

Dive into Deep Learning for NLP

5. Contextual Representations

Haibin Lin

gluon-nlp.mxnet.io

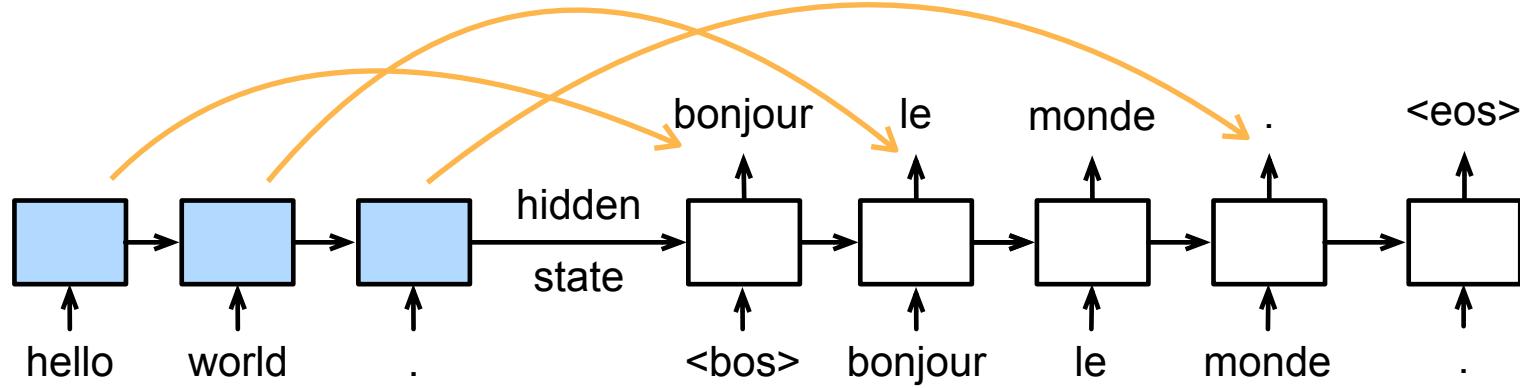
13:15-14:15	Natural Language Processing and Deep Learning Basics
14:15-14:25	Break
14:25-15:15	Context-free Representations with Word Embeddings
15:15-15:55	Machine Translation and Sequence Generation
15:55-16:35	Contextual Representations with BERT
16:35-16:45	Break
16:45-17:15	Model Deployment with TVM

Attention



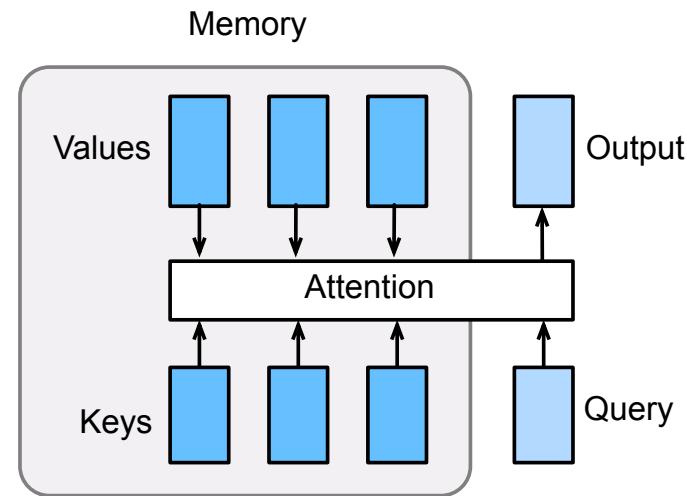
Motivation

- Each generated token might be related to different source tokens



Attention Layer

- An attention layer explicitly select related information
 - Its memory consists of key-value pairs
 - The output will be close to the values whose keys are similar to the query



Attention Layer

- Assume query $\mathbf{q} \in \mathbb{R}^{d_q}$ and the memory $(\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_n, \mathbf{v}_n)$ with $\mathbf{k}_i \in \mathbb{R}^{d_k}$ $\mathbf{v}_i \in \mathbb{R}^{d_v}$
- Compute n scores a_1, \dots, a_n with $a_i = \alpha(\mathbf{q}, \mathbf{k}_i)$
- Use softmax to obtain the attention weights

$$b_1, \dots, b_n = \text{softmax}(a_1, \dots, a_n)$$

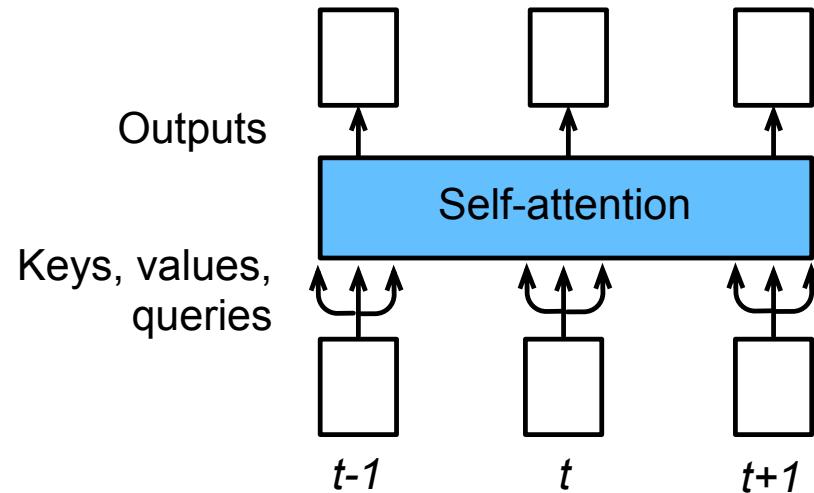
- The output is a weighted sum of values

$$\mathbf{o} = \sum_{i=1}^n b_i \mathbf{v}_i$$

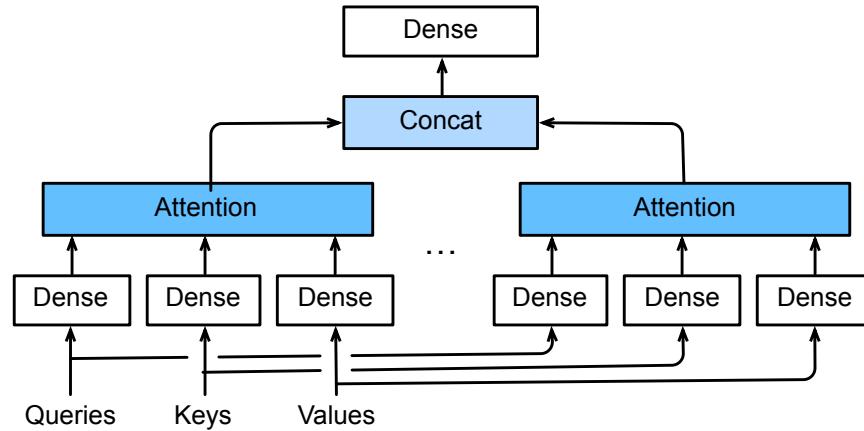
Vary α to get different attention layers

Self-attention

- To generate n outputs with n inputs, we can copy each input into a key, a value and a query
- No sequential information is preserved
- Run in parallel



Multi-head Attention



$\mathbf{W}_q^{(i)} \in \mathbb{R}^{p_q \times d_q}$, $\mathbf{W}_k^{(i)} \in \mathbb{R}^{p_k \times d_k}$, and $\mathbf{W}_v^{(i)} \in \mathbb{R}^{p_v \times d_v}$

$\mathbf{o}^{(i)} = \text{attention}(\mathbf{W}_q^{(i)}\mathbf{q}, \mathbf{W}_k^{(i)}\mathbf{k}, \mathbf{W}_v^{(i)}\mathbf{v}) \quad \text{for } i = 1, \dots, h$

$$\mathbf{o} = \mathbf{W}_o \begin{bmatrix} \mathbf{o}^{(1)} \\ \vdots \\ \mathbf{o}^{(h)} \end{bmatrix}$$



Positional Encoding

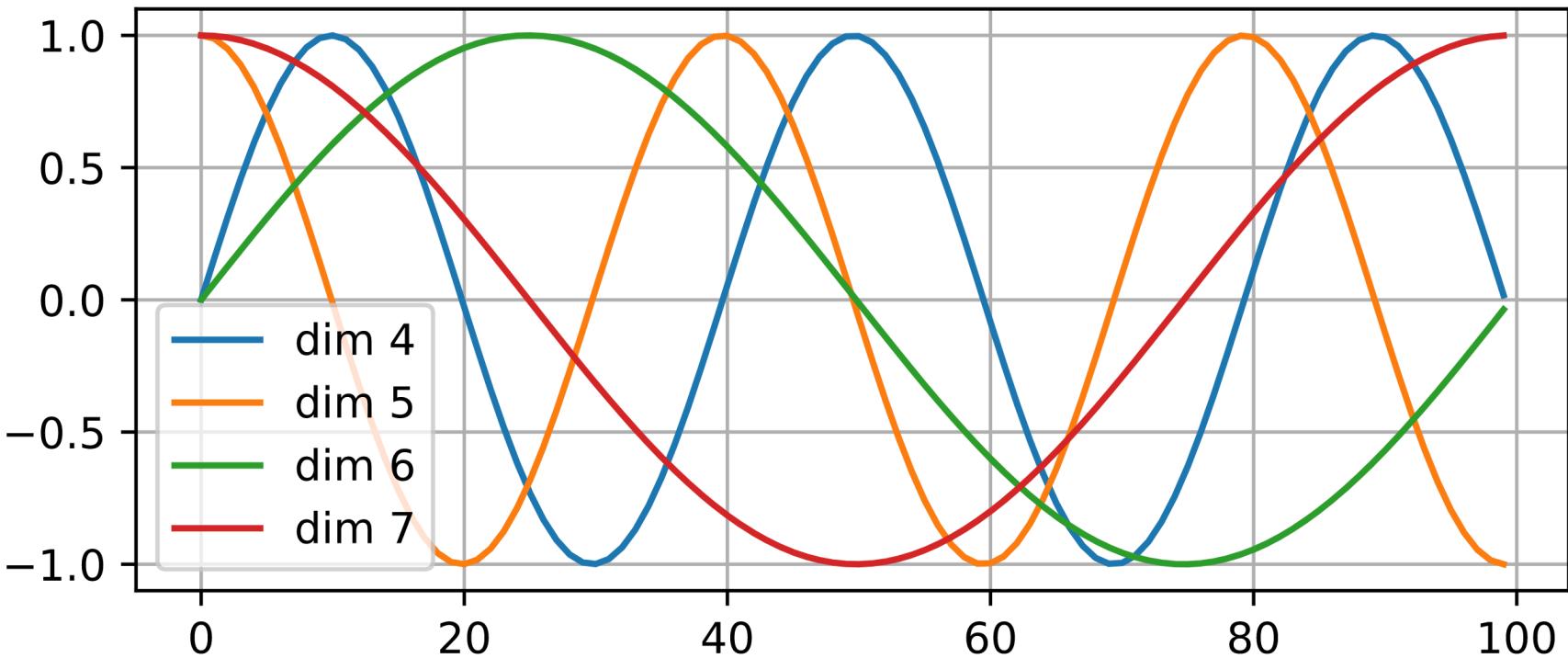
- Assume embedding output $X \in \mathbb{R}^{l \times d}$ with shape (seq len, embed dim)
- Create $P \in \mathbb{R}^{l \times d}$ with

$$P_{i,2j} = \sin(i/10000^{2j/d})$$

$$P_{i,2j+1} = \cos(i/10000^{2jd})$$

- Output $X + P$

Positional Encoding

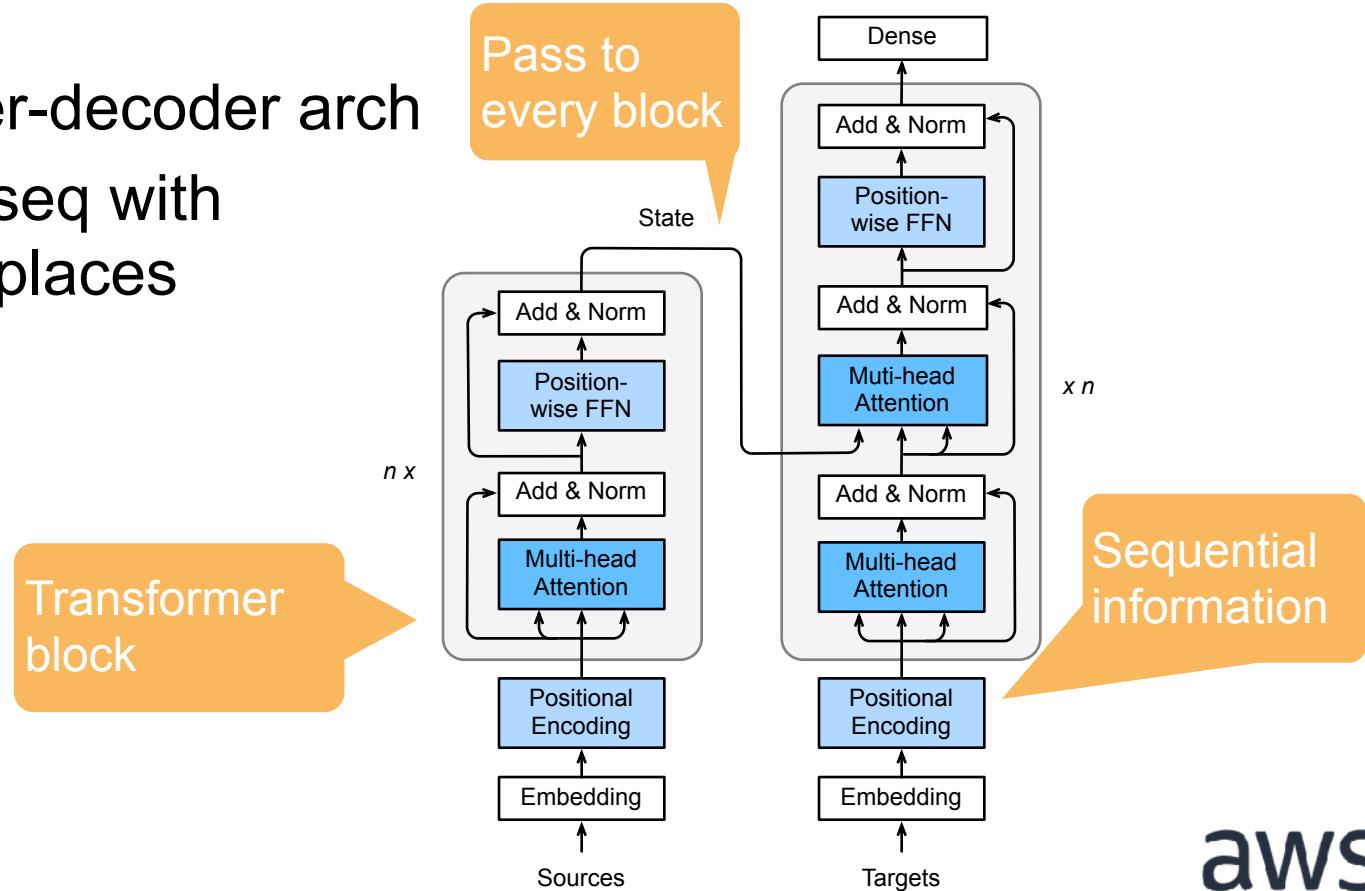


Transformer



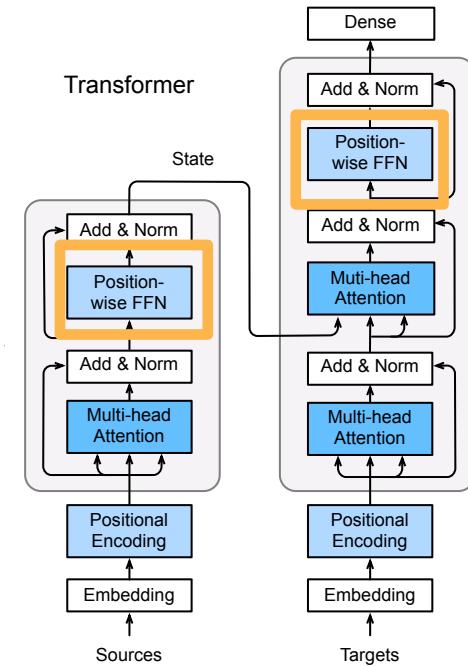
Transformer Architecture

- It's an encoder-decoder arch
- Differ to seq2seq with attention in 3 places



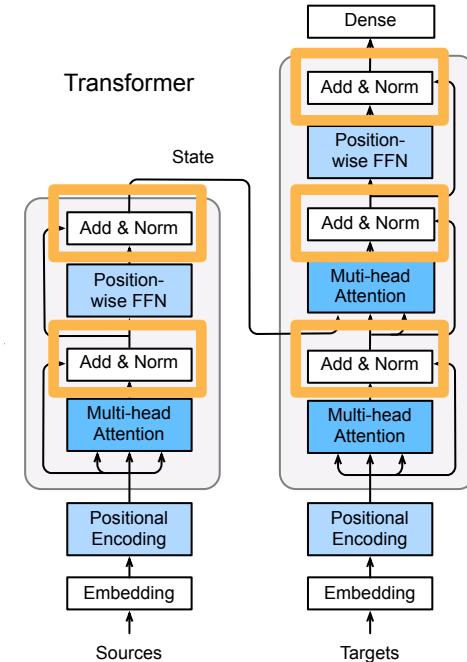
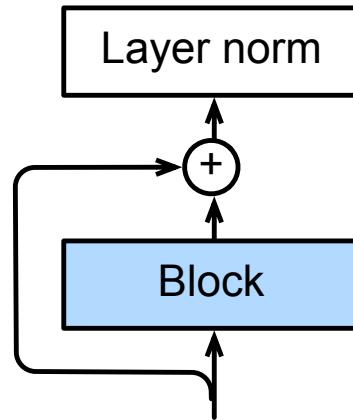
Position-wise Feed-Forward Networks

- Reshape input (batch, seq len, fea size) into (batch * seq len, fea size)
- Apply a two layer MLP
- Reshape back into 3-D
- Equals to apply two (1,1) conv layers



Add and Norm

- Layer norm is similar to batch norm
- But the mean and variances are calculated along the last dimension
- `X.mean(axis=-1)` instead of the first batch dimension in batch norm `X.mean(axis=0)`



**Word2Vec/FastText/GloVE
is great. Can we do better?**

BERT

Bidirectional Embedding from
Transformers

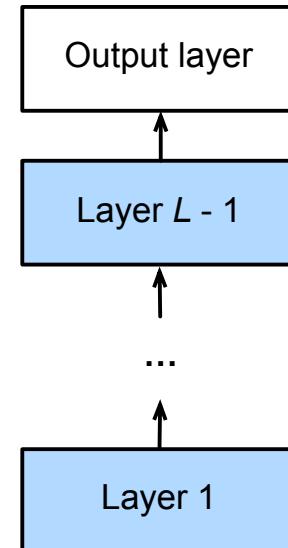


Motivation of BERT

- A fine-tuning based approach
- The pre-trained model capture sufficient data information
- Only need to add a simple output layer for a new task

NLP

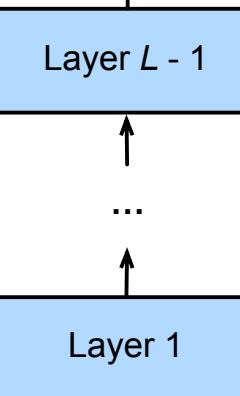
Positive



CV



Classifier



Feature
extractor

I love this movie

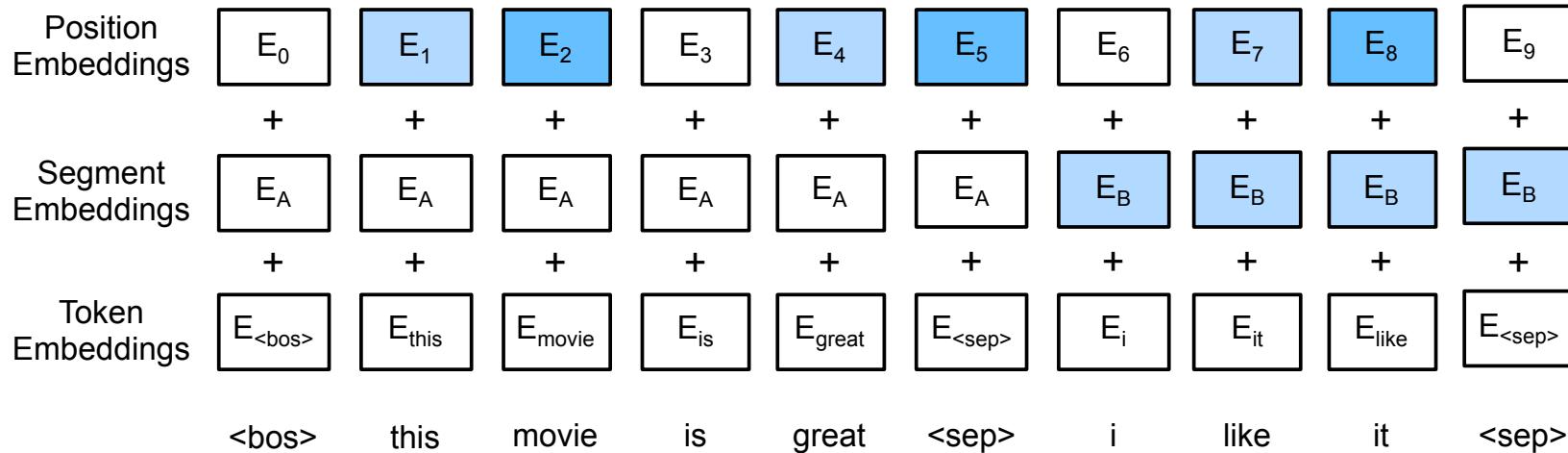


BERT Architecture

- A (big) Transformer encoder (without the decoder)
- Two variants:
 - Base: #blocks = 12, hidden size = 768, #heads = 12, #parameters = 110M
 - Large: #blocks = 24, hidden size = 1024, #heads = 16, #parameters = 340M
- Train on large-scale corpus (books and wikipedia) with > 3B words

Modification of inputs

- Each example is a pair of sentences
- Add an additional segment embedding



Pre-training Task 1: Masked Language Model

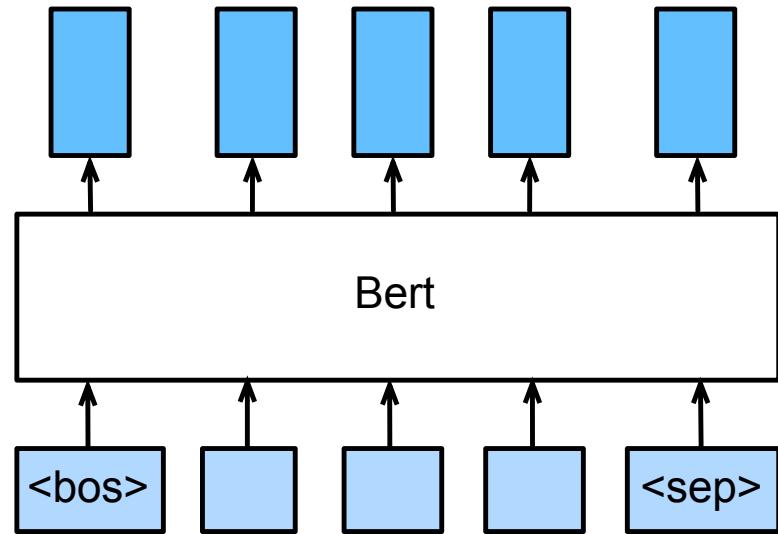
- Randomly mask (e.g. 15%) tokens in each sentence, predict these masked tokens
 - Transformer is bidirectional, which breaks the unidirectional limit of standard LM
- No mask token (<mask>) in fine turning tasks
 - 80% of the time, replace selected tokens with <mask>
 - 10% of the time, replace with randomly picked tokens
 - 10% of the time, keep the original tokens

Pre-training Task 2: Next Sentence Prediction

- 50% of time, choose a sequential sentence pair
 - <bos> this movie is great <sep> i like it <sep>
- 50% of time, choose a random sentence pair
 - <bos> this movie is great <sep> hello world <sep>
- Feed the Transformer output of <bos> into a dense layer to predict if it is a sequential pair

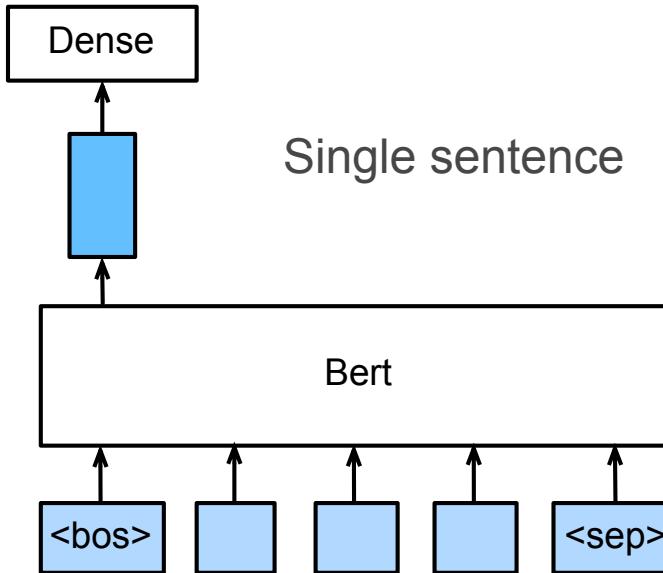
BERT for Fine Tuning

- Bert returns a feature vector for each token that captures the context information
- Different fine-tuning tasks use a different set of vectors

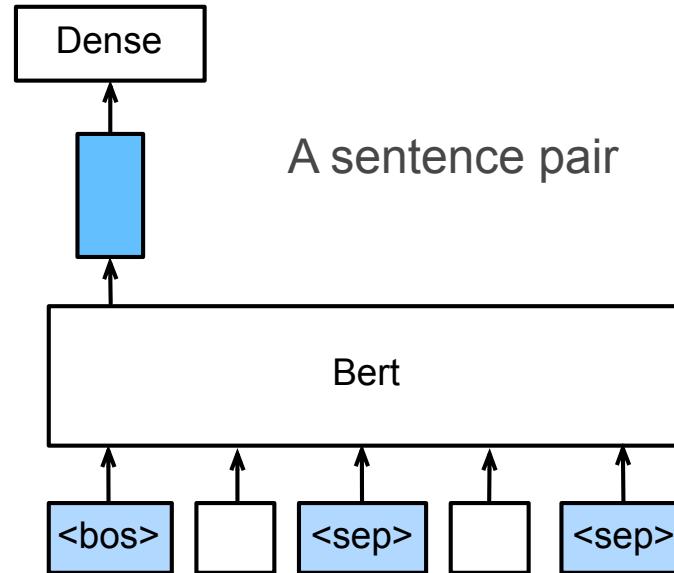


Sentences Classification

- Feed the <bos> token vector into a dense output layer



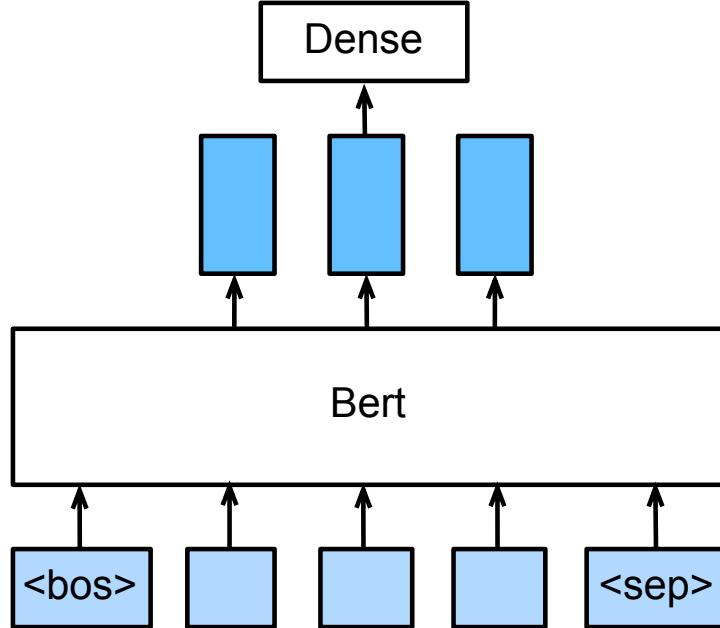
Single sentence



A sentence pair

Named Entity Recognition

- Identify if a token is a named entity such as person, org, and locations...
- Feed each non-special token vector into a dense output layer



Question Answering

- Given a question and a description text, find the answer, which is a text segment in the description
- Given p_i the i -th token in the desperation, learn s so that

$$p_1, \dots, p_T = \text{softmax}(\langle s, v_1 \rangle, \dots, \langle s, v_T \rangle)$$

p_i is the probability i -th token is the segment start. Same for the end

