CS-568 Deep Learning

Nazar Khan

PUCIT

Language Modelling

Outline

1. Modelling input text as numeric vectors
2. Text generation
3. Language translation

Modelling text as numeric vectors

- *Corpus*: Consider a dataset of news articles.
- *Vocabulary*: Set V^{in} of (all or most frequent) unique words in the corpus.
- Assume size of vocabulary is K^{in} words.
- Each word can be represented using 1-of-K coding.
- For example, k-th word in V can be represented as

$$\mathbf{y}_k = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where 1 appears at the k-th index.

Inefficiency of 1-hot vectors

- 1-of-K coding is
  1. *tremendously inefficient* since K^2 numbers represent K words only, and
  2. *highly unrealistic* since 1-hot vectors are orthogonal while words have similarities.

Workaround: Embedding Matrix

- Project word vectors onto lower dimensional space via *projection/embedding* matrix E.

$$\mathbf{e} = E\mathbf{y}$$

- Matrix E is of size $D \times K^{\text{in}}$ where $D << K^{\text{in}}$.
- Optimal matrix E can be learned as part of the network parameters.

Output

- Let output language have a vocabulary V^{out} of K^{out} words.
- Then output layer is softmax on K^{out} neurons.

Loss

- For a sentence of T_n words, we can use cross-entropy between output sequence and target sequence.

$$\mathcal{L}_n\left(\left(\mathbf{y}^{(1)},\mathbf{y}^{(2)},\dots,\mathbf{y}^{(T_n)}\right),\left(\mathbf{t}^{(1)},\mathbf{t}^{(2)},\dots,\mathbf{t}^{(T_n)}\right)\right) = -\sum_{t=1}^{T_n}\sum_{j=1}^{K^{\text{out}}} t_j^{(t)} \ln y_j^{(t)}$$

$$= -\sum_{t=1}^{T_n} \ln y_{\text{target}}^{(t)}$$

- Training can be performed using BPTT on a corpus (typically) containing millions of words.
- Each sentence constitutes one training example.

Text Generation

- Problem: generate a sequence of words $w^{(1)}, w^{(2)}, \ldots$.
- We will add two new words to the vocabulary.
  - sos: start of sentence
  - eos: end of sentence
- Solution:
  1. At time $t = 1$, feed $w^{(0)}$ the sos word. That is, starting vector is $\mathbf{x}^{(0)} = \mathbf{0}$.
  2. Compute probability distribution $\mathbf{y}^{(1)}$.
  3. Sample a word $w^{(1)}$ from this distribution.
     - 3.1 argmax, or
     - 3.2 random sampling based on probabilities in $\mathbf{y}^{(1)}$, or
     - 3.3 any other sampling method.
  4. At every time step $t = 1, \ldots,$ feed $w(t-1)$ as input, generate probability distribution $\mathbf{y}^{(t)}$ and sample next word $w(t)$ from it.
  5. Continue until eos is sampled.

Language Translation

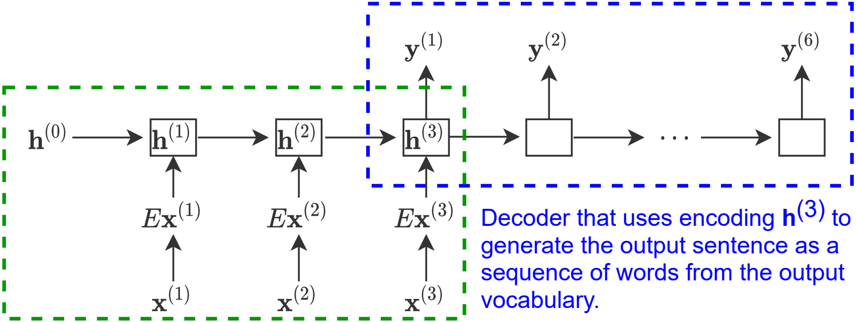
Zaid slapped Khalid ⟶ زید نے خالد کو تھپڑ مارا

$\mathbf{x}^{(1)}$ $\mathbf{x}^{(2)}$ $\mathbf{x}^{(3)}$ $\mathbf{y}^{(6)}$ $\mathbf{y}^{(5)}$ $\mathbf{y}^{(4)}$ $\mathbf{y}^{(3)}$ $\mathbf{y}^{(2)}$ $\mathbf{y}^{(1)}$

Language Translation

$$\begin{tikzcd}
\mathbf{h}^{(0)} \arrow[r] & \boxed{\mathbf{h}^{(1)}} \arrow[r] & \boxed{\mathbf{h}^{(2)}} \arrow[r] & \boxed{\mathbf{h}^{(3)}} \\
 & E\mathbf{x}^{(1)} \arrow[u] & E\mathbf{x}^{(2)} \arrow[u] & E\mathbf{x}^{(3)} \arrow[u] \\
 & \mathbf{x}^{(1)} \arrow[u] & \mathbf{x}^{(2)} \arrow[u] & \mathbf{x}^{(3)} \arrow[u]
\end{tikzcd}$$

Language Translation

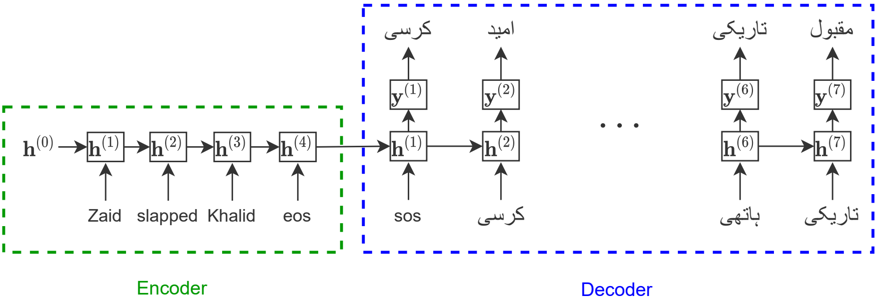


Decoder that uses encoding $\mathbf{h}^{(3)}$ to generate the output sentence as a sequence of words from the output vocabulary.

Encoder that produces $\mathbf{h}^{(3)}$ as the encoding of the whole input sequence.

Language Translation

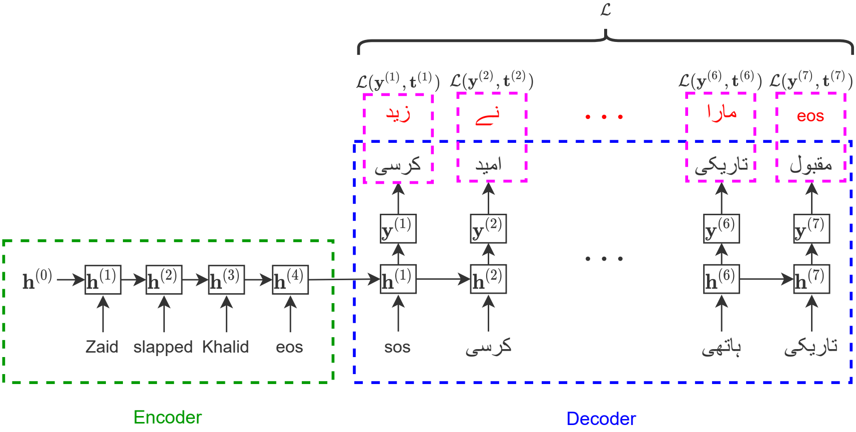
A better decoder



Make probability distribution $\mathbf{y}^{(t+1)}$ depend on *word drawn* from $\mathbf{y}^{(t)}$ as well.

$$y_j^{(t)} = P(o^{(t)} = V_j | \underbrace{o^{(t-1)}, o^{(t-2)}, \dots, o^{(1)}}_{\text{all words output so far}}, \underbrace{w^{(1)}, w^{(2)}, \dots, w^{(T_{\text{in}})}}_{\text{all input words}})$$

Language Translation
Training



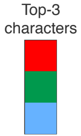
Language Translation

Testing: Finding the most likely output

- As mentioned earlier, sampling of words can be accomplished via
  1. argmax on each $\mathbf{y}^{(t)}$, or
  2. random sampling from each $\mathbf{y}^{(t)}$
- Both sampling methods produce locally optimal words.
- A better but costlier alternative is to find a globally optimal output sequence.

Beam Search

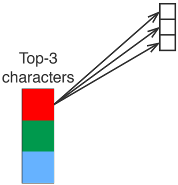
At time $t = 1$, pick the M most probable options instead of all K^{out} options.



$t = 1$

Beam Search

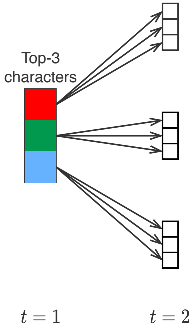
Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



$t = 1$ $t = 2$

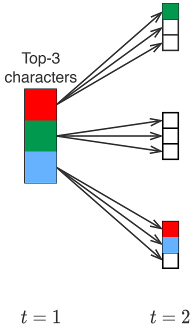
Beam Search

Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



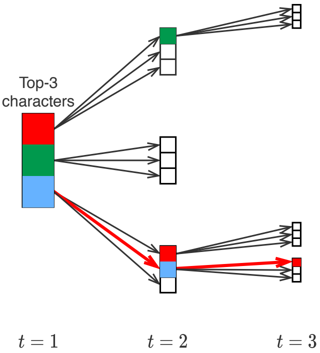
Beam Search

Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



Beam Search

Conditioned on each *path* at $t = 2$, pick the M most probable options at $t = 3$.



Beam Search

- A sequence is terminated when eos is drawn.
- When no unterminated sequence remains, select the most likely sequence across all terminating sequences.