Neural Machine Translation

Philipp Koehn

6 October 2020
Language Models

- Modeling variants
  - feed-forward neural network
  - recurrent neural network
  - long short term memory neural network

- May include input context
Feed Forward Neural Language Model

Wi

h

Ew

Softmax

FF

Output Word

Hidden Layer

Embedding

History

Wi

Wi

Wi

Wi-4

Wi-3

Wi-2

Wi-1
Recurrent Neural Language Model

\[ y_i \] → \text{the} → Output Word
\[ t_i \] → \text{Softmax} → Output Word Prediction
\[ h_j \] → \text{RNN} → Recurrent State
\[ E \ x_j \] → \text{Embed} → Input Word Embedding
\[ x_j \] → \text{<s>} → Input Word

Predict the first word of a sentence
Recurrent Neural Language Model

Predict the second word of a sentence
Re-use hidden state from first word prediction
Recurrent Neural Language Model

Predict the third word of a sentence
... and so on
Recurrent Neural Language Model

The diagram illustrates a recurrent neural network (RNN) model for language processing. The model takes an input word embedding and processes it through a series of RNN layers, each of which computes a hidden state. The output of the RNN is then passed through a softmax layer to predict the next word in the sequence. The process continues until the end of the sentence is reached, indicated by the end-of-sentence symbol. The model uses both input and recurrent states to generate the output sequence of words.
Recurrent Neural Translation Model

- We predicted the words of a sentence

- Why not also predict their translations?
• Obviously madness

• Proposed by Google (Sutskever et al. 2014)
What is Missing?

- Alignment of input words to output words

⇒ Solution: attention mechanism
neural translation model with attention
• Inspiration: recurrent neural network language model on the input side
Hidden Language Model States

• This gives us the hidden states

- RNN - RNN - RNN - RNN - RNN - RNN - RNN

• These encode left context for each word

• Same process in reverse: right context for each word

- RNN - RNN - RNN - RNN - RNN - RNN - RNN
Input encoder: concatenate bidirectional RNN states

Each word representation includes full left and right sentence context
Input is sequence of words $x_j$, mapped into embedding space $\bar{E} x_j$

Bidirectional recurrent neural networks

\[
\hat{h}_j = f(\hat{h}_{j+1}, \bar{E} x_j) \\
\overrightarrow{h}_j = f(\overrightarrow{h}_{j-1}, \bar{E} x_j)
\]

Various choices for the function $f()$: feed-forward layer, GRU, LSTM, ...
We want to have a recurrent neural network predicting output words.
• We want to have a recurrent neural network predicting output words

• We feed decisions on output words back into the decoder state
Decoder

- We want to have a recurrent neural network predicting output words

- We feed decisions on output words back into the decoder state
- Decoder state is also informed by the input context
• Decoder is also recurrent neural network over sequence of hidden states $s_i$

$$s_i = f(s_{i-1}, E y_{i-1}, c_i)$$

• Again, various choices for the function $f()$: feed-forward layer, GRU, LSTM, ...

• Output word $y_i$ is selected by computing a vector $t_i$ (same size as vocabulary)

$$t_i = W(U s_{i-1} + V E y_{i-1} + C c_i)$$

then finding the highest value in vector $t_i$

• If we normalize $t_i$, we can view it as a probability distribution over words

• $E y_i$ is the embedding of the output word $y_i$
• Given what we have generated so far (decoder hidden state)
• ... which words in the input should we pay attention to (encoder states)?
• Given: – the previous hidden state of the decoder $s_{i-1}$
  – the representation of input words $h_j = (\overleftarrow{h_j}, \overrightarrow{h_j})$

• Predict an alignment probability $a(s_{i-1}, h_j)$ to each input word $j$
  (modeled with with a feed-forward neural network layer)
• Normalize attention (softmax)

\[ \alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_k \exp(a(s_{i-1}, h_k))} \]
• Relevant input context: weigh input words according to attention: $c_i = \sum_j \alpha_{ij} h_j$
• Use context to predict next hidden state and output word
training
Comparing Prediction to Correct Word

- Current model gives some probability $t_i[y_i]$ to correct word $y_i$
- We turn this into an error by computing cross-entropy: $-\log t_i[y_i]$
• Math behind neural machine translation defines a computation graph
• Forward and backward computation to compute gradients for model training
Unrolled Computation Graph

E yi → Embed
yi → <s> das Haus ist groß ...
- log ti [yi] → Cost Softmax

RNN
Si → RNN
C_i → Weighted Sum
α_ij → Attention

h_{ij} → RNN
\bar{h}_{ij} → RNN

E x_j → Embed
x_j → <s> the house is big ...

Output Word Embeddings
Output Word
Error
Output Word Prediction
Decoder State
Input Context
Attention
Right-to-Left Encoder
Left-to-Right Encoder
Input Word Embedding
Input Word

Philipp Koehn  Machine Translation: Neural Machine Translation 6 October 2020
Batching

• Already large degree of parallelism
  – most computations on vectors, matrices
  – efficient implementations for CPU and GPU

• Further parallelism by batching
  – processing several sentence pairs at once
    – scalar operation \(\rightarrow\) vector operation
    – vector operation \(\rightarrow\) matrix operation
    – matrix operation \(\rightarrow\) 3d tensor operation

• Typical batch sizes 50–100 sentence pairs
Batches

- Sentences have different length
- When batching, fill up unneeded cells in tensors

⇒ A lot of wasted computations
Mini-Batches

• Sort sentences by length, break up into mini-batches

• Example: Maxi-batch 1600 sentence pairs, mini-batch 80 sentence pairs
Overall Organization of Training

- Shuffle corpus
- Break into maxi-batches
- Break up each maxi-batch into mini-batches
- Process mini-batch, update parameters
- Once done, repeat
- Typically 5-15 epochs needed (passes through entire training corpus)
deeper models
Deeper Models

- Encoder and decoder are recurrent neural networks
- We can add additional layers for each step
- Recall shallow and deep language models

- Adding residual connections (short-cuts through deep layers) help
Deep Decoder

- Two ways of adding layers
  - deep transitions: several layers on path to output
  - deeply stacking recurrent neural networks

- Why not both?
• Previously proposed encoder already has 2 layers
  – left-to-right recurrent network, to encode left context
  – right-to-left recurrent network, to encode right context

⇒ Third way of adding layers