Neural Models for Sequence Prediction --- Recurrent Neural Networks

Sunita Sarawagi
IIT Bombay
sunita@iitb.ac.in
Sequence Modeling taks

- Classify sequences $x_1, \ldots, x_n \rightarrow y$. E.g. sentiment classification, normal versus abnormal traffic
- Next word in a sequence $x_1, \ldots, x_n \rightarrow x_{n+1}$. E.g. language modeling
- Label per token in sequence $x_1, \ldots, x_n \rightarrow y_1, \ldots, y_n$. E.g. POS, speech recognition.
- Sequence prediction tasks $x \rightarrow y_1, \ldots, y_m$ E.g. Translation, conversation assistant.
More examples

- Forecasting (Time Series)

\[ y_t, y_{t-1}, \ldots, y_L \] outputs e.g.: demand for an item

\[ x_1, x_2, \ldots, x_t \] inputs

\[ x_t : \text{[day of the week, holiday?, temperature, month]} \]
RNN: Recurrent Neural Network

- A model to process variable length 1-D input
- In CNN, each hidden output is a function of corresponding input and some immediate neighbors.
- In RNN, each output is a function of a 'state' summarizing all previous inputs and current input. State summary computed recursively.
- RNN allows deeper, longer range interaction among parameters than CNNs for the same cost.
RNNs: Basic type

- Notation:
  - $h_t$ to denote state instead of $z_t$
  - Input to RNN is $x_t$, instead of $y_t$
RNN: forward computation example.

\[ x_1, y_1, \quad x_i \in \mathbb{R} \]

\[ x_2, y_2 \]

\[ \vdots \]

\[ x_t, y_t, \quad h^0 = 0 \]

\[ t=1 \]

\[ \sigma(Wh^0 + Ux_1 + b) = h^{(1)} \]

\[ O^{(1)} = Vh^{(1)} + c \]

Loss \[ = L_1 = (y_1 - O^{(1)})^2 \]

\[ t=2 \]

\[ \sigma(Wh^{(1)} + Ux_2 + b) = h^{(2)} \]

\[ O^{(2)} = Vh^{(2)} + c \]

\[ L_2 = (y_2 - O^{(2)})^2 \]
RNN for text (Predict next word) – word embeddings

$x_t \in \text{word} \in [1 \ldots V]$

Vocabulary size:
$V = 30k$

$x_t \in \mathbb{R}^V$

Learned parameters

$S = V \times d$

Typically $d = 300$
Training a sequence model

- **Maximum Likelihood**
  \[ P(y|x, \theta) = \prod_{t=1}^{n} P(y_t|y_1, \ldots, y_{t-1}, x, \theta) \]

- **Mechanism of training**
  - Input to RNN is the true tokens upto time t-1
  - Output is the probability distribution over tokens
  - Maximize the probability of the correct token.

- **Advantages**
  - Easy. Generative --- token at a time. Sound-- full dependency!
Training data

\[ D = \{(x^i_1, y^i_1), \ldots, (x^i_N, y^i_N)\} \]

LM: \( y^i = x^i \) shifted by one to the left

Example \( x^i \): The cat is sleeping

\( y^i \): cat is sleeping EOS
Training RNN parameters

Backpropagation through time

- Unroll graph along time
- Compute gradient through back-propagation exactly as in feedforward networks
- Sum up the gradient from each layer since parameters are shared.
Backpropagation through time

\[ L = L_{t-1} + L_t \]

\[ \frac{\partial L}{\partial V} = \frac{\partial L_{t-1}}{\partial V} + \frac{\partial L_t}{\partial V} \]

\[ = \frac{\partial L_{t-1}}{\partial O_{t-1}} O_{t-1}^{h-1} \frac{\partial h}{\partial O_{t-1}} + \frac{\partial L_t}{\partial h} h_t \]

\[ \frac{\partial L}{\partial W} = \frac{\partial L_{t-1}}{\partial W} + \frac{\partial L_t}{\partial W} \]

\[ = \frac{\partial L_{t-1}}{\partial O_{t-1}} \frac{\partial O_{t-1}}{\partial W} O_{t-1}^{h-1} \frac{\partial h}{\partial O_{t-1}} + \frac{\partial L_t}{\partial h} \frac{\partial h}{\partial W} + \frac{\partial L_t}{\partial O_t} \frac{\partial O_t}{\partial W} \frac{\partial h}{\partial O_t} \]

\[ = O(W^2) \]
Exploding and vanishing gradient problem

Product of non-linear interactions: gradient either small or large $w^t$
Fixes for vanishing/exploding gradient problem

- No parameters for updating state: state is a "reservoir" of all past inputs, output is a learned function of state. E.g. Echo state networks, Liquid networks
- Multiple time scales: add direct connection from far inputs instead of depending on state to capture all far-off inputs.
- Shortcomings of above:
  - How far back we look at each t is same for all t and cannot be changed for different times or different inputs
  - Only accumulate information, cannot forget information.
- Solution: Gated RNNs e.g. LSTMs
Gated RNNs

- Gates control which part of the long past is used for current prediction
- Gates also allow forgetting of part of the state
- LSTM: Long Short Term Memory, one of the most successful gated RNNs.
- An excellent introductions here:
  - [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
  - [http://blog.echen.me/2017/05/30/exploring-lstms/](http://blog.echen.me/2017/05/30/exploring-lstms/)
The sequence prediction task

- **Given a complex input** $x$
  - Example: sentence(s), image, audio wave

- **Predict a sequence** $y$ of discrete tokens $y_1, y_2, \ldots, y_n$
  - Typically a sequence of words.
  - A token can be any term from a huge discrete vocabulary.
  - Tokens are inter-dependent
    - Not $n$ independent scalar classification task.

\[ x \overset{\text{Neural network}}{\rightarrow} y = y_1, y_2, \ldots, y_n \]
Motivation

- Applicable in diverse domains spanning language, image, and speech processing.
- Before deep learning each community solved the task in their own silos → lot of domain expertise
- The promise of deep learning: as long as you have lots of labeled data, domain-specific representations learnable
- This has brought together these communities like never before!
Translation

Where can I find healthy and traditional Indian food?

स्वस्थ और पारंपरिक भारतीय भोजन कहां मिल सकता है?

- Pre-DL translation systems were driven by transfer grammar rules painstakingly developed by linguists and elaborate phrase translation.
- Whereas, modern neural translation systems are scored almost 60% better than these domain-specific systems.
Image captioning

Context: *x*

Predicted sequence: *y*


- Early systems: either template-driven or transferred captions from related images
- Modern DL systems have significantly pushed the frontier on this task.
Conversation assistance

Context: x

Predicted sequences: y

From https://research.googleblog.com/search?updated-max=2016-06-20T05:00:00-07:00&max-results=7&start=35&by-date=false
Syntactic parsing

Context: x

Predicted sequence: y

John has a dog. ➔ (S (NP NNP )NP (VP VBZ (NP DT NN )NP )VP . )s
Speech recognition

Context: $x$ (Speech spectrogram)

Output: $Y$ (Phoneme Sequence)

Rice University
Challenges

● Capture long range dependencies
  ○ No conditional independencies assumed
  ○ Example during correct anaphora resolution in output sentence:
    ■ How is your son? I heard he was unwell.

● Prediction space highly open-ended
  ○ No obvious alignment with input unlike in tasks like POS, NER
  ○ Sequence length not known. Long correct response has to compete with short ones
    ■ How are you?
      ● “Great” Vs “Great, how about you?”
The Encoder Decoder model for sequence prediction

- Encode \( x \) into a fixed-D real vector \( X \)
- Decode \( y \) token by token using a RNN
  - Initialize a RNN state with \( X \)
  - Repeat until RNN generates a EOS token
    - Feed as input previously generated token
    - Get a distribution over output tokens, and choose best.
The Encoder Decoder model for sequence prediction

- Encode \( x \) into a fixed-D real vector \( X \)
- Since \( Y \) has many parts, need a graphical model to express the joint distribution over constituent tokens \( y_1, \ldots, y_n \).

Specifically, we choose a special Bayesian network, called a RNN

\[
P(y|x, \theta) = \prod_{t=1}^{n} P(y_t|y_1, \ldots, y_{t-1}, x, \theta)
\]
Encoder decoder model

- Let $v_x$ = fixed dimensional vector summary of input $x$
- Difficult to define CPD $Pr(y_t|y_1, \ldots, y_{t-1}, v_x)$ with variable length of parents. Need to share parameters.
- Redesign the BN by summarizing parents as state.

$$Pr(y_t|y_1, \ldots, y_{t-1}, v_x, \theta) = P(y_t|z_t, \theta), \quad (1)$$

where $z_t$ is a state vector implemented using a recurrent neural network as

$$z_t = \begin{cases} v_x & \text{if } t = 0, \\ \text{RNN}(z_{t-1}, y_{t-1}, \theta_R) & \text{otherwise}. \end{cases} \quad (2)$$
Encoder-decoder model

- Models full dependency among tokens in predicted sequence
  - Chain rule \( P(y|x, \theta) = \prod_{t=1}^{n} P(y_t|y_1, \ldots, y_{t-1}, x, \theta) \)
  - No conditional independencies assumed unlike in CRFs

- Training:
  - Maximize likelihood. Statistically sound!

- Inference
  - Find \( y \) with maximum probability \( \rightarrow \) intractable given above
  - Beam search: branch & bound expansion of frontier of ‘beam width’
    - Probability of predicted sequence increases with increasing beam width.
Inference

- Finding the sequence of tokens $y_1, \ldots, y_n$ for which product of probabilities is maximized
- Cannot find the exact MAP efficiently since fully connected Bayesian network $\Rightarrow$ intractable junction tree. The states $z$ are high-dimensional real-vectors.
- Solution: approximate inference
  - Greedy
  - Beam-search
Encoder-decoder for sequence to sequence learning

Context: $x$

Predicted sequence: $y$

Choose high probability token and feed to next step.

$\Pr(y_t|y_1, \ldots, y_{t-1}, x)$

RNN to generate $y$

RNN e.g. LSTMs to summarize $x$ token-by-token

Embedding layer to convert each word to a fixed-D real vector

Where does the encoder-decoder model fail?

- Single vector cannot capture enough of input.
  - Fix: Attention (Bahdanau 2015, several others)

- Slow training: RNNs processed sequentially, replace with
  - CNN (Gehring, ICML 2017)
  - Transformer (Self Attention(Vaswani, June 2017))

- Training loss flaws
  - Global loss functions
Single vector not powerful enough ---> revisit input

Deep learning term for this ⇒ Attention!

How to learn attention automatically, and in a domain neutral manner?

Single vector not powerful enough ---> revisit input

Deep learning term for this ⇒ Attention!

H = हाल, के, वर्ष, में, आर्थिक, विकास, धीमा,

End-to-end trained and magically learns to align automatically given enough labeled data

Example of attention in translation

Nice animated explanations for attention.

https://distill.pub/2016/augmented-rnns/#attentional-interfaces
Same attention logic applies to other domains too

Attention over CNN-derived features of different regions of image

Attention in image captioning. Attention over CNN

A bird flying over a body of water.

A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a...
Attention in Speech to Text Models

We see that attention is focussed in middle part and nicely skips the prefix and suffix that is silence.

Diagram from https://distill.pub/2016/augmented-rnns/
Google’s Neural Machine Translation (GNMT) model

- 8 layers
- Residual connections
- Bidirectional LSTMs
- Special wordpiece tokenization to handle rare words
- 2-layer attention logic
- Length normalization, coverage penalty, low-precision inference
- Works on many language pairs
- 60% better than existing phrase based system on human evaluation.
## Results

<table>
<thead>
<tr>
<th></th>
<th>PBMT</th>
<th>GNMT</th>
<th>Human</th>
<th>Relative Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>English → Spanish</td>
<td>4.885</td>
<td>5.428</td>
<td>5.504</td>
<td>87%</td>
</tr>
<tr>
<td>English → French</td>
<td>4.932</td>
<td>5.295</td>
<td>5.496</td>
<td>64%</td>
</tr>
<tr>
<td>English → Chinese</td>
<td>4.035</td>
<td>4.594</td>
<td>4.987</td>
<td>58%</td>
</tr>
<tr>
<td>Spanish → English</td>
<td>4.872</td>
<td>5.187</td>
<td>5.372</td>
<td>63%</td>
</tr>
<tr>
<td>French → English</td>
<td>5.046</td>
<td>5.343</td>
<td>5.404</td>
<td>83%</td>
</tr>
<tr>
<td>Chinese → English</td>
<td>3.694</td>
<td>4.263</td>
<td>4.636</td>
<td>60%</td>
</tr>
</tbody>
</table>
Summary

- Deep learning based models for sequence prediction has revolutionized and unified many diverse domains.
- 2015-2018 has seen several improvements to the encoder-decoder method
  - Increase capacity via input attention
  - Eschew RNN bottleneck via multi-layer self-attention
  - Fix loss function via better calibration and global conditioning
- Other interesting developments not covered
  - Memory networks for remembering rare events (Kaiser, ICLR 2017)
What next?

- Move away from black-box, batch-trained, monolithic models to transparent models with more control from humans and evolving continuously.

- Generalize to other structured learning tasks
  - No natural ordering of variables.
Thank you.
Where does the encoder-decoder model fail?

- Single vector cannot capture enough of input
  - Fix: Attention

- Slow training: RNNs processed sequentially, replace with
  - CNN (Gehring, ICML 2017)
  - Attention (Vaswani, June 2017)

- Training loss flaws
  - Systematic bias against long sequences
  - Not aligned with whole sequence error during inference
    - Generate sequences during training, score their errors and minimize
Attention is enough. No need for RNN

Attention weighted sum of previous layer

Edge weights determined by self-attention. Multiple of these

Positional embedding of each input word

Sum up word and position embedding

Compute position embedding, lookup word embedding

One-hot word, and position(1,2,..)
6 of these to capture different granularity of bindings among input tokens.

Repeat similar 6-layers to replace RNN for decoder too and between decoder and encoder.

Tokens at all positions processed in parallel --- only sequentiality among the 6 layers which are fixed.

Author’s slides https://www.slideshare.net/ilblackdragon/attention-is-all-you-need
Example: how attention replaces RNN state

Attention around “making” converts it to phrase “making more difficult”
RNNs/CNNs no longer indispensable for sequence prediction

Attention captures relevant bindings at much lower cost
Where does the encoder-decoder model fail?

- Single vector cannot capture enough of input.
  - Fix: Attention

- Slow training: RNNs processed sequentially, replace with
  - CNN (Gehring, ICML 2017)
  - Attention (Vaswani, June 2017)

- Training loss flaws
  - Poor calibration
  - Not aligned with whole sequence error during inference
    - Generate sequences during training, score their errors and minimize
Bias against longer sequences

26% ED predictions of zero length. None in data.

ED over-predicts short sequences

Severely under-predicts large sequences
Surprising drop in accuracy with better inference

For long sequences, accuracy drops when inference predicts a higher scoring sequence ---- why?
Two Causes

1. Lack of calibration
2. Local conditioning
Lack of calibration

- Next token probabilities not well-calibrated.
  - A 0.9 probability of $y_t = \text{"EOS"}$, does not imply 90% chance of correctness.
- Bane of several modern neural architectures e.g. Resnets, not just sequence models
  - High in accuracy but low in reliability!
    - Mostly over-confident.
  - See: On Calibration of Modern Neural Networks, ICML 2017
Calibration plots
Investigating reasons for poor calibration
Reasons for poor calibration

● Observations
  a. End of sequence token is seriously over-confident
  b. Calibration is worse when encoder attention is diffused.
  c. Other unexplained reasons.
Kernel embedding based trainable calibration measure

- Train models to minimize weighted combination of 0/1 error and calibration of confidence scores.
Corrected calibrations
Fixing calibration leads to higher accuracy

1. Beam search for predicting highest probability sequence
   a. Grows token-by-token a beam of highest scoring prefixes
   b. Poor calibration misleads beam-search
Two Causes

1. Lack of calibration
2. Local conditioning
Problems of local conditioning

Local conditioning causes the log-probability of each correct token to saturate (get very close to zero) even when the correct sequence does not have the highest probability.

\[
P(y|x, \theta) = \prod_{t=1}^{n} P(y_t|y_1, \ldots, y_{t-1}, x, \theta)
\]
Local conditioning for sequence prediction

\[ \log \Pr(y_t | y_1, \ldots, y_{t-1}) \]

<table>
<thead>
<tr>
<th>t=</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>-0.01</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-6</td>
</tr>
<tr>
<td>1</td>
<td>-6</td>
<td>-0.4</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-6</td>
</tr>
<tr>
<td>0</td>
<td>-6</td>
<td>-1.4</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-6</td>
</tr>
<tr>
<td>E</td>
<td>-6</td>
<td>-1.8</td>
<td>-1.7</td>
<td>-1.6</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-0.01</td>
</tr>
</tbody>
</table>

Positive sequence: “S,1,1,1,1,1,1,E”, Negative sequence: “S,0,E”.

Margin between position and negative sequence optimized by ED local loss is \(-0.4 - (-1.4) = 1!\)

Log-probability of positive sequence = -1.9
Log-probability of negative sequence = -0.4
Margin between positive and negative sequence = -1.5!
ED objective is zero even when prediction is wrong

More training data will not help if your training loss is broken!
How to fix the ED training loss?

Avoid local conditioning, use global conditioning

$$\Pr(Y_i|X_i, \theta) = \frac{e^{S(Y_i|X_i, \theta)}}{\sum_{Y \in \text{sample}} e^{S(Y|X_i, \theta)}}$$

Use for

- Applications, like conversation where response restricted to be from a whitelist of responses
- Else, sample responses adaptively during training

More details in Length bias in Encoder Decoder Models and a Case for Global Conditioning by Siege and Sarawagi. EMNLP’16
Results

Global conditioning predicts long sequences whereas ED predicts none.

A method using global conditioning is more accurate.

Length normalized encoder-decoder models.
Thank you!
Properties of a good loss function for training

- **Scoring models**

  \[(X, Y) \rightarrow \text{Model } (\theta) \rightarrow S(Y|X, \theta) \in \mathbb{R}\]

- **Inference:** find \(Y\) with highest score

- **Training:** minimize loss per labeled instance \(\{(X_i, Y_i)\}\)
  - If loss \(\sim 0\), then correct output \(Y_i\) has the highest score.
  - Not true for encoder decoder models!
Peculiar biases of predictions from ED model

● ED over-predicting short sequences
  ○ Even after accounting for the fact that short messages are more common given any particular context.
● Increasing the beam width sometimes decreased quality!

These observations are on models trained with billions of examples for a conversation task.
Datasets

- Reddit – comments on user posts
  - 41M posts, 501M comments
- Open Subtitles – subtitles on non-English movies
  - 319M lines of text

For each data set:

- 100K top messages = predicted set.
- 20K top tokens used to encode tokens into ids.