which consists of the Switchboard (SWBD) and CallHome (CH) subsets. As baseline systems we use an HMM-GMM system and HMM-DNN system. We evaluate our DBRNN trained using CTC by decoding with several character-level language models: 5-gram, 7-gram, densely connected neural networks with 1 and 3 hidden layers (NN-1, and NN-3), as well as recurrent neural networks s with 1 and 3 hidden layers. We additionally include results from a state-of-the-art HMM-based system (HMM-DNN-SHF) which does not report performance on all metrics we evaluate (NR).
Sample Test Utterances

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>Transcription</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Truth</td>
<td>yeah i went into the i do not know what you think of <em>fidelity</em> but</td>
</tr>
<tr>
<td></td>
<td>HMM-GMM</td>
<td>yeah when the i don’t know what you think of fidel it even them</td>
</tr>
<tr>
<td></td>
<td>CTC+CLM</td>
<td>yeah i went to i don’t know what you think of fidelity but um</td>
</tr>
<tr>
<td>(2)</td>
<td>Truth</td>
<td>no no speaking of weather do you carry a altimeter slash <em>barometer</em></td>
</tr>
<tr>
<td></td>
<td>HMM-GMM</td>
<td>no i’m not all being the weather do you uh carry a uh helped emitters last</td>
</tr>
<tr>
<td></td>
<td></td>
<td>brahms her</td>
</tr>
<tr>
<td></td>
<td>CTC+CLM</td>
<td>no no beating of whether do you uh carry a uh a time or less barometer</td>
</tr>
<tr>
<td>(3)</td>
<td>Truth</td>
<td>i would ima- well yeah it is i know you are able to stay home with them</td>
</tr>
<tr>
<td></td>
<td>HMM-GMM</td>
<td>i would amount well yeah it is i know um you’re able to stay home with them</td>
</tr>
<tr>
<td></td>
<td>CTC+CLM</td>
<td>i would ima- well yeah it is i know uh you’re able to stay home with them</td>
</tr>
</tbody>
</table>

Table 2: Example test set utterances with a ground truth transcription and hypotheses from our method (CTC+CLM) and a baseline HMM-GMM system of comparable overall WER. The words *fidelity* and *barometer* are not in the lexicon of the HMM-GMM system.
Analysis

Figure 3: Character probabilities from the CTC-trained neural network averaged over monophone segments created by a forced alignment of the HMM-GMM system. Time is measured in frames, with 0 indicating the start of the monophone segment. The vertical dotted line indicates the average duration of the monophone segment. We show only characters with non-trivial probability for each phone while excluding the blank and space symbols.

Probabilities generally rise slightly later in the phone segment as compared to consonants. This may occur to avoid the large contextual variations in vowel pronunciations at phone boundaries. For certain consonants we observe CTC probability spikes before the monophone segment begins, as is the case for sh.

The probabilities for sh additionally exhibit multiple modes, suggesting that CTC may learn different behaviors for the two common spellings of the sibilant sh: the letter sequence “sh” and the letter sequence “ti.”

6 Conclusion

We presented an LVCSR system consisting of two neural networks integrated via beam search decoding that matches the performance of an HMM-GMM system on the challenging Switchboard corpus. We built on the foundation of Graves and Jaitly (2014) to vastly reduce the overall complexity required for LVCSR systems. Our method yields a complete first-pass LVCSR system with about 1,000 lines of code — roughly an order of magnitude less than high performance HMM-GMM systems. Operating entirely at the character level yields a system which does not require assumptions about a lexicon or pronunciation dictionary, instead learning orthography and phonics directly from data. We hope the simplicity of our approach will facilitate future research in improving LVCSR with CTC-based systems and jointly training LVCSR systems for SLU tasks. DNNs have already shown great results as acoustic models in HMM-DNN systems. We free the neural network from its complex HMM infrastructure, which we view as the first step towards the next wave of advances in speech recognition and language understanding.

Acknowledgments

We thank Awni Hannun for his contributions to the software used for experiments in this work. We also thank Peng Qi and Thang Luong for insightful discussions, and Kenneth Heafield for help with the KenLM toolkit. Our work with HMM-GMM systems was possible thanks to the Kaldi toolkit and its contributors. Some of the GPUs used in this work were donated by the NVIDIA Corporation. AM was supported as an NSF IGERT Traineeship Recipient under Award 0801700. ZX was supported by an NDSEG Graduate Fellowship.

Image from Maas et al., “Lexicon Free Conversational Speech Recognition with Neural Networks”, NAACL 15