Introduction to Seq2Seq Modeling

Bill Watson

S&P Global

November 22, 2019

What are Sequence to Sequence Models?

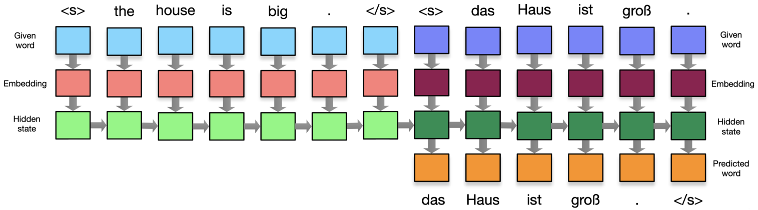
- Generally used to convert one set of tokens into another
  - Many - to - Many RNN
- Bleak view: map a sequence of indexes to another independent set of indexes

Encoder-Decoder Architecture

Overview: Encoders and Decoders

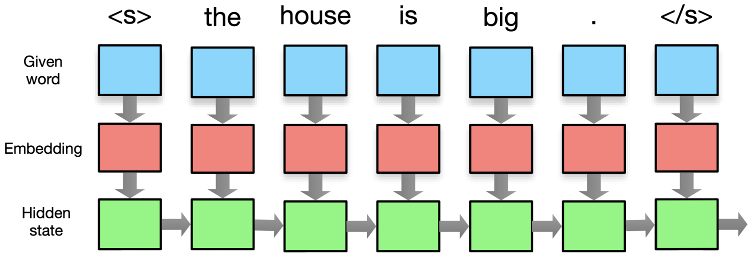
- We have 2 sub-networks: an **Encoder** and a **Decoder**
- Encoders
  - Give the source sentence meaning
- Decoders
  - Given the source sentence, emit a variable-length sequence
- We will discuss how to connect the two for joint training

Overview: Encoders and Decoders



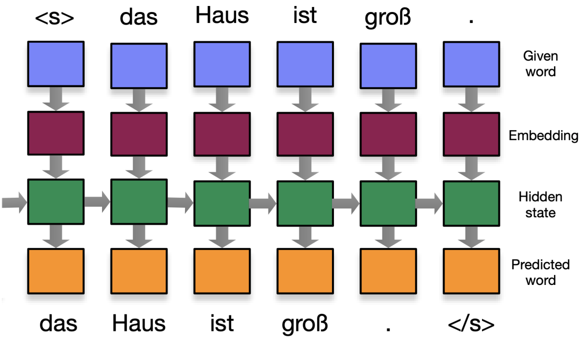
- Encapsulation allows for flexible design choices
- **Embeddings**
  - Pre-trained
  - DIY
- **Recurrent Layer**
  - Type
  - Depth
  - Directionality

What makes an Encoder?



- Recall: **Encoders** give the source sentence meaning
- Effectively a language model, without a layer to predict the next word
- Idea is to pass on the hidden state, and possibly use the encodings directly

What makes a Decoder?



- Recall: **Decoders** provide a new sequence conditioned on the Encoder's hidden state
- Starts with the Encoder's hidden state, and predicts one token at a time
- Re-feed the predicted token back into the decoder

Word Embeddings: Practical Considerations

Recap: Word Embeddings

$$
\begin{array}{rcl}
\texttt{the} & \rightarrow & \\
\texttt{cat} & \rightarrow & \\
\texttt{is} & \rightarrow & \\
\texttt{black} & \rightarrow &
\end{array}
\begin{bmatrix}
\texttt{1.23} & \texttt{-0.58} & \texttt{0.22} & \texttt{-0.80} & \texttt{0.61} \\
\texttt{-1.10} & \texttt{1.23} & \texttt{0.17} & \texttt{-0.21} & \texttt{1.43} \\
\texttt{-0.26} & \texttt{0.70} & \texttt{0.27} & \texttt{0.59} & \texttt{-1.04} \\
\texttt{-1.13} & \texttt{-0.81} & \texttt{-0.53} & \texttt{-0.59} & \texttt{0.26}
\end{bmatrix}
$$

- A trick to map tokens to vector representations

Recap: Pretrained Word Embeddings

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

DIY Embeddings

- We can always train our own!

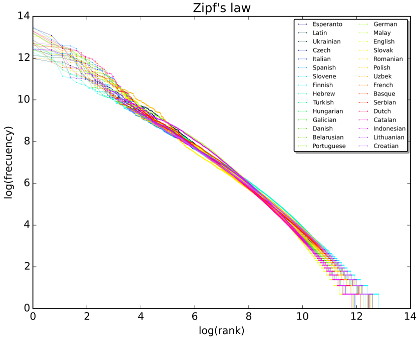
Special Tokens for Sequence Modeling

- ⟨PAD⟩ → Padding / Masking
- ⟨UNK⟩ → Unknown words
- ⟨SOS⟩ → Start of Sentence
- ⟨EOS⟩ → End of Sentence

⟨SOS⟩	Data	Science	is	the	⟨UNK⟩	!	⟨EOS⟩
⟨SOS⟩	I	⟨UNK⟩	for	S&P	.	⟨EOS⟩	⟨PAD⟩
⟨SOS⟩	Hello	World	!	⟨EOS⟩	⟨PAD⟩	⟨PAD⟩	⟨PAD⟩

Mangaging the Vocab Size

- Languages are unevenly distributed
- Many rare words, names $\rightarrow$ inflates the size of the vocabulary
- **Problem:**
  - Large embedding matrices 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*, ...

- Compound Splitting
  - homework → home·work
  - website → 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!**

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

- **Solution 1:** Replace with a $\langle$NUM$\rangle$ token, but

$\texttt{I pay}\ \langle\text{NUM}\rangle\ \texttt{in May}\ \langle\text{NUM}\rangle = \texttt{I pay}\ \langle\text{NUM}\rangle\ \texttt{in May}\ \langle\text{NUM}\rangle$

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 $\langle \text{NUM} \rangle$ token, but

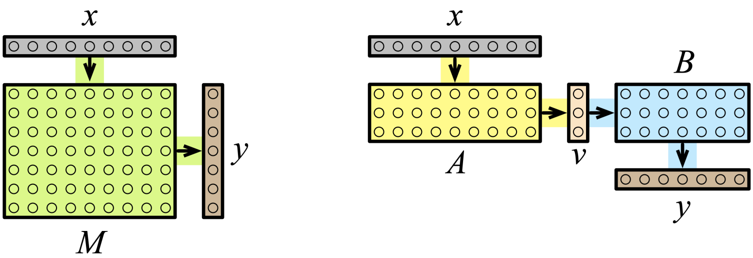
`I pay` $\langle \text{NUM} \rangle$ `in May` $\langle \text{NUM} \rangle$ $=$ `I pay` $\langle \text{NUM} \rangle$ `in May` $\langle \text{NUM} \rangle$

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

`I pay 555.55 in May 5555` $>$ `I pay 5555 in May 555.55`

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

Factored Decomposition



- **Problem:** Large input and output vectors
  - $|x| = 20,000,\ |y| = 50,000 \rightarrow |M| = 1,000,000,000$
- **Solution:** Use a bottleneck with smaller matrices A, B
  - $|v| = 100 \rightarrow |A| = 2,000,000,\ |B| = 5,000,000$
  - Total Parameters: $7,000,000$

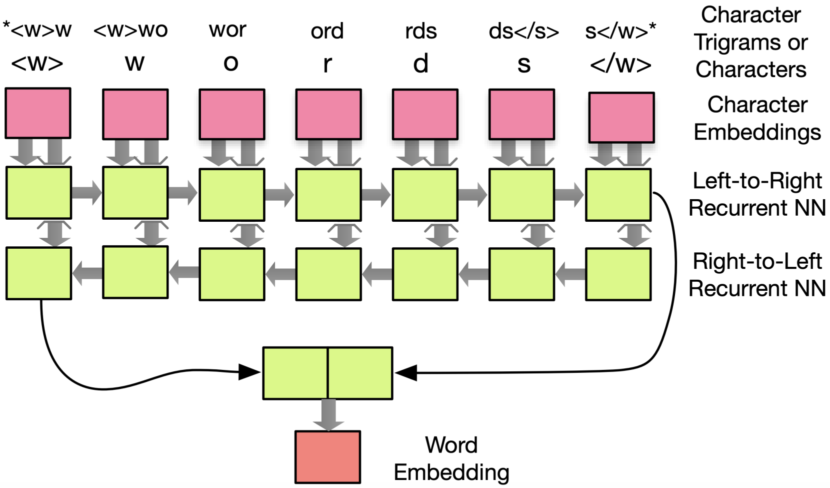
Character-Based Models

- Instead use embeddings for character string
  `b e a u t i f u l`
- Idea is to induce embeddings for unseen morphological variants:
  `beautiful`
- Tokens are single characters, symbols, whitespace

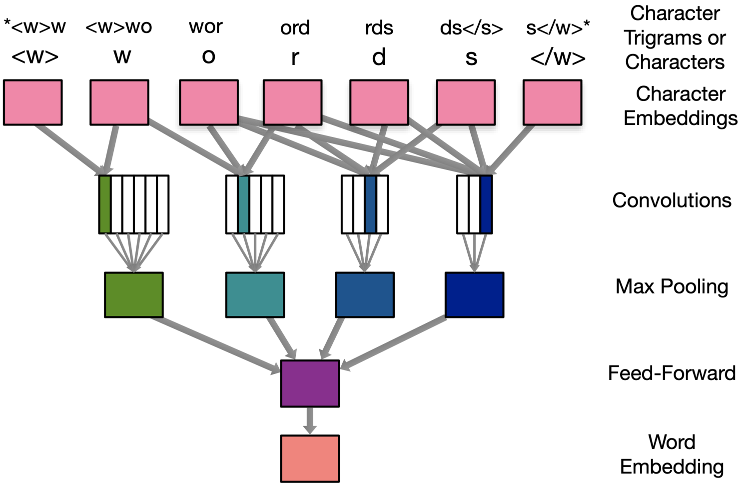
Character-Based Models

- Instead use embeddings for character string
  `b e a u t i f u l`
- Idea is to induce embeddings for unseen morphological variants:
  `beautiful`
- Tokens are single characters, symbols, whitespace
- Generally poor performance

Character-Based Models



Character-Based Models



BPE Subwords

```
            t h e · f a t · c a t · i s · i n · t h e · t h i n · b a g
t h → th    th e · f a t · c a t · i s · i n · th e · th i n · b a g
a t → at    th e · f at · c at · i s · i n · th e · th i n · b a g
i n → in    th e · f at · c at · i s · in · th e · th in · b a g
th e → the  the · f at · c at · i s · in · the · th in · b a g
```

- Breaks words into subwords
  - Starts with the character set
  - Merges the most frequent pairs, one per iteration
- Unsupervised (accidental) morphology (frequency suffixes)

BPE Tokenization on A Tale of Two Cities by Charles Dickens

- Unique Characters: 78
- Unique BPE Tokens (1K iterations): $1,071$
- Unique BPE Tokens (5K iterations): $4,864$
- Unique Tokens: $10,153$

```
it was the b@@ est of times ,
it was the wor@@ st of times ,
it was the age of w@@ is@@ do@@ m ,
it was the age of foo@@ li@@ sh@@ ness ,
it was the e@@ po@@ ch of beli@@ e@@ f ,
it was the e@@ po@@ ch of in@@ cre@@ du@@ l@@ ity ,
it was the s@@ ea@@ son of light ,
it was the s@@ ea@@ son of dar@@ k@@ ness ,
it was the sp@@ r@@ ing of hope ,
it was the win@@ ter of des@@ pa@@ ir
```

Modeling Recurrent Relations

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 $\mathbf{0}$

Long Short Term Memory (LSTM)

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

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

- 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)

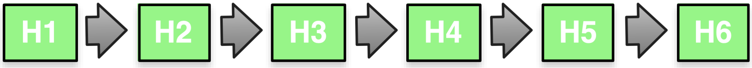
$$
\text{Gates} \rightarrow \begin{cases} r_t = \sigma(W_{ir} x_t + b_{ir} + W_{hr} h_{(t-1)} + b_{hr}) \\ z_t = \sigma(W_{iz} x_t + b_{iz} + W_{hz} h_{(t-1)} + b_{hz}) \\ n_t = \tanh(W_{in} x_t + b_{in} + r_t \odot (W_{hn} h_{(t-1)} + b_{hn})) \end{cases}
$$

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

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

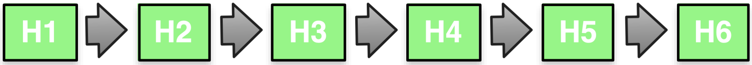
Aside: Different Perspectives on Deep Recurrent Models

- So far we've only seen Left to Right Sequencing

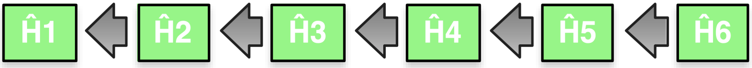


Aside: Different Perspectives on Deep Recurrent Models

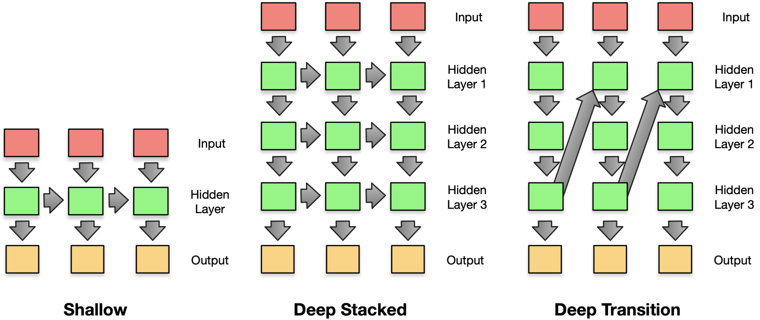
- So far we've only seen Left to Right Sequencing



- Why not Right to Left?

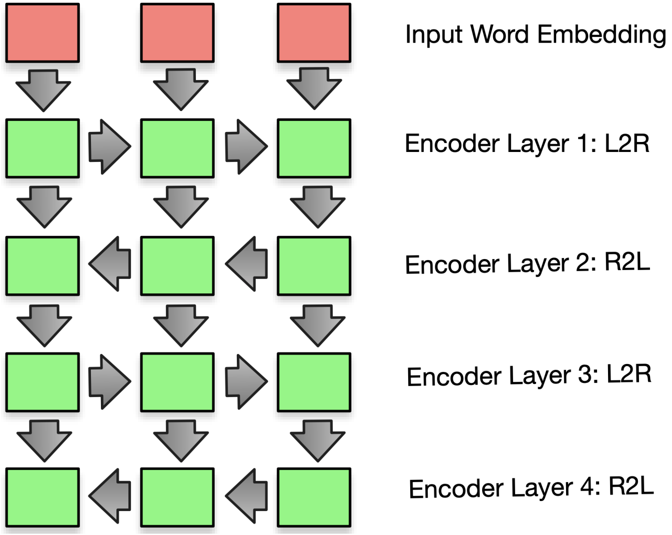


Aside: Different Perspectives on Deep Recurrent Models

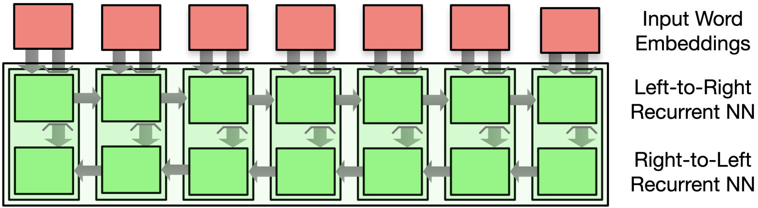


- Experiment with different stacking techniques

Alternating Recurrent Directions



Bidirectional Sequence Modeling



- Can capture both left and right context
- Implementation usually concatenates RNN states

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} c_{t-1}	$B, N_L \cdot N_D, H_{out}$ -	$B, N_L \cdot N_D, H_{out}$ $B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$ -
h_t c_t	$B, N_L \cdot N_D, H_{out}$ -	$B, N_L \cdot N_D, H_{out}$ $B, N_L \cdot N_D, H_{out}$	$B, N_L \cdot N_D, H_{out}$ -
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

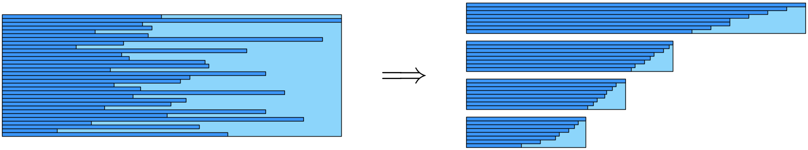
- Assume the embedding $E\left[\langle \texttt{PAD} \rangle\right] = \mathbf{0}$
- Are we safe?

Aside: The Influence of Padding in RNNs

- Assume the embedding $E[\langle \mathsf{PAD} \rangle] = \mathbf{0}$
- Are we safe? **NO!**
- Because of the bias term, zero input does not result in a zero output
- This alters the hidden state being passed onto the next iteration
- **Question:** Does this mean we learn the amount of padding for a given sequence?

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

Teacher Forcing

- Instead of refeeding the predicted token, replace it with the true token randomly
- This is only done during training, not inference

$$y_{i+1} = \begin{cases} \text{argmax}_j \, \theta_i & \mathcal{U}(0,1) < \text{TF} \\ t_{i+1} & \textit{else} \end{cases}$$

- t_{i+1} is the true token
- TF is the teacher forcing ratio

Cross Entropy and Label Smoothing

$$\ell(\mathbf{x}_i, y_i) = -\underbrace{x_{y_i}}_{max} + \overbrace{\log \sum_j \exp x_j}^{min}$$

- 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

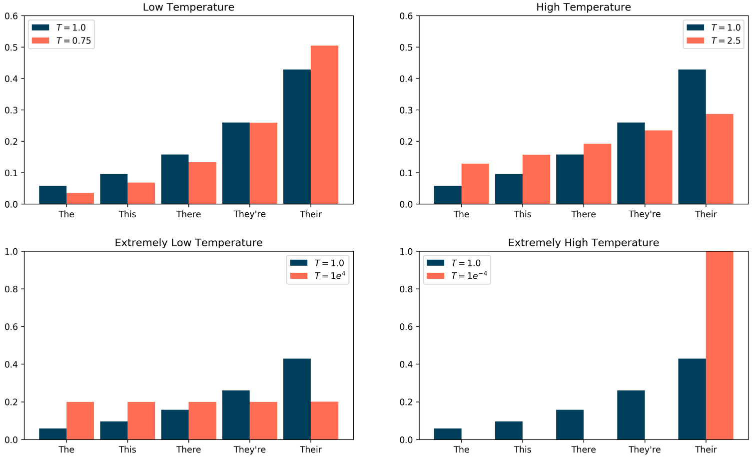
$$p(y_i) = \frac{\exp x_{y_i}}{\sum_j \exp x_j}$$

- Smoothed Softmax with Temperature T

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

- As $T \to \infty$, the distribution is smoother, uniform
- AS $T \to 0$, the distribution approaches a kronecker delta centered on the class with the most mass

Visualizing Temperature



Monte Carlo Decoding

- Recall how we select the next token:
  - **Greedy:** Top token weight
  - **Teacher Forcing:** Randomly select the true token
- Note that the outputs are a distribution over the target vocabulary
- **Use these weights in a multinomial to randomly select a continuation**

$$y_i \sim \text{Multinomial}\left(\theta_i\right)$$

Different Token Decoding Schemes

- **Greedy:**

$$y_{i+1} = \operatorname*{argmax}_j \theta_i$$

- **Teacher Forcing:**

$$y_{i+1} = \begin{cases} \operatorname{argmax}_j \theta_i & \mathcal{U}(0,1) < \text{TF} \\ t_{i+1} & \textit{else} \end{cases}$$

- **Monte Carlo:**

$$y_{i+1} \sim \text{Multinomial}\left(\theta_i\right)$$

- θ_i are the output weights from the Decoder
- t_{i+1} is the true token at position $i+1$
- TF is the teacher forcing ratio

Masked Loss

- Remember, we don't care what gets predicted after seeing a $\langle \texttt{EOS} \rangle$
- Hence, we need to mask out the loss for predicted tokens associated with $\langle \texttt{PAD} \rangle$

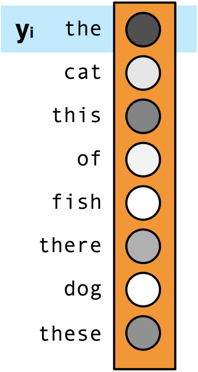
$$
\begin{array}{cccccccc}
\texttt{pred} & \rightarrow & \textit{le} & \textit{le} & \textit{chat} & \textit{chat} & \textit{chat} & \textit{chat} \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 & & \ell & \ell & \ell & \ell & \ell & 0 \\
 & & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\texttt{true} & \rightarrow & \textit{le} & \textit{chat} & \textit{est} & \textit{noir} & \langle \texttt{EOS} \rangle & \langle \texttt{PAD} \rangle
\end{array}
$$

- **Solution:** Zero out elements by either:
  - Multiply pad outputs by 0
  - Specify the label to ignore in Cross Entropy call

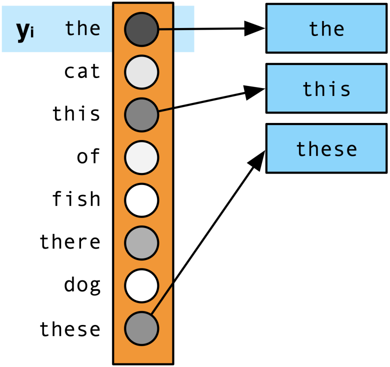
$$\ell(\mathbf{x}_i, y_i) = \mathbb{1}_{\{y_i \neq \langle \texttt{PAD} \rangle\}} \cdot \left(-x_{y_i} + \log \sum_j \exp x_j \right)$$

Decoding: Making better Translations

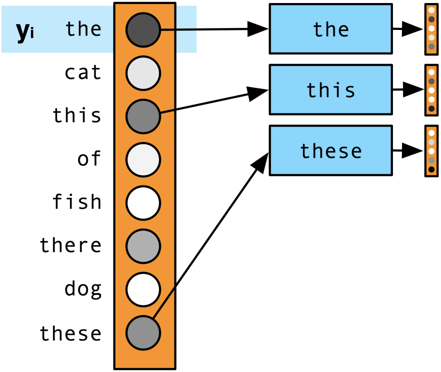
Beam Search



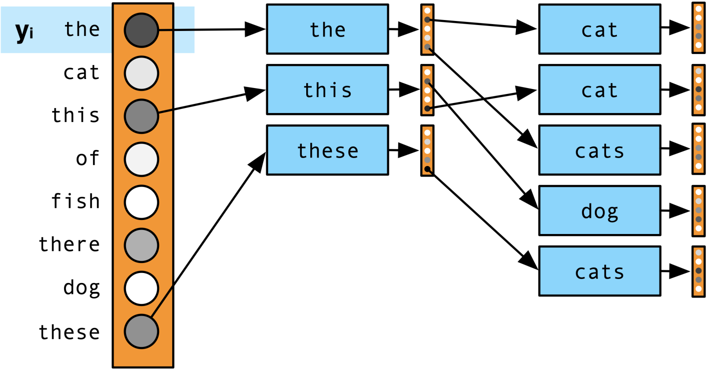
Beam Search



Beam Search

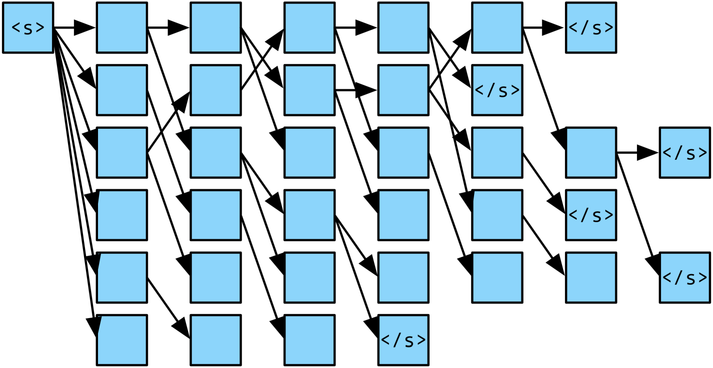


Beam Search

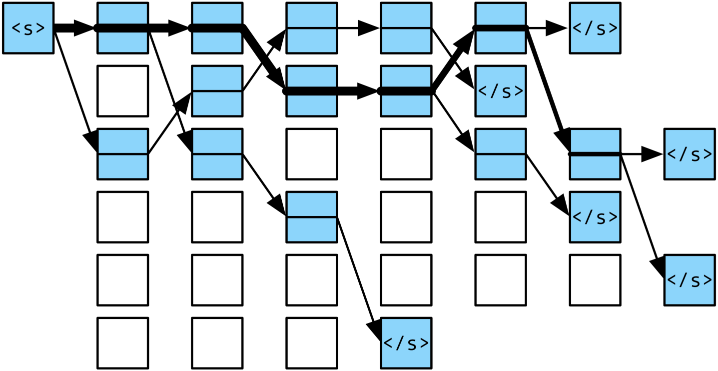


y_i
the
cat
this
of
fish
there
dog
these
the
this
these
cat
cat
cats
dog
cats

Beam Search



Beam Search

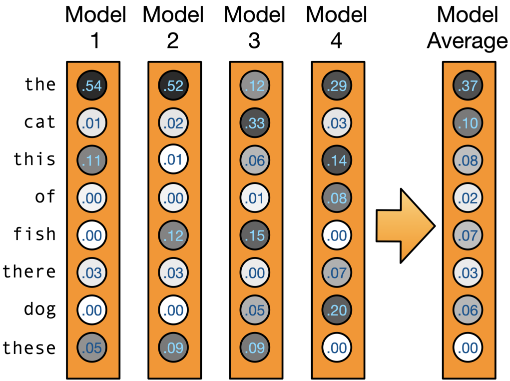


Monte Carlo Beam Search

$$y_i \sim \text{Multinomial}\left(\begin{array}{r|c|}\texttt{the} & .54 \\ \texttt{cat} & .01 \\ \texttt{this} & .11 \\ \texttt{of} & .00 \\ \texttt{fish} & .00 \\ \texttt{there} & .03 \\ \texttt{dog} & .00 \\ \texttt{these} & .05\end{array}\right)$$

- Why not sample *n* words based on their probabilities?
- Adds more diversity to beam search results

Ensembling



- Why not average different models?
- Random initialization leads to different local solutions
- Could also use model dumps from different iterations

Applications

Recall: Encoders, Decoders, and Seq2Seq Models

- **Encoders** given a sequence meaning
- **Decoders** generate a new sequence
- **Seq2Seq** generate sequences conditioned on another sequence

What to use for which application?

- **Encoders**
  - POS Tagging
  - Sentence Embeddings
  - Anything where you are given the sentence at test time
- **Decoders**
  - Text Generation
  - Language Modeling
  - Anything where you need to create a sequence at test time
- **Seq2Seq**
  - Translation
  - Speech Recognition
  - Summarization
  - Question/Answering
  - Anything where you convert a sequence into another sequence

Tools, References, and Further Reading

Papers

- Sutskever et al., Sequence to Sequence Learning with Neural Networks
- Cho et al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
- Sennrich et al., Neural Machine Translation of Rare Words with Subword Units
- Koehn, Neural Machine Translation
- Koehn, Six Challenges for Neural Machine Translation

Tutorials

- Pytorch
  - Official PyTorch Seq2Seq Tutorial
  - PyTorch Seq2Seq with Torchtext
  - Ben Trevett Seq2Seq Tutorial
- Tensorflow
  - NMT with Attention

Libraries

- Facebook: fairseq (PyTorch)
- Open NMT (PyTorch)
- Open NMT (Tensorflow)

Thank You!