Introduction to Seq2Seq Modeling

Bill Watson

S&P Global

November 22, 2019

What are Sequence to Sequence Models?

-

Common Applications for Seq2Seq Models

- Machine Translation
- Speech Recognition
- Text Summarization

Encoder-Decoder Architecture

Overview: Encoders and Decoders

- We have 2 sub-networks: an **Encoder** and a **Decoder**
- Encoders
  - Give the source sentence meaning
- Decoders
  - Given the source sentence, emit a variable-length sequence
- We will discuss how to connect the two for joint training

Overview: Encoders and Decoders

- Encapsulation allows for flexible design choices:
  - Embeddings
    - Pre-trained
    - DIY
  - Recurrent Layer
    - Type
    - Depth
    - Directionality

What makes an Encoder?

What makes a Decoder?

Word Embeddings: Practical Considerations

Recap: Word Embeddings

Recap: Pretrained Word Embeddings

- Co-occurence Matrix
- Pointwise Mutal Information
- SVD Co-occurence
- Ngram
- CBOW, Skip Gram
- GloVe
- Sentence Embeddings: ELMo

DIY Embeddings

- We can always train our own!

Special Tokens for Sequence Modeling

- PAD $\rightarrow$ Padding / Masking
- UNK $\rightarrow$ Unkown words
- SOS $\rightarrow$ Start of Sentence
- EOS $\rightarrow$ End of Sentence

Mangaging the Vocab Size

- Languages are unevely distributed
- Many rare words, names $\rightarrow$ inflates the size of the vocabulary
- **Problem:**
  - Large embedding matricies for source, target language
  - Large output layers for prediction and softmax
- Naive Solution: Limit the vocab size to most frequent

Morphology, Compounding, and Transliteration

- Morphological Analysis

  tweet, *tweets*, *tweeted*, *tweeting*, *retweet*, . . .

- Compund Splitting
  - homework $\rightarrow$ home work
  - website $\rightarrow$ web site
- Names, Places, Proper Nouns
  - Hoboken, Baltimore, Obama, Michelle
  - Can do Transliteration

Handling Numbers

- Do we really need to encode every number? **NO!**

  I pay 950.00 in May 2007 $>$ I pay 2007 in May 950.00

Handling Numbers

- Do we really need to encode every number? **NO!**

  I pay 950.00 in May 2007 $>$ I pay 2007 in May 950.00

- Solution 1: Replace with a NUM token, but

  I pay NUM in May NUM $=$ I pay NUM in May NUM

Handling Numbers

- Do we really need to encode every number? **NO!**

  $$\text{I pay 950.00 in May 2007} > \text{I pay 2007 in May 950.00}$$

- Solution 1: Replace with a NUM token, but

  $$\text{I pay NUM in May NUM} = \text{I pay NUM in May NUM}$$

- Solution 2: Replace each digit with a unique symbol, e.g. 5

  $$\text{I pay 555.55 in May 5555} > \text{I pay 5555 in May 555.55}$$

- This reduces the need for embeddings, when we can simply do transliteration

Factored Decomposition

Backoff

Character Models

BPE Subwords

Modeling Recurrent Relations

Recap: Recurrent Layers

Vanilla RNNs

$$h_t = \tanh\left(\overbrace{W_{ih}x_t + b_{ih}}^{input} + \underbrace{W_{hh}h_{t-1} + b_{hh}}_{hidden}\right)$$

- h_t is the hidden state at time t
- x_t is the input at time t
- h_{t-1} is the previous hidden state
- h_0 is initialized to 0

Long Short Term Memory (LSTM)

$$
\text{Gates} \rightarrow \begin{cases} i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \end{cases}
$$

$$
\text{Outputs} \rightarrow \begin{cases} c_t = f_t * c_{t-1} + i_t * g_t \\ h_t = o_t * \tanh(c_t) \end{cases}
$$

- h_t is the hidden state at time t
- c_t is the cell state at time t
- x_t is the input at time t

Gated Recurrent Units (GRU)

$$
\text{Gates} \rightarrow \begin{cases} r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\ z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\ n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})) \end{cases}
$$

$$
\text{Outputs} \rightarrow \left\{ \; h_t = (1 - z_t) * n_t + z_t * h_{(t-1)} \right.
$$

- h_t is the hidden state at time t
- x_t is the input at time t
- h_{t-1} is the previous hidden state
- h_0 is initialized to 0

Aside: Different Perspective on Deep Recurrent Models

Aside: Dimensionality of Inputs and Outputs

Type	RNN	LSTM	GRU
In	B, L, H_{in}	B, L, H_{in}	B, L, H_{in}
h_{t-1}	$B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$
c_{t-1}	-	$B, N_L \cdot N_D, H_{out}$	-
h_t	$B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$
c_t	-	$B, N_L \cdot N_D, H_{out}$	-
Out	$B, L, N_D \cdot H_{out}$	$B, L, N_D \cdot H_{out}$	$B, L, N_D \cdot H_{out}$

- B is the batch size
- L is the sequence length
- N_D is the number of directions
- N_L is the number of layers
- H_{in}, H_{out} are the input and hidden size

Aside: The Influence of Padding in RNNs

Right to Left Sequence Modeling

- So far, we've only seen Left to Right Sequencing
- Why not Right to Left?

Bidirectional Sequence Modeling

- Why not both directions?
- We concatenate the results of each direction together

Putting together an Encoder

The Components of an Encoder

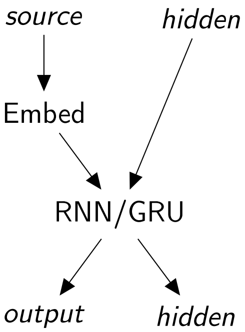


Figure: Model Architecture for Encoder.

Putting together an Decoder

The Components of a Decoder

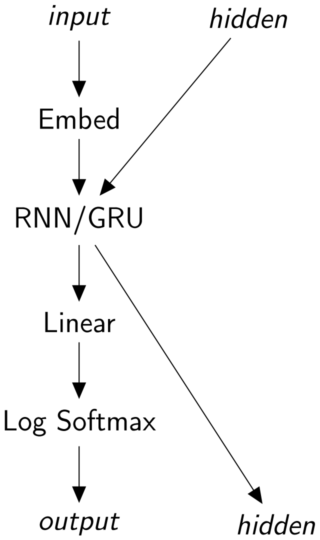


Figure: Decoder with No Attention.

Connecting the Encoder

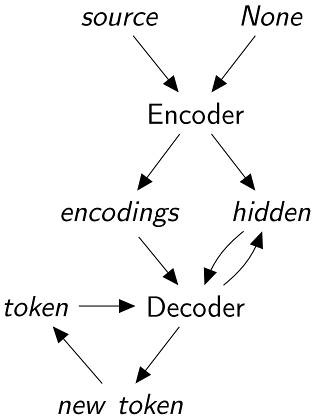


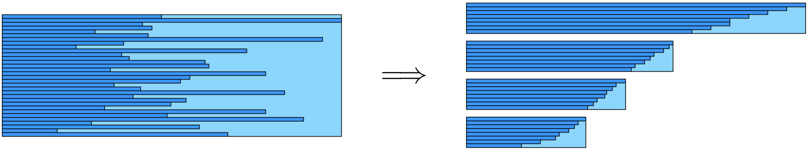
Figure: Model Architecture Overview for Encoder-Decoder.

Decoder Inference: Making Predictions

Teacher Forcing

Training Considerations

Increasing Throughput through Batching



- We can pad sentences of different lengths to increase batch size
- While also minimizing the use of padding
- Matrix Operations are faster

Cross Entropy and Label Smoothing

$$\text{loss}(\mathbf{x}_i, y_i) = -\underbrace{x_{y_i}}_{\textit{maximize}} + \overbrace{\log \sum_j \exp x_j}^{\textit{minimize}}$$

- Softmax and Cross-Entropy loss assign all the probability mass to a single word
  - LogSumExp is minimized on confident predictions
- Solution: smooth the distribution

Cross Entropy and Label Smoothing

- Softmax

$$p(y_i) = \frac{\exp x_{y_i}}{\sum_j \exp x_j}$$

- Smoothed Softmax with Temperature T

$$p(y_i) = \frac{\exp(x_{y_i}/T)}{\sum_j \exp(x_j/T)}$$

- As $T \to \infty$, the distribution is smoother

Visualizing Temperature

Masked Loss

Decoding: Making better Translations

Beam Search

Monte Carlo Beam Search

Ensembling

Reranking

Tools, References, and Further Reading

Refrences & Further Reading

- Machine Learning: A Probabilistic Perspective by Kevin Murphy