INFO251 – Applied Machine Learning

Lab 12

Suraj R. Nair

Topics

- Summary: RNNs and Transformers
- Embeddings
- Transformers:
  - Positional encoding

Encoder Decoder Architecture

- RNNs and the context bottleneck

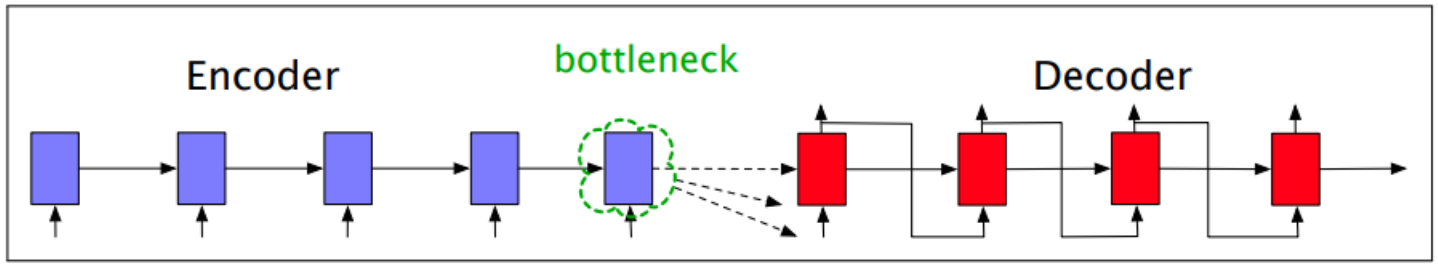


Figure 9.20 Requiring the context c to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

Jurafsky and Martin, Speech and Language Processing (3 ed. draft)

Attention

- The attention mechanism:
  - a solution to the bottleneck problem
  - allows the decoder to get information from all the hidden states of the encoder
  - Idea:
    - create single fixed-length vector (weighted sum of all the encoder hidden states)
    - generate this vector at each decoding step
    - relevance of an encoder state to a decoder state: dot-product attention (relevance = similarity)

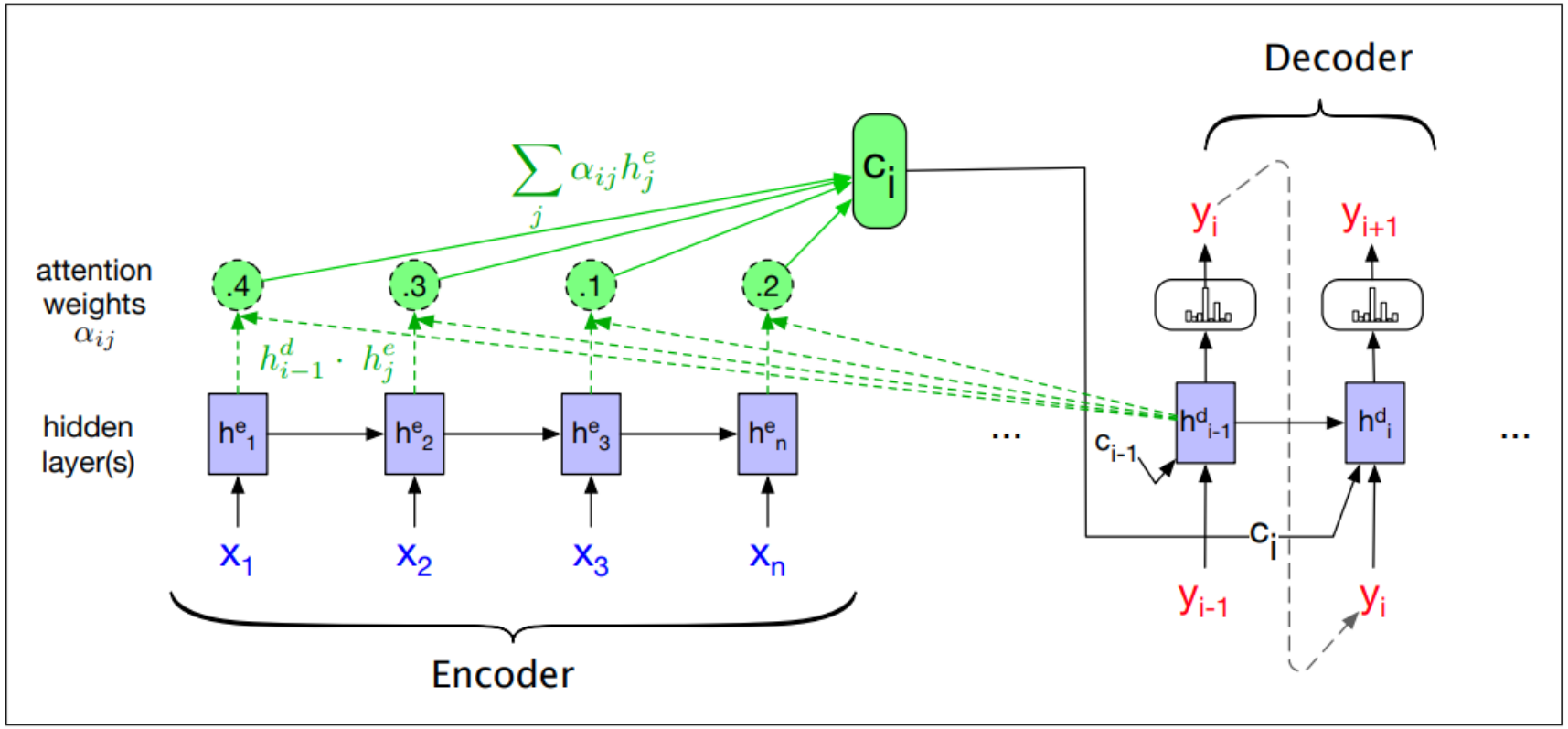
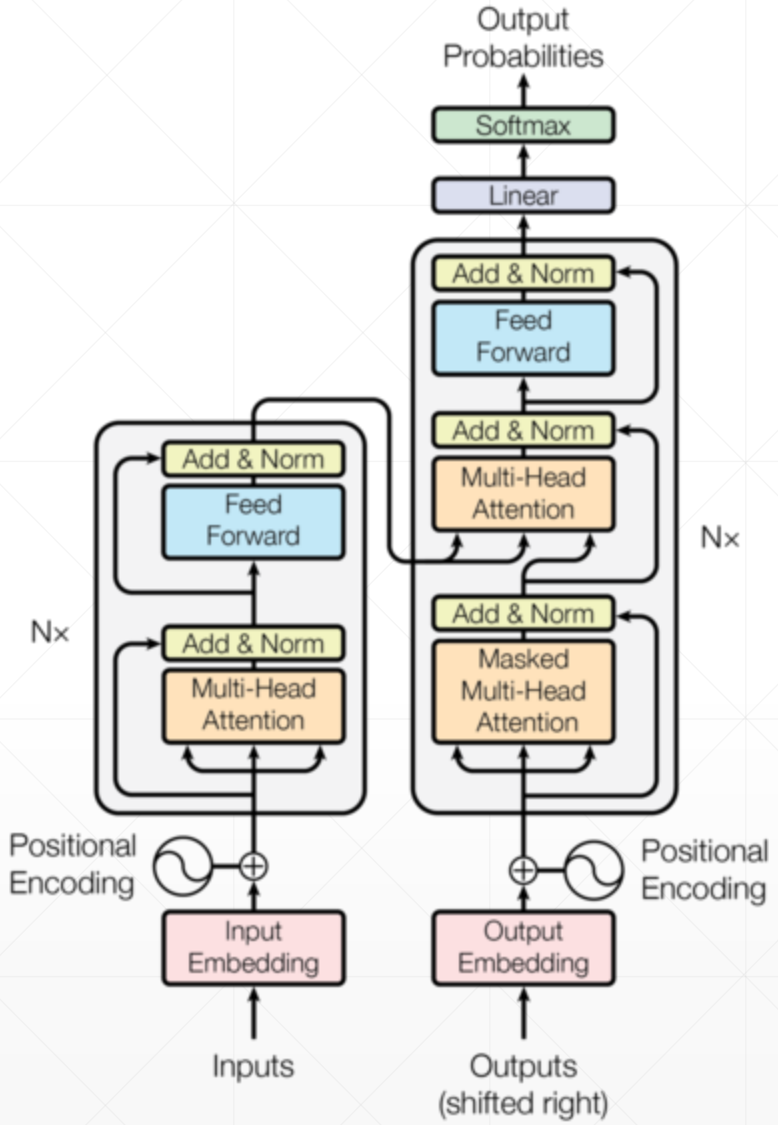


Figure 9.22 A sketch of the encoder-decoder network with attention, focusing on the computation of $\mathbf{c}_i$. The context value $\mathbf{c}_i$ is one of the inputs to the computation of $\mathbf{h}^d_i$. It is computed by taking the weighted sum of all the encoder hidden states, each weighted by their dot product with the prior decoder hidden state $\mathbf{h}^d_{i-1}$.

Transformers

- Attention is all you need (Vaswani et al. 2017)
- The annotated transformer (line by line implementation)
- Key concepts
  - Learns its own word embeddings
  - Positional encoding
  - Self-attention / cross-attention



RNNs vs Transformers

- RNNs
  - Sequential dependencies
  - Challenging to capture longer-range dependence (LSTMs address this, somewhat)
  - Difficult to parallelize

RNNs v/s Transformers

- Transformers:
  - Parallel processing
  - Self-attention → long-range dependencies
  - Large amounts of data needed for training
  - Attention mechanism: some level of interpretability

Positional Encoding

"Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension ***d*** as the embeddings, so that the two can be summed."

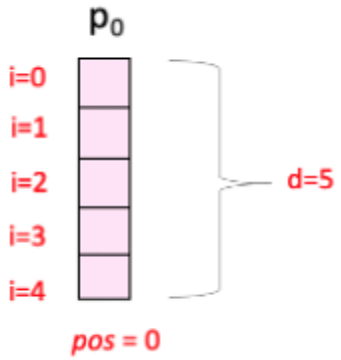
Vaswani et al (2017)

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

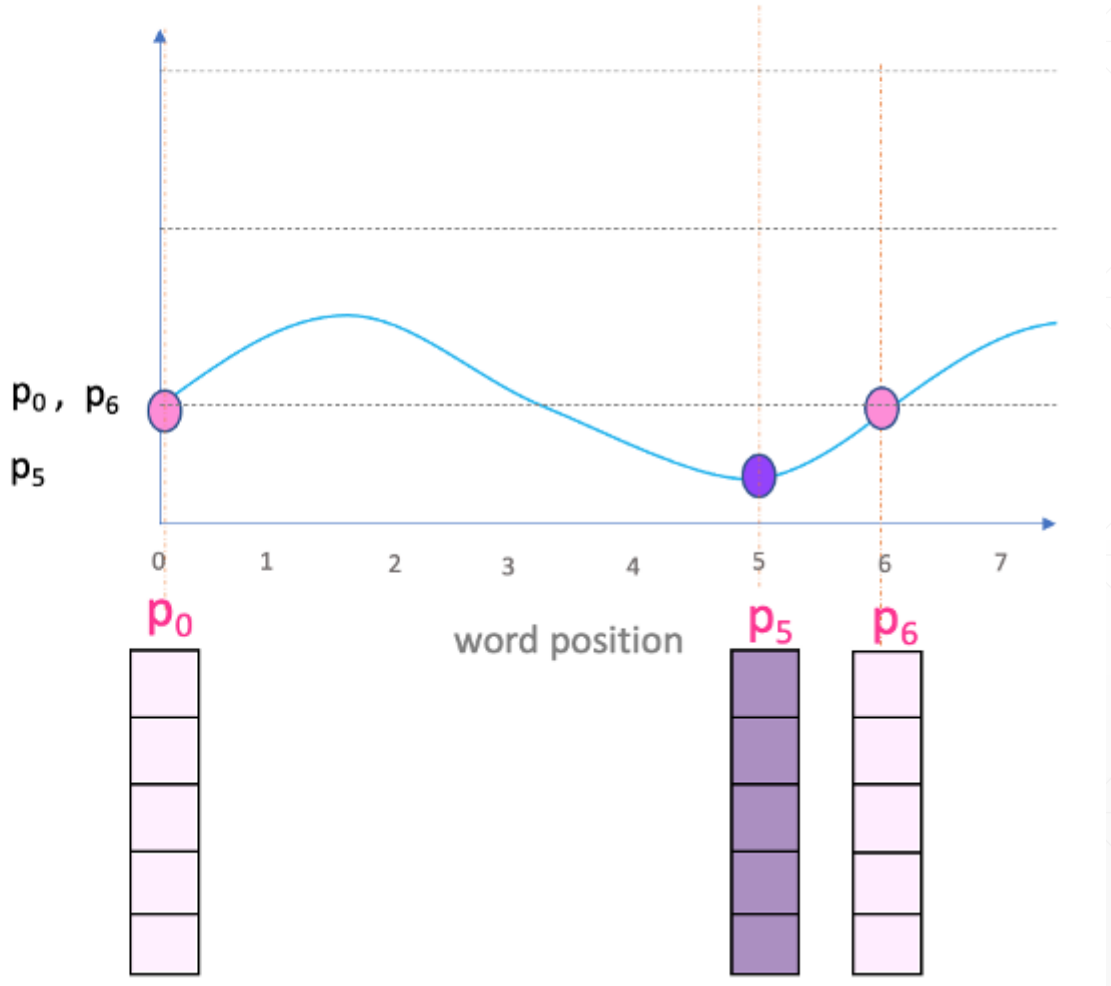
Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$



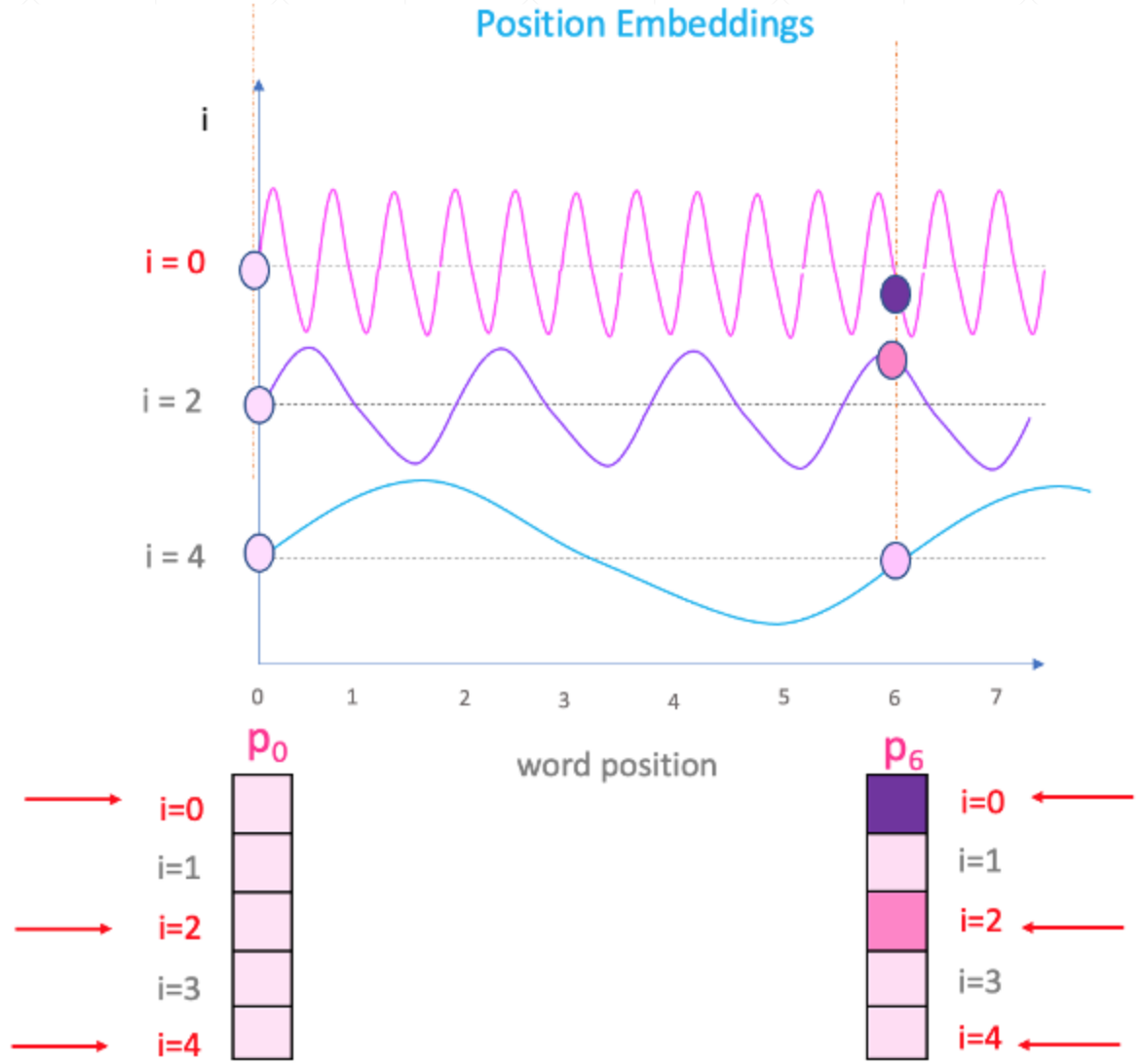
Source

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$



Source

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

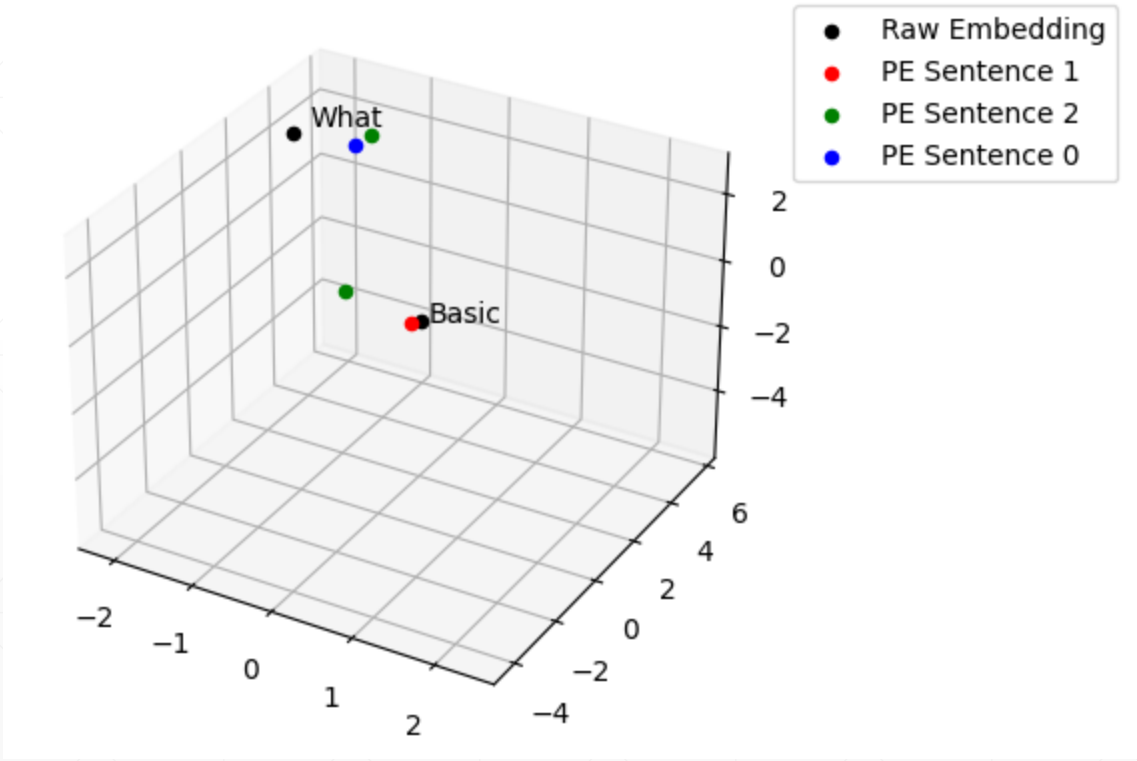


Source

Toy Example

- Vocabulary: "Hello! This is an example of a paragraph that has been split into its basic components. I wonder what will come next! Any guesses?“
- New Sequences:
  - "I wonder what will come next!“
  - "This is a basic example paragraph."
  - "Hello, what is a basic split?"

Embeddings


Raw Embedding
PE Sentence 1
PE Sentence 2
PE Sentence 0
What
Basic
2
0
−2
−4
6
4
2
0
−2
−4
−2
−1
0
1
2