Sequence to Sequence Learning

Yonghui
Agenda

- Introduction
- Language modeling
- Machine translation (NMT)
- Speech recognition (ASR)
- Speech synthesis (TTS)
- Other sequence model applications
  - Image captioning
  - Syntactic parsing
  - Etc
Introduction

- **Sequence to sequence learning:**
  - Try to learn a mapping from one sequence to another sequence

- **Examples include**
  - Machine translation (MT)
  - Automatic speech recognition (ASR)
  - Speech synthesis (TTS)
  - Handwriting generation

- **Seq2se learning using Encoder/decoder with attention model architecture has achieved state of the art results on many of the problems**
Language modeling
## Language Modeling

| context       | target | $P(w_t|w_{t-1}, w_{t-2}, \ldots w_{t-5})$ |
|---------------|--------|----------------------------------------|
| the cat sat on the | mat    | 0.15                                  |
| $w_{t-5}$ $w_{t-4}$ $w_{t-3}$ $w_{t-2}$ $w_{t-1}$ | $w_t$   |                                        |
| the cat sat on the | rug    | 0.12                                  |
| the cat sat on the | hat    | 0.09                                  |
| the cat sat on the | dog    | 0.01                                  |
| the cat sat on the | the    | 0                                     |
| the cat sat on the | sat    | 0                                     |
| the cat sat on the | robot  | ?                                     |
| the cat sat on the | printer| ?                                     |
n-grams

The slide illustrates a model for predicting the probability of a word given a context. The formula for the probability is given as:

$$m_{w,c} \propto \frac{\#(w,c)}{\#(c)}$$

Examples:
- The cat sat on the mat
- The cat drinks milk
- The dog chases the cat
- The paws of the cat
- The cat chases the rat
- The rat eats cheese
- The rat eats the mat
The Chain Rule

\[ P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1) \]

<table>
<thead>
<tr>
<th>the</th>
<th>cat</th>
<th>sat</th>
<th>on</th>
<th>the</th>
<th>mat</th>
<th>( P(w_1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>cat</td>
<td>sat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>( P(w_2</td>
</tr>
<tr>
<td>the</td>
<td>cat</td>
<td>sat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>( P(w_3</td>
</tr>
<tr>
<td>the</td>
<td>cat</td>
<td>sat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>( P(w_4</td>
</tr>
<tr>
<td>the</td>
<td>cat</td>
<td>sat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>( P(w_5</td>
</tr>
<tr>
<td>the</td>
<td>cat</td>
<td>sat</td>
<td>on</td>
<td>the</td>
<td>mat</td>
<td>( P(w_6</td>
</tr>
</tbody>
</table>
Markov assumption

\[ P(w_1, w_2, \ldots, w_{T-1}, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n+1}) \]
N-grams model improvements

- Very intuitive, easy to understand
- Sparsity
  - “w, c”, or “c” may not appear in a corpus at all, but the probability \( P(w|c) \) shouldn’t be 0
- Limited context
  - The length of the context is very limited
- Back-off models
  - Back off from trigram model to a bigram model, or from a bigram model to a unigram model.
  - Kneser–Ney smoothing
  - Katz’s smoothing
Neural language model

- Two key ingredients: neural embeddings and recurrent neural networks
Neural embedding

\[ p(w_t | w_1, \ldots, w_{t-1}) = p_\theta(w_t | f_\theta(w_1, \ldots, w_{t-1})) \]

\[ P(w | c) = \frac{e^{s_\theta(w,c)}}{\sum_{v=1}^{V} e^{s_\theta(v,c)}} \]
Recurrent Neural Network Language Models


“persistent memory”: state variable for arbitrarily long contexts

\[ z_t = \tanh(Wz_{t-1} + Uw_t) \]
\[ p(w_{t+1}) = \text{softmax}(Bz_t) \]

the cat
Recurrent Neural Network Language Models


“persistent memory”: state variable for arbitrarily long contexts

\[ z_t = \tanh(W z_{t-1} + U w_t) \]

\[ p(w_{t+1}) = \text{softmax}(B z_t) \]
Recurrent Neural Network Language Models


“persistent memory”: state variable for arbitrarily long contexts

\[ z_t = \tanh(Wz_{t-1} + Uw_t) \]

\[ p(w_{t+1}) = \text{softmax}(Bz_t) \]
Recurrent Neural Network Language Models
Recurrent Neural Network Language Models

Slide Credit: Piotr Mirowski
Recurrent Neural Network Language Models
Recurrent Neural Network Language Models
What do we Optimize?

$$\theta^* = \arg \max_\theta E_{w \sim \text{data}} \log P_\theta(w_1, \ldots, w_T)$$
RNN language model

- Recurrent neural network based language model by Tomas Mikolov et al at Interspeech 2010

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>KN 5gram</td>
<td>93.7</td>
</tr>
<tr>
<td>feedforward NN</td>
<td>85.1</td>
</tr>
<tr>
<td>recurrent NN</td>
<td>80.0</td>
</tr>
<tr>
<td>4xRNN + KN5</td>
<td>73.5</td>
</tr>
</tbody>
</table>

Simple experiment: 4M words from Switchboard corpus

Feedforward networks used here are slightly different than what Bengio & Schwenk use

Exact architecture, e.g. num layers, num nodes and etc, used in the experiments is not clear.
Long short-term memory network

\[ i_t = \sigma (W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i) \]
\[ f_t = \sigma (W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f) \]
\[ c_t = f_t c_{t-1} + i_t \tanh (W_{xc} x_t + W_{hc} h_{t-1} + b_c) \]
\[ o_t = \sigma (W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_t + b_o) \]
\[ h_t = o_t \tanh (c_t) \]
More powerful RNN

- To further improve the power of RNN networks, one can stack a few layers of RNN/LSTM.
Residual connection

- For deep RNN network, usually you need residual connections to stabilize/speed up training
A much better RNN language model

- **Exploring the Limits of Language Modeling**, by Rafal Jozefowicz

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
<th>Number of Params [billions]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGMOID-RNN-2048 (<em>Ji et al.</em>, 2015a)</td>
<td>68.3</td>
<td>4.1</td>
</tr>
<tr>
<td>Interpolated KN 5-gram, 1.1B n-grams (<em>Chelba et al.</em>, 2013)</td>
<td>67.6</td>
<td>1.76</td>
</tr>
<tr>
<td>Sparse Non-Negative Matrix LM (<em>Shazeer et al.</em>, 2015)</td>
<td>52.9</td>
<td>33</td>
</tr>
<tr>
<td>RNN-1024 + MaxEnt 9-gram features (<em>Chelba et al.</em>, 2013)</td>
<td>51.3</td>
<td>20</td>
</tr>
<tr>
<td>LSTM-512-512</td>
<td>54.1</td>
<td>0.82</td>
</tr>
<tr>
<td>LSTM-1024-512</td>
<td>48.2</td>
<td>0.82</td>
</tr>
<tr>
<td>LSTM-2048-512</td>
<td>43.7</td>
<td>0.83</td>
</tr>
<tr>
<td>LSTM-8192-2048 (No Dropout)</td>
<td>37.9</td>
<td>3.3</td>
</tr>
<tr>
<td>LSTM-8192-2048 (50% Dropout)</td>
<td>32.2</td>
<td>3.3</td>
</tr>
<tr>
<td>2-Layer LSTM-8192-1024 (BIG LSTM)</td>
<td>30.6</td>
<td>1.8</td>
</tr>
</tbody>
</table>
The state of the art LM

- **Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer**, by Noam Shazeer and et al

- Stacked LSTM layers + Mixture of Expert layer in between
MOE

Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

\[ y = \sum_{i=1}^{n} G(x)_i E_i(x) \]
### MOE LM results

- Results on LM1B dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity 10 epochs</th>
<th>Test Perplexity 100 epochs</th>
<th>#Parameters excluding embedding and softmax layers</th>
<th>ops/timestep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Published Results</td>
<td>34.7</td>
<td>30.6</td>
<td>151 million</td>
<td>151 million</td>
</tr>
<tr>
<td>Low-Budget MoE Model</td>
<td>34.1</td>
<td></td>
<td>4303 million</td>
<td>8.9 million</td>
</tr>
<tr>
<td>Medium-Budget MoE Model</td>
<td>31.3</td>
<td></td>
<td>4313 million</td>
<td>33.8 million</td>
</tr>
<tr>
<td>High-Budget MoE Model</td>
<td><strong>28.0</strong></td>
<td></td>
<td>4371 million</td>
<td>142.7 million</td>
</tr>
</tbody>
</table>
Neural Machine Translation
seq2seq

- Sequence to Sequence Learning with Neural Networks, by Ilya et al
  - Concatenate the source and target sequence
  - Only make prediction on the target sequence

\[
P(y_1, \ldots, y_{T'}, x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t \mid v, y_1, \ldots, y_{t-1})
\]
Decoding in a Nutshell (Beam Size 2)

\[ y^* = \arg\max_{y_1, \ldots, y_{T'}} P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) \]
Important Tricks

● When the model was first proposed, no one would have believed that it can solve the translation problem
● But it worked quite well
● Tricks that are important for the model quality
  ○ Reverse the source sequence
  ○ Deep lstms (4 layers lstm networks)
Seq2Seq experimental results

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahdanau et al. [2]</td>
<td>28.45</td>
</tr>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Single forward LSTM, beam size 12</td>
<td>26.17</td>
</tr>
<tr>
<td>Single reversed LSTM, beam size 12</td>
<td>30.59</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 1</td>
<td>33.00</td>
</tr>
<tr>
<td>Ensemble of 2 reversed LSTMs, beam size 12</td>
<td>33.27</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 2</td>
<td>34.50</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 12</td>
<td><strong>34.81</strong></td>
</tr>
</tbody>
</table>

Table 1: The performance of the LSTM on WMT’14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.
Limitation of the model

- **Information bottleneck**
  - Regardless of the sequence length, source sequence is encoded using fixed size vectors
  - Solution: Attention

- **Out of vocabulary (OOV) problems**
  - Fixed size vocab, constrained by computational budget and GPU RAM.
  - All words not in vocab will be mapped to the same <UNK> token.
  - More of a problem for morphologically rich languages like Russian and Polish.
  - Solution: WordPiece or BytePair encoding
Attention

- *Generating Sequences With Recurrent Neural Networks*, by Alex Graves
  - Motonic, only moves left to right
  - Proposed to solve the handwriting synthesis issue
  - It is referred to Gaussian Mixture Model (GMM) Attention in the literature

- Generalized by Dzmitry Bahdanau in his paper “Neural Machine Translation by Jointly Learning to Align and Translate”
  - No more monotonicity constraints
Online handwriting synthesis

Figure 14: Mixture density outputs for handwriting synthesis. The top heatmap shows the predictive distributions for the pen locations, the bottom heatmap shows the mixture component weights. Comparison with Fig. 10 indicates that the synthesis network makes more precise predictions (with smaller density blobs) than the prediction-only network, especially at the ends of strokes, where the synthesis network has the advantage of knowing which letter comes next.
GMM Attention

- K Gaussian components
- Each defines a density distribution over the character sequence

Given a length $U$ character sequence $c$ and a length $T$ data sequence $x$, the soft window $w_t$ into $c$ at timestep $t$ ($1 \leq t \leq T$) is defined by the following discrete convolution with a mixture of $K$ Gaussian functions

$$
\phi(t, u) = \sum_{k=1}^{K} \alpha_t^k \exp \left( -\beta_t^k (\kappa_t^k - u)^2 \right)
$$

(46)

$$
w_t = \sum_{u=1}^{U} \phi(t, u)c_u
$$

(47)
Gaussian Mixture Model

\[ \theta_{\mu,\sigma^2}(x) \]

- \( \mu = 0, \ \sigma^2 = 0.2, \)
- \( \mu = 0, \ \sigma^2 = 1.0, \)
- \( \mu = 0, \ \sigma^2 = 5.0, \)
- \( \mu = -2, \ \sigma^2 = 0.5, \)
GMM attention

- GMM params are updated at every timestep.
- Attention params are estimated using decoder rnn hidden states.
- Modeled as shift from the previous center. Hence monotonicity guarantee.

\[
(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_h h^1_p h^1_t + b_p
\]

\[
\alpha_t = \exp(\hat{\alpha}_t)
\]

\[
\beta_t = \exp(\hat{\beta}_t)
\]

\[
\kappa_t = \kappa_{t-1} + \exp(\hat{\kappa}_t)
\]
Figure 13: **Window weights during a handwriting synthesis sequence**

Each point on the map shows the value of $\phi(t, u)$, where $t$ indexes the pen trace along the horizontal axis and $u$ indexes the text character along the vertical axis. The bright line is the alignment chosen by the network between the characters and the writing. Notice that the line spreads out at the boundaries between characters; this means the network receives information about next and previous letters as it makes transitions, which helps guide its predictions.
A generalization of GMM attention


\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}
\]

Figure 1: The graphical illustration of the proposed model trying to generate the \(t\)-th target word \(y_t\) given a source sentence \((x_1, x_2, \ldots, x_T)\).
Attention benefits

- At any point of decoding, the model only focuses on the most relevant part of the source, much like how humans do.
- Encoding of the source sequence is now distributed over all the source words.
  - Longer sequence is encoded using more bits.
Attention visualization

Attention probability matrix can be nicely visualized
Experimental result

Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.
Convolution Attention

- *Attention-Based Models for Speech Recognition*, by Jan Chorowski et al
- Key idea: explicitly incorporate a location information
- Encourages attention to gradually move forward

We extend this content-based attention mechanism of the original model to be location-aware by making it take into account the alignment produced at the previous step. First, we extract $k$ vectors $f_{i,j} \in \mathbb{R}^k$ for every position $j$ of the previous alignment $\alpha_{i-1}$ by convolving it with a matrix $F \in \mathbb{R}^{k \times r}$:

$$f_i = F \ast \alpha_{i-1}. \quad (8)$$

These additional vectors $f_{i,j}$ are then used by the scoring mechanism $e_{i,j}$:

$$e_{i,j} = w^\top \tanh(Ws_{i-1} + Vh_j + Uf_{i,j} + b) \quad (9)$$
Multi-headed attention

- Simple extension to single headed attention
- Multiple attention runs in parallel. Each individual attention may focus on a different region in the input
Motonic Attention

- Online and Linear-Time Attention by Enforcing Monotonic Alignments, by Colin
- Linear time complexity
- Guarantees monotonicity

Figure 2. Schematic of our novel monotonic stochastic decoding process. At each output timestep, the decoder inspects memory entries (indicated in gray) from left-to-right starting from where it left off at the previous output timestep and chooses a single one (indicated in black). A black node indicates that memory element $h_j$ is aligned to output $y_i$. White nodes indicate that a particular input-output alignment was not considered because it violates monotonicity. Arrows indicate the order of processing and dependence between memory entries and output timesteps.
Solve the OOV problem

- Move to sub-word unit
- Character model
  - Sequence length too long
- Wordpiece model strikes a good balance between character model and word model
  - Common words are still just just one wordpiece
  - Rare words are often decomposed into a morphologically meaningful way, word-root and suffix
WordPiece model / Bytepair Encoding

- Japanese and Korean voice search, by Mike Schuster
- Same as bytepair encoding, which Rico Sennrich first adopted for NMT
  - Neural Machine Translation of Rare Words with Subword Units
- Minimal description length
  - Find an encoding of words subject to vocab size limit such that a corpus can be encoded using minimal number of tokens
Simple greedy algorithm

- Very simple algorithm
  - Start from all characters
  - Iteratively merge most frequent bi-grams
  - Stop when vocab size is reached
- Most common word is a single wordpiece

- **Word:** Jet makers feud over seat width with big orders at stake
- **wordpieces:** _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake
One more important trick

- Rare words are often copied from source to target in verbatim
- To facilitate such copying, source and target should share the same WordPiece model
  - Rare words will be segmented exactly the same way
Google Neural Machine Translation

- All the techniques combined
  - Very deep lstm stack
  - Attention
  - Wordpiece model

- Large scale NMT
  - Carefully engineered
  - Pieces are carefully tuned and assembled together

- Some novel inventions
  - Better decoding algorithm
  - Quantized training and inference
  - ...
Multilingual Model

- Model several language pairs in single model
  - We ran first experiments in 2/2016, surprisingly this worked!
- Prepend source with additional token to indicate target language
  - Translate to Spanish:
    - `<2es>` How are you `</s>` -> Cómo estás `</s>`
  - Translate to English:
    - `<2en>` Cómo estás `</s>` -> How are you `</s>`
- No other changes to model architecture!
  - Extremely simple and effective
  - Usually with shared WPM for source/target
- Benefits
  - simplifies model deployment
  - improves translation quality
Multilingual Model

<table>
<thead>
<tr>
<th>Single</th>
<th>Multi</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.5</td>
<td>35.1</td>
</tr>
<tr>
<td>38.0</td>
<td>37.3</td>
</tr>
</tbody>
</table>

Translation:

<2es> How are you </s> Cómo estás </s>
<2en> Cómo estás </s> How are you </s>
Zero-Shot Translation

<table>
<thead>
<tr>
<th></th>
<th>single</th>
<th>multi</th>
</tr>
</thead>
<tbody>
<tr>
<td>En</td>
<td>34.5</td>
<td>35.0</td>
</tr>
<tr>
<td>Pt</td>
<td>44.5</td>
<td>43.7</td>
</tr>
</tbody>
</table>

Zero-shot (pt->es):

<2es> Como você está </s> Cómo estás </s>

23.0 BLEU
Some more recent NMT models

- MOE model, from Google Brain
- Conv seq2seq model, from Facebook
- Transformer Model, from Google Brain
Mixture of experts NMT model

- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, by Noam Shazeer et al
- Based on the GNMT model architecture
  - Add a Mixture of Experts layer between the first and second LSTM layers
  - Adds a lot of params, but actually reduces number of compute
<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
<th>Test BLEU</th>
<th>ops/timenstep</th>
<th>Total #Parameters</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoE with 2048 Experts</td>
<td>2.69</td>
<td>40.35</td>
<td>85M</td>
<td>8.7B</td>
<td>3 days/64 k40s</td>
</tr>
<tr>
<td>MoE with 2048 Experts (longer training)</td>
<td><strong>2.63</strong></td>
<td><strong>40.56</strong></td>
<td>85M</td>
<td><strong>8.7B</strong></td>
<td><strong>6 days/64 k40s</strong></td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>2.79</td>
<td>39.22</td>
<td>214M</td>
<td>278M</td>
<td>6 days/96 k80s</td>
</tr>
<tr>
<td>GNMT+RL (Wu et al., 2016)</td>
<td>2.96</td>
<td>39.92</td>
<td>214M</td>
<td>278M</td>
<td>6 days/96 k80s</td>
</tr>
<tr>
<td>PBMT (Durrani et al., 2014)</td>
<td></td>
<td></td>
<td></td>
<td>37.0</td>
<td></td>
</tr>
<tr>
<td>LSTM (6-layer) (Luong et al., 2015b)</td>
<td></td>
<td></td>
<td></td>
<td>31.5</td>
<td></td>
</tr>
<tr>
<td>LSTM (6-layer+PosUnk) (Luong et al., 2015b)</td>
<td></td>
<td></td>
<td></td>
<td>33.1</td>
<td></td>
</tr>
<tr>
<td>DeepAtt (Zhou et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td>37.7</td>
<td></td>
</tr>
<tr>
<td>DeepAtt+PosUnk (Zhou et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td>39.2</td>
<td></td>
</tr>
</tbody>
</table>
Conv sequence to sequence model

- Convolutional Sequence to Sequence Learning, by Jonas Gehring et al
- Main difference from GNMT, lstm is replaced by gated linear units

\[ v([A \ B]) = A \otimes \sigma(B) \]

- Other difference from GNMT
  - Attention at every decoding layer
  - Dot product based attention
  - Position embedding
Embeddings

Convolutions

Gated Linear Units

Attention

Dot products

They agree

Sie stimmen zu
Conv sequence to sequence learning

- Very nice results
- Appears to be very sensitive to params initialization, hard to reproduce

<table>
<thead>
<tr>
<th>WMT’14 English-French</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wu et al. (2016) GNMT (Word 80K)</td>
<td>37.90</td>
</tr>
<tr>
<td>Wu et al. (2016) GNMT (Word pieces)</td>
<td>38.95</td>
</tr>
<tr>
<td>Wu et al. (2016) GNMT (Word pieces) + RL</td>
<td>39.92</td>
</tr>
<tr>
<td>ConvS2S (BPE 40K)</td>
<td>40.51</td>
</tr>
</tbody>
</table>
Transformer Model

- *Attention is all you need*, by Ashish and Noam from the Google Brain team.
- **Building blocks**
  - Multi-headed Self-attention
  - Multi-layer Multi-headed Attention
  - Hand engineered position embedding
  - Residual connections and layer normalization to stabilize model training
- It is the current state of the art
Figure 1: The Transformer - model architecture.
Three ways of attention

- Encoder-Decoder Attention
- Encoder Self-Attention
- Masked Decoder Self-Attention
Self-Attention

Convolution

Self-Attention
Self-Attention

Convolution

Self-Attention
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td>ByteNet [17]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [37]</td>
<td></td>
<td>39.2</td>
</tr>
<tr>
<td>GNMT + RL [36]</td>
<td>24.6</td>
<td>39.92</td>
</tr>
<tr>
<td>ConvS2S [9]</td>
<td>25.16</td>
<td>40.46</td>
</tr>
<tr>
<td>MoE [31]</td>
<td>26.03</td>
<td>40.56</td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [37]</td>
<td></td>
<td>40.4</td>
</tr>
<tr>
<td>GNMT + RL Ensemble [36]</td>
<td>26.30</td>
<td>41.16</td>
</tr>
<tr>
<td>ConvS2S Ensemble [9]</td>
<td>26.36</td>
<td>41.29</td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>38.1</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td><strong>28.4</strong></td>
<td><strong>41.0</strong></td>
</tr>
</tbody>
</table>
Further reading

- [Neural Machine Translation](#), book chapter
Speech Recognition
Sequence models in ASR

● Covers a few popular end-to-end ASR models: CTC, RNN-T, LAS
● CTC is by Alex Graves
  ○ *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*, by Alex Graves
● RNN-T is by Alex Graves
  ○ *Sequence Transduction with Recurrent Neural Networks*, by Alex Graves
● LAS is by Google Brain and Yoshua’s group
  ○ *Listen, Attend and Spell*, by William Chan et al
  ○ *Attention-Based Models for Speech Recognition*, by Jan Chorowski et al
Speech frontend

- To transform the raw speech signal into a format that is more suitable for ASR
  - Keeps only the essential information in speech
- Trend is speech frontend is getting simpler and simpler
  - Modern ASR system is capable of extracting useful information from noisy signal directly.
  - Complex frontend runs the risk of throwing away useful information
- Research suggest that you can even get rid of the ASR directly.
  - Learning the Speech Front-end With Raw Waveform CLDNNs, by Tara and et al.
Log mel filter bank energies

- Take a short time Fourier transform of the speech signal with an appropriate window size and shift. Typical window size and shift are 25ms and 10ms respectively.
- Convert frequency energies from linear scale to a mel scale, using triangular overlapping windows.
- Take the log of the energy
CTC model

- *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*, by Alex Graves
- It is a quite old model, proposed in 2005
- Used in Baidu’s DeepSpeech and DeepSpeech2 systems
Alignment

- Model output is frame synchronous
  - The model output one label symbol at every frame
- To match the output sequence to the input sequence in length
  - blank symbol, “_”
  - repeated symbols

The cat sat on the mat

Label sequence:
```
t  h  h  _  _  e  e  _  _  m  _  a  t  t
```

Alignment #1:
```
t  t  _  h  e  e  _  _  m  m  a  _  t  _  _
```

Alignment #2:
```
t  t  _  h  e  e  _  _  m  m  a  _  t  _  _
```

Spectrogram
CTC loss

- CTC loss assumes “temporal independence” in the output layer
- Given an alignment of label sequence to input sequence, $P(\text{alignment} \mid \text{input})$ is trivial to compute
  - It is simply the product of $P(\text{label symbol})$ at each frame
- $P(\text{label} \mid \text{input}) = \sum_{\text{all valid alignments}}(P(\text{alignment} \mid \text{input}))$
  - Exponentially many valid alignments
  - Dynamic programming algorithm available to compute this sum exactly
- CTC loss = $E_{\{\text{label, input}\}} \log(P(\text{label} \mid \text{input}))$
Figure 3. illustration of the forward backward algorithm applied to the labelling ‘CAT’. Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.
CTC decoding

- **Greedy decoding**
  - Find an alignment such that $P(\text{alignment}|\text{input})$ is maximized
  - Take the most probable symbol at each time step
  - Remove repeated symbols and blank symbols

- **Dynamic programming based decoding**
  - Find the label sequence such that $P(\text{label}|\text{input})$ is maximized
  - Can use an algorithm similar to the one used in training
Nice extension of the CTC model

- **Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling**, by Hairong Liu et al
- Segmentation is fixed in standard CTC, e.g. “CAT” to “C”, “A” and “T”.
- Key idea is to allow the network to learn a valid segmentation at the same time
  - CAT -> “C”, “A”, “T”
  - CAT -> “C”, “AT”
  - CAT -> “CA”, “T”
  - CAT -> “CAT”
Figure 1. Illustration of the states and the forward-backward transitions for the label ‘CAT’. Here we let $G$ be the set of all uni-grams and bi-grams of the English alphabet. The set of all valid states $S$ for the label $l = ‘CAT’$ are listed to the left. The set of states and transitions that are common to both vanilla and Gram-CTC are in black, and those that are unique to Gram-CTC are in orange. In general, any extension that collapses back to $l$ is a valid transition - For example, we can transition into (‘CAT’, 1) from (‘CAT’, 1), (‘CA’, 2), (‘CA’, 1) and (‘CA’, 0) but not from (‘CAT’, 0) or (‘CAT’, 2).
CTC

- It is fairly stable to train
  - Guaranteed monotonicity in alignment
- Naturally produces an alignment as a by-product
  - Can be useful for other applications
- Very weak language model
  - Usually need to combine with a strong external language model for it to perform well
RNN-T

- Proposed to address the limitations in the CTC model, namely the temporal independence assumption
- Two recurrent neural networks
  - One on the source sequence
  - The other on the label sequence
- Target side RNN allows the model to learn a much better build-in language model
- A separate joint network on top to combine the source sequence RNN and the target sequence RNN
RNN-T

Source RNN

Target RNN

Joint network

$u \times t$ joint networks organized into a lattice
RNN-T

- A special symbol “null” is introduced
- When “null” is generated, the model moves one step forward in the source sequence
- At training time, dynamic programming is used to sum up probabilities in all the path
- Each prediction path is \((u + t)\) steps long
RNN-T

- Definitely stronger built-in language model
- It is harder to train, but one can pre-train the source RNN with a CTC loss, and target RNN using a LM loss on some large text corpus
- Decoder RNN is completely independent of the source
Attention based models

- Very similar to the translation model
  - speech as input instead of text
- Gaining popularity
  - Attention-Based Models for Speech Recognition, by Jan Chorowski et al
  - Listen, Attend and Spell, by William Chan et al
  - Very Deep Convolutional Networks for End-to-End Speech Recognition, by Yu Zhang et al
  - Latent Sequence Decompositions, by William Chan et al
- Research from my group suggest attention based models are very promising for ASR
Attention based models

- Offers the best modeling flexibility
  - Decoder has strictly more information as it conditions on the encoding of the source
  - No artificial statistical assumptions, like temporal independence assumption in CTC, or the independence assumption among the joint networks in RNN-T

- In theory should be the best one to use
  - In practice depends very much on model tuning as well
LAS

- Listen: Acoustic model
- Attend: Dynamic time warping
- Spell: language model


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence $x$ into high level features $h$, the speller is an attention-based decoder generating the $y$ characters from $h$. 

Grapheme characters $y_i$ are modelled by the CharacterDistribution

AttentionContext creates context vector $c_i$ from $h$ and $s_i$
How much would a woodchuck chuck
Comparing Sequence-to-Sequence Models

Attention-based models have the encoder feeding the decoder with acoustic information.
A Comparison of Sequence-to-Sequence Models for Speech Recognition,

Table 1: WERs (%) on various test sets for the models compared in this work. The attention-based model with two decoder layers is the single best sequence-to-sequence model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean</th>
<th>Noisy</th>
<th>numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dict</td>
<td>vs</td>
<td>dict</td>
</tr>
<tr>
<td>Baseline Uni. CDP</td>
<td>6.4</td>
<td>9.9</td>
<td>8.7</td>
</tr>
<tr>
<td>Baseline BiDi. CDP</td>
<td>5.4</td>
<td>8.6</td>
<td>6.9</td>
</tr>
<tr>
<td>End-to-end systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTC-grapheme³</td>
<td>39.4</td>
<td>53.4</td>
<td>-</td>
</tr>
<tr>
<td>RNN Transducer</td>
<td>6.6</td>
<td>12.8</td>
<td>8.5</td>
</tr>
<tr>
<td>RNN Trans. with att.</td>
<td>6.5</td>
<td>12.5</td>
<td>8.4</td>
</tr>
<tr>
<td>Att. 1-layer dec.</td>
<td>6.6</td>
<td>11.7</td>
<td>8.7</td>
</tr>
<tr>
<td>Att. 2-layer dec.</td>
<td><strong>6.3</strong></td>
<td><strong>11.2</strong></td>
<td><strong>8.1</strong></td>
</tr>
<tr>
<td>Architecture</td>
<td>SWBD WER</td>
<td>CH WER</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Iterated-CTC [29]</td>
<td>11.3</td>
<td>18.7</td>
<td></td>
</tr>
<tr>
<td>BLSTM + LF MMI [21]</td>
<td>8.5</td>
<td>15.3</td>
<td></td>
</tr>
<tr>
<td>LACE + LF MMI $^4$ [28]</td>
<td>8.3</td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td>Dilated convolutions [25]</td>
<td>7.7</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>CTC + Gram-CTC [17]</td>
<td>7.3</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>BLSTM + Feature fusion[23]</td>
<td>7.2</td>
<td>12.7</td>
<td></td>
</tr>
<tr>
<td>CTC [17]</td>
<td>9.0</td>
<td>17.7</td>
<td></td>
</tr>
<tr>
<td>RNN-Transducer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beam Search NO LM</td>
<td>8.5</td>
<td>16.4</td>
<td></td>
</tr>
<tr>
<td>Beam Search + LM</td>
<td>8.1</td>
<td>17.5</td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beam Search NO LM</td>
<td>8.6</td>
<td>17.8</td>
<td></td>
</tr>
<tr>
<td>Beam Search + LM</td>
<td>8.6</td>
<td>17.8</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1.** WER comparison against previous published results on Fisher-Switchboard Hub5’00 benchmark using *in-domain* data. We only list results using single models here. All the previous works reported WER using language models. We don’t leverage any speaker information in our models, though it has been shown to reduce WER in previous works [28, 25].

Speech Synthesis
Speech synthesis

● It is the reverse problem of speech recognition
● Existed for a long time
● Straightforward solution is Concatenative TTS
  ○ Stitch together pieces of recorded speech, with smoothing at the boundaries
  ○ Quite intelligible
  ○ Prosody is not natural, sounds very robotic
  ○ Widely used
Neural based TTS

- Fully neural network based TTS system is getting popular
  - **Wavenet** from Google DeepMind
  - **TacoTron** from Google
  - **DeepVoice** from Baidu
  - **VoiceLoop** from Facebook
- Wavenet can be seen as a high quality vocoder
- TacoTron, DeepVoice and VoiceLoop are all end to end TTS systems
- Will focus on DeepVoice and TacoTron in this talk
DeepVoice

- Follows more traditional approach
- It is a pipelined system
  - text -> phoneme
  - phoneme -> duration
  - phoneme + duration -> Pitch
  - Wavenet like vocoder
DeepVoice

Figure 1: Inference system diagram: first text-phonemes dictionary conversion, second predict phoneme durations, third upsample and generate $F_0$, finally feed $F_0$ and phonemes to vocal model.
TacoTron

- It is just one end to end model
- The model predicts linear spectrograms from text directly
  - So no explicit duration model, pitch model and etc
- Used fixed Griffin-Lim algorithm to convert from spectrograms to speech
  - Griffin-Lim as inverse FFT
- Could benefit from a better vocoder
  - e.g. Wavenet
- Some examples can be found:
  - https://google.github.io/tacotron/
Figure 1: Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.
Other applications of sequence models
What else we can do with sequence models?

- Image captioning
  - *Show and Tell: A Neural Image Caption Generator*, by Oriol Vinyals
- Syntactic constituency parsing
  - *Grammar as a Foreign Language*, by Oriol Vinyals
- Handwriting synthesis
  - *Generating Sequences With Recurrent Neural Networks*, by Alex Graves
- Many more ...
Image Captioning

\[ p(\text{English} \mid \text{French}) \]

\[ p(\text{English} \mid \text{Image}) \]

A man holding a tennis racquet on a tennis court.

A group of young people playing a game of Frisbee

Two pizzas sitting on top of a stove top oven

A man flying through the air while riding a snowboard
Syntactic parsing

John has a dog. $\rightarrow$ (S (NP NNP)_{NP} (VP VBZ (NP DT NN)_{NP})_{VP} . )_{S}

Figure 2: Example parsing task and its linearization.
Handwriting synthesis
Conclusion

- Introduced many popular sequence models for language modeling, for machine translation, automatic speech recognition, speech synthesis, and others
- Advanced topics that I didn’t cover
  - Scheduled sampling
  - Sequence training to directly optimize for the final metric
  - Online sequence models
  - Language model integration into NMT, ASR and etc
Q & A