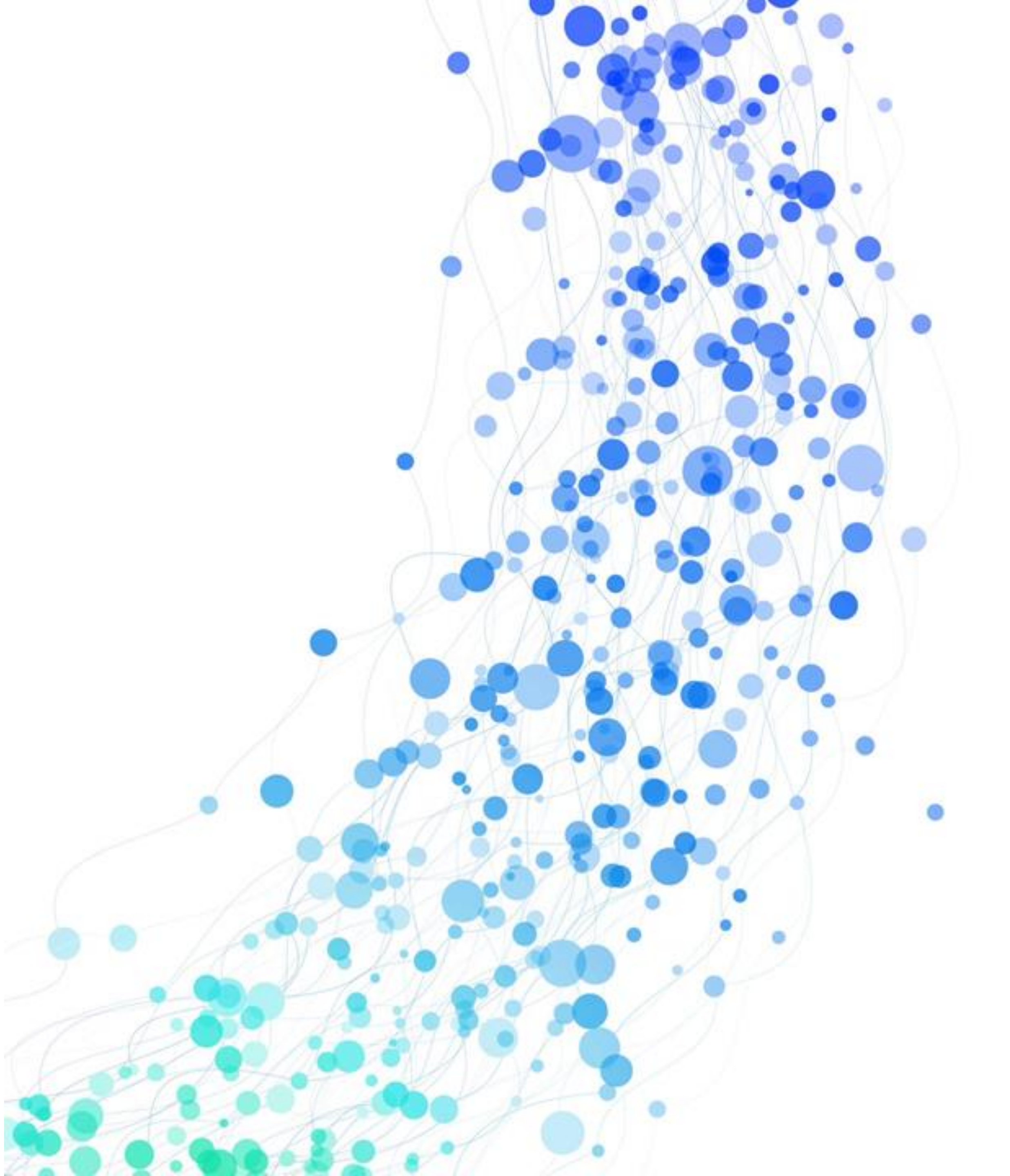
LECTURE 27: INTRO TO LARGE LANGUAGE MODELS



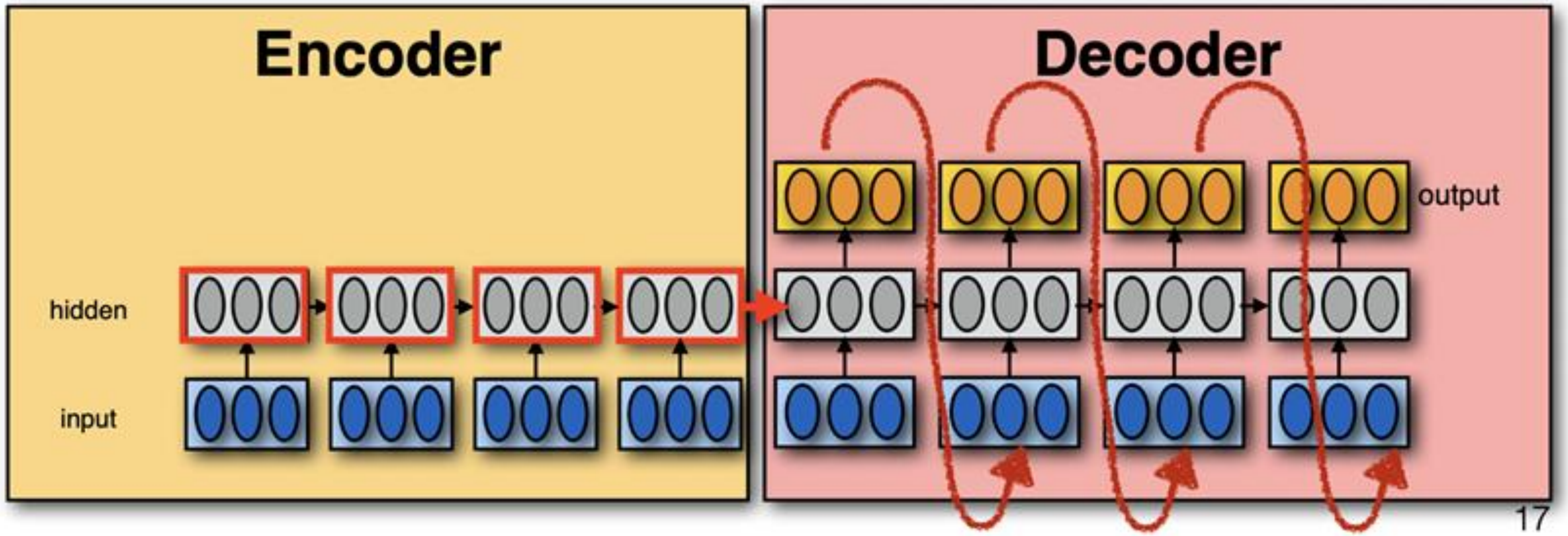
Mehmet Can Yavuz, PhD.

Adapted from Julia Hockenmaier, NLP S2023 - course material
https://courses.grainger.illinois.edu/cs447/sp2023/

RECAP: SEQ2SEQ, TRANSFORMERS

ENCODER-DECODER (SEQ2SEQ) MODEL

- The **decoder** is a language model that generates an output sequence **conditioned on the input** sequence.
  - **Vanilla RNN**: condition on the **last** hidden state
  - **Attention**: condition on **all** hidden states



TRANSFORMERS USE SELF-ATTENTION

- **Attention so far** (in seq2seq architectures):
  - In the ***decoder*** (which has access to the complete input sequence), compute attention weights over ***encoder*** positions that depend on each ***decoder*** position
- **Self-attention:**
  - If the ***encoder*** has access to the complete input sequence,
  - we can also compute attention weights over ***encoder*** positions that depend on each ***encoder*** position

self-attention:

encoder

- For each *decoder* position *t*…,
  - …Compute an attention weight for each *encoder* position *s*
    - …Renormalize these weights (that depend on *t*) w/ softmax to get a new weighted avg. of the input sequence vectors

TRANSFORMER ARCHITECTURE

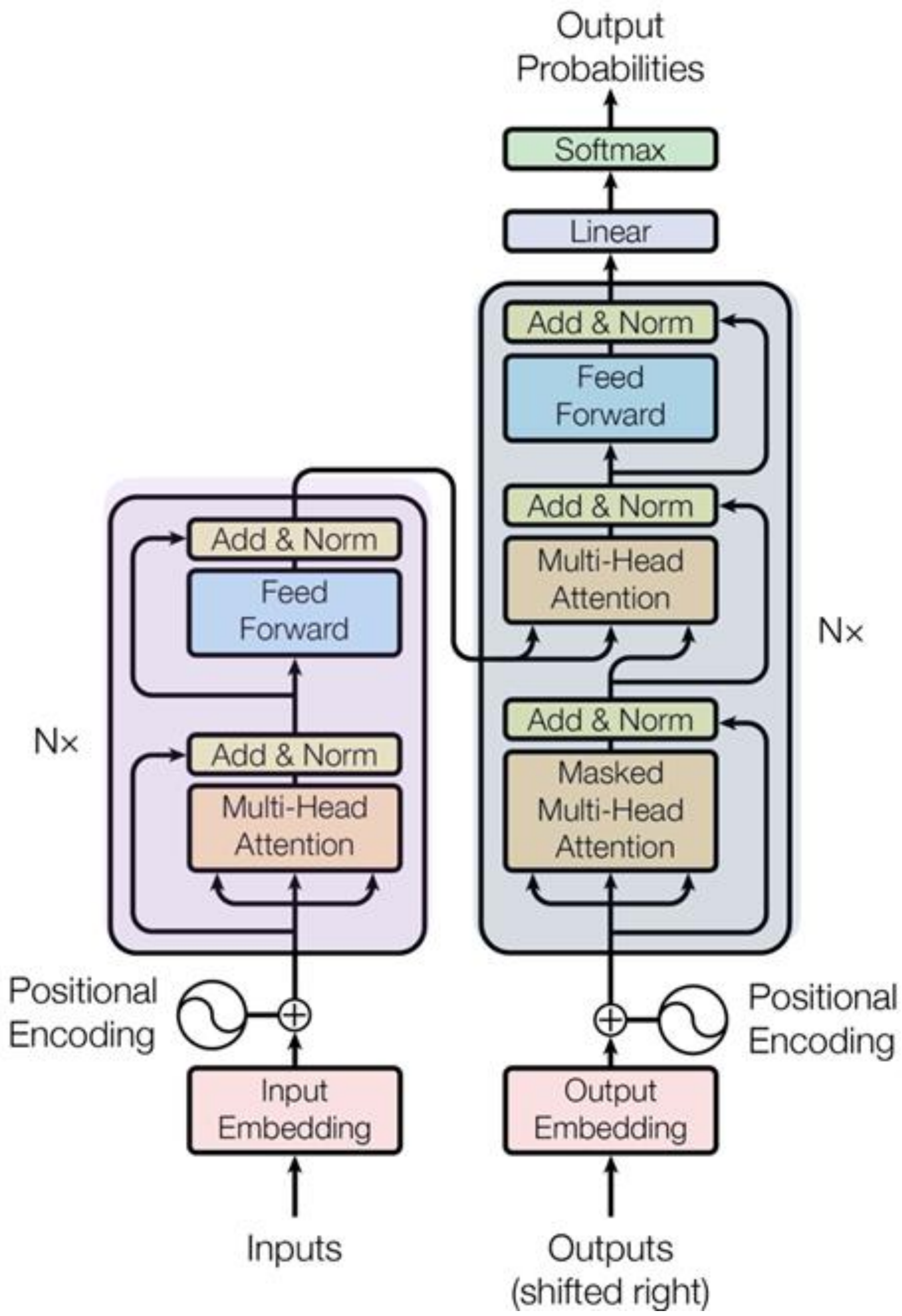
•Non-Recurrent Encoder-Decoder architecture

— No hidden states

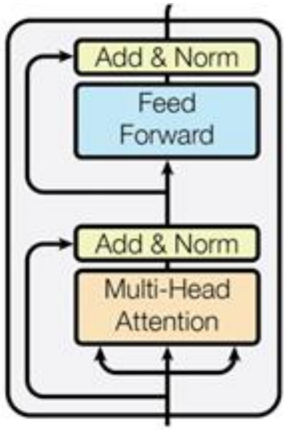
— Context information captured via attention and positional encodings

— Consists of stacks of layers with various sublayers

- Vaswani et al, NIPS 2017



ENCODER VASWANI ET AL, NIPS 2017



A stack of **N=6 identical layers**

All layers and sublayers are 512-dimensional

Each layer consists of **two sublayers**

— one **multi-head self attention** layer

— one **position-wise feed forward** layer

Each sublayer is followed by an **"Add & Norm"** layer:

… a **residual connection** $\mathbf{x} + \mathrm{Sublayer}(\mathbf{x})$

(the input $\mathbf{x}$ is added to the output of the sublayer)

... followed by a **normalization step**

(using the mean and standard deviation of its activations)

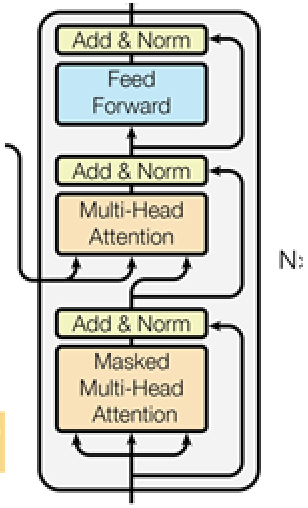
$$\mathrm{LayerNorm}(\mathbf{x} + \mathrm{Sublayer}(\mathbf{x}))$$

DECODER VASWANI ET AL, NIPS 2017

A stack of N=6 identical layers
All layers and sublayers are 512-dimensional

Each layer consists of **three** sublayers
— one **masked multi-head self attention layer**
over **decoder** output
(masked, i.e. ignoring future tokens)
— one **multi-headed attention layer**
over **encoder** output
— one **position-wise feed forward layer**

Each sublayer has a residual connection
and is normalized: $\mathrm{LayerNorm}(\mathbf{x} + \mathrm{Sublayer}(\mathbf{x}))$



SUBWORD TOKENIZATION

BPE TOKENIZATION (SENNRICH ET AL, ACL 2016)

BytePair Encoding (Gage 1994): a compression algorithm that iteratively replaces the most common pair of adjacent bytes with a single, unused byte

BPE tokenization: introduce new tokens by merging the most common adjacent pairs of tokens

Start with all characters, plus a special end-of-word character
Introduce new token by merging the most common pair of adjacent tokens.

(Assumption: each individual token will still occur in a different context, so we will also keep both tokens in the vocabulary)

Machine translation: train one tokenizer across both languages (better generalization for related languages)

WORDPIECE TOKENIZATION (WU ET AL, 2016)

Part of Google's LSTM-based Neural Machine Translation system (https://arxiv.org/pdf/1609.08144.pdf)

Segment words into **subtokens** (with special word boundary symbols to recover original tokenization)

- **Input:** Jet makers feud over seat width with big orders at stake
- **Output:** _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

Training of Wordpiece:

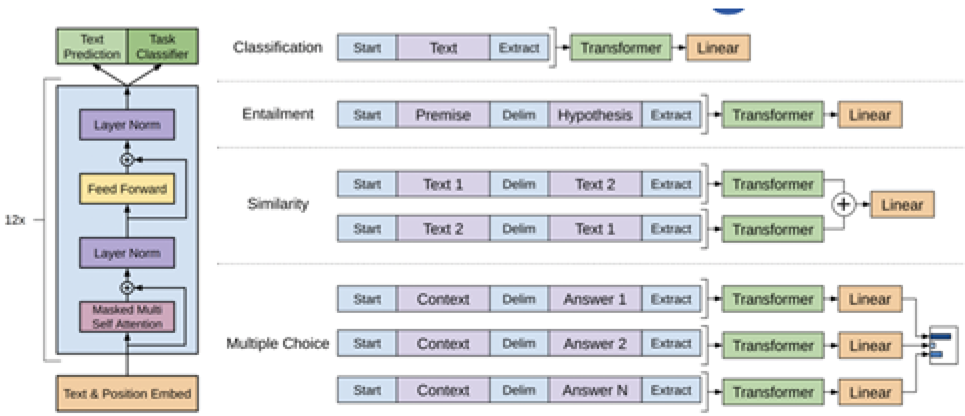
- Specify desired number of tokens, D
- Add word boundary token (at beginning of words)
- Optimization task: greedily merge adjacent characters to improve log-likelihood of data until the vocabulary has size D.

SUBWORD REGULARIZATION (KUDO, ACL 2018)

- **Observation:** Subword tokenization can be ambiguous
  - Can this be harnessed?
- **Approach:** Train a (translation) model with (multiple) subword segmentations that are sampled from a character-based **unigram language model**
- **Training the unigram model:**
  - Start with an overly large seed vocabulary V (all possible single-character tokens and many multi-character tokens)
  - Randomly sample a segmentation from the unigram model
  - Decide which multi-character words to remove from V based on how the likelihood decreases by removing them
  - Stop when the vocabulary is small enough.

GPT

GENERATIVE PRE-TRAINING (RADFORD ET AL, 2018)



Auto-regressive 12-layer transformer **decoder**

- Each token only conditioned on preceding context
- BPE tokenization (|V| = 40K), 768 hidden size, 12 attention heads

Pre-trained on raw text as a language model

- (Maximize the probability of predicting the next word)

Fine-tuned on labeled data (and language modeling)

- Include new **start**, **delimiter** and **end** tokens, plus **linear** layer added to last layer of **end token** output.

BERT

BERT (DEVLIN ET AL, NAACL 2019)

Fully bidirectional transformer encoder

- BERT$_{base}$: 12 layers, hidden size=768, 12 att'n heads (110M parameters)
- BERT$_{large}$: 24 layers, hidden size=1024, 16 attention heads (340M parameters)

Input: sum of token, positional, segment embeddings

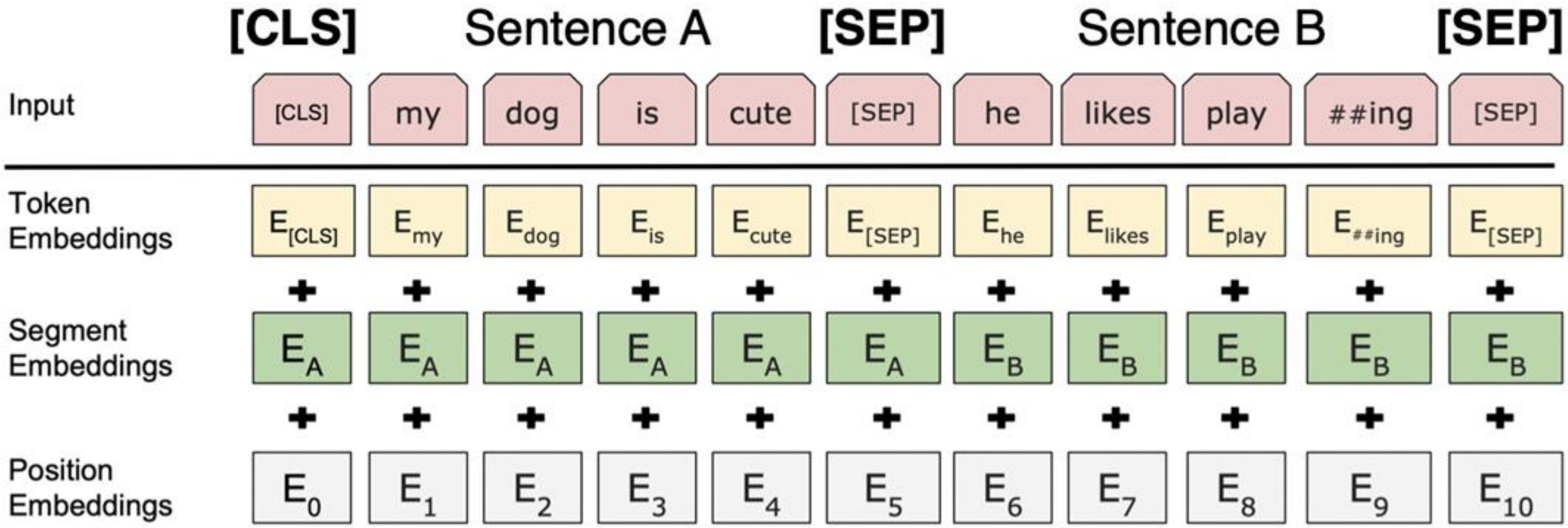
- **Segment embeddings** (A and B): is this token part of sentence A (before SEP) or sentence B (after SEP)?

[CLS] and [SEP] tokens: added during pre-training

Pre-training tasks:

- – Masked language modeling
- – Next sentence prediction

BERT INPUT


[CLS]
Sentence A
[SEP]
Sentence B
[SEP]
Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

PRE-TRAINING TASKS

BERT is jointly pre-trained on two tasks:

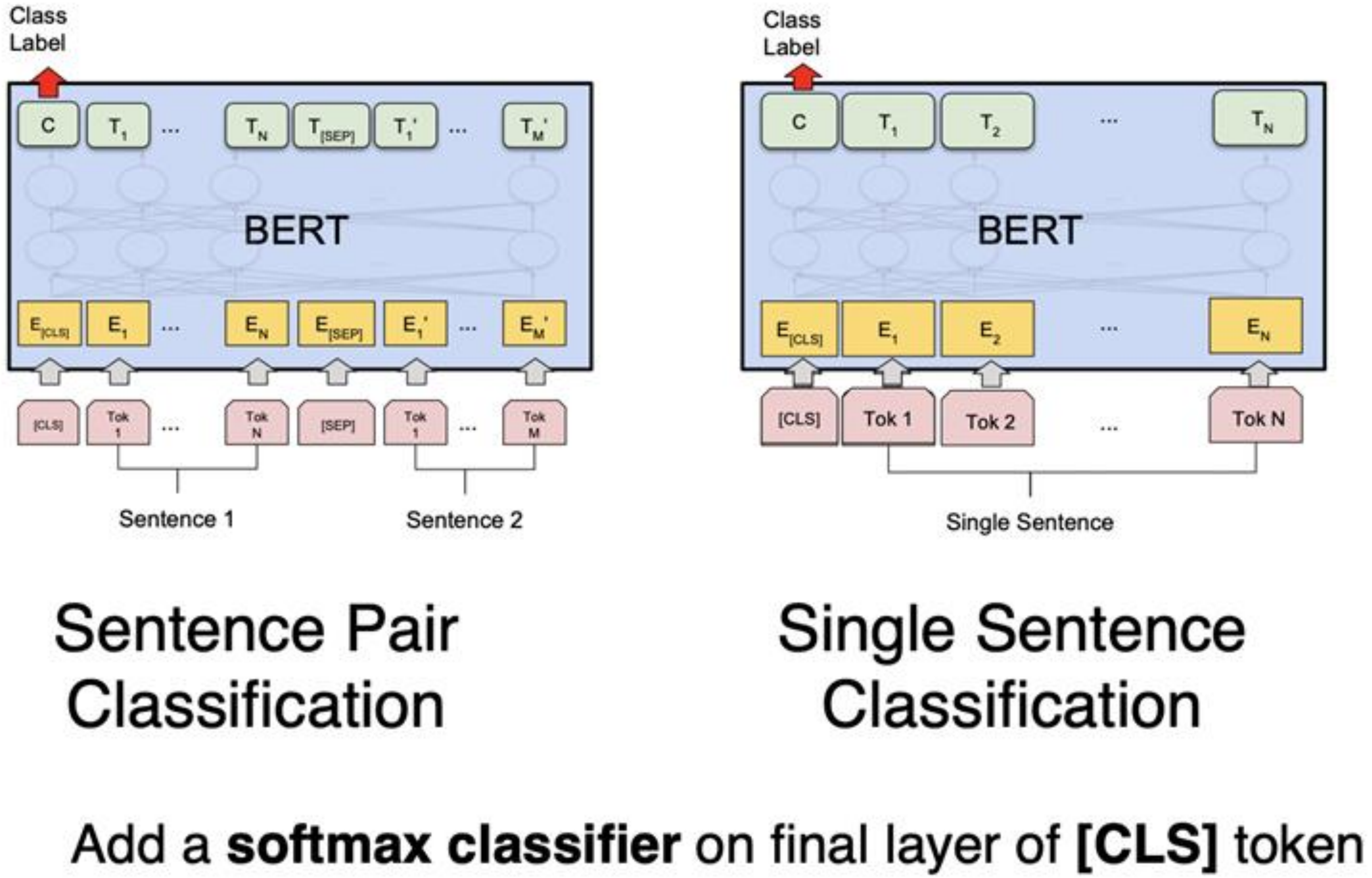
Next-sentence prediction: [based on CLS token]

- Does sentence B follow sentence A in a real document?

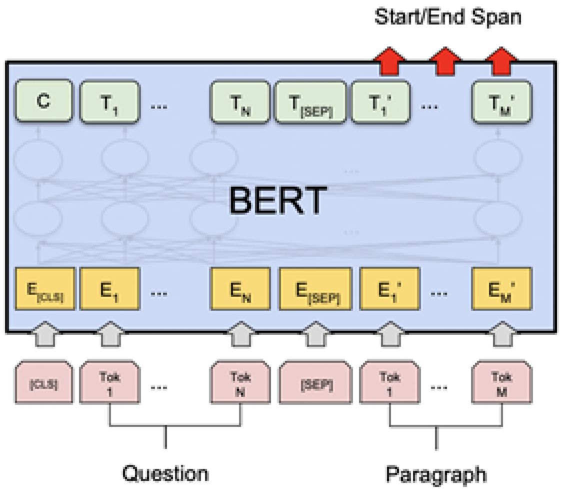
Masked language modeling:

- **15%** of tokens are randomly chosen as **masking tokens**
- 10% of the time, a masking token remains unchanged
- 10% of the time, a masking token is replaced by a random token
- **80% of the time**, a masking token is replaced by **[MASK]** and the **output layer has to predict the original token**

USING BERT FOR CLASSIFICATION



USING BERT FOR QUESTION-ANSWERING



Input: [CLS] question [SEP] answer passage [SEP]

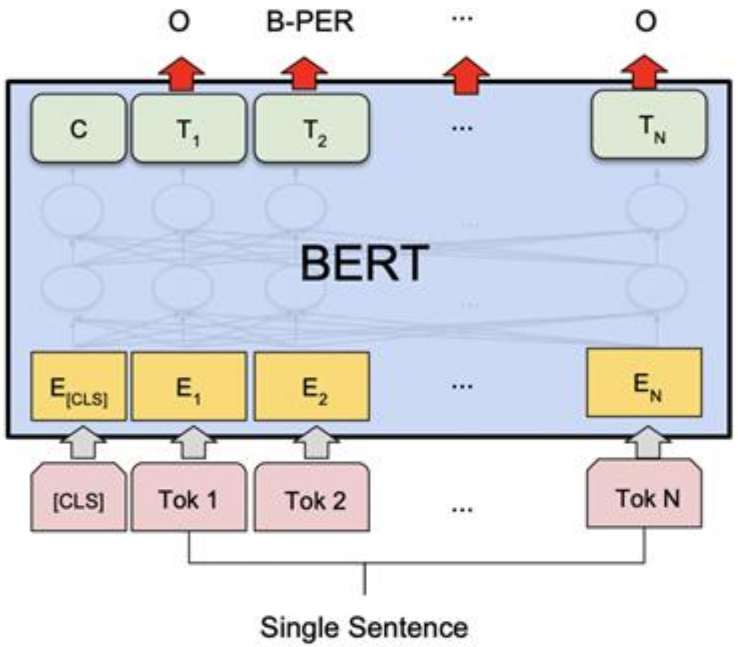
Learn to predict a **START** and an **END token** on answer tokens

Represent START and END as H-dimensional vectors *S*, *E*

Find the most likely start and end tokens in the answer by computing a softmax over the dot product of all token embeddings T_i and S (or *E*)

$$P(T_i \text{ is start}) = \frac{\exp(T_i \cdot S)}{\sum_j \exp(T_j \cdot S)}$$

USING BERT FOR SEQUENCE LABELING



Add a **softmax classifier** to the tokens in the sequence

FINE-TUNING BERT

To use BERT on any task, it needs to be fine-tuned:

— Add any new parts to the model

- (e.g. classifier layers)
  - This will add **new parameters** (initialized randomly)

— Retrain the entire model (update all parameters)

MORE COMPACT BERT MODELS (TURC ET AL., 2019)

Pre-training and fine-tuning works well on much smaller BERT variants

https://arxiv.org/abs/1908.08962

Additional improvements through knowledge distillation:

- – **Pre-train** a compact model ('student') in the standard way
- – Train/Fine-tune a large model ('teacher') on the target task
- – **Knowledge distillation** step: Train the student on noisy task predictions made by teacher
- – Fine-tune student on actual task data

Students can have more layers (but smaller embeddings) than models trained in the standard way

BERT VARIANTS

RoBERTA (LIU ET AL. 2019)

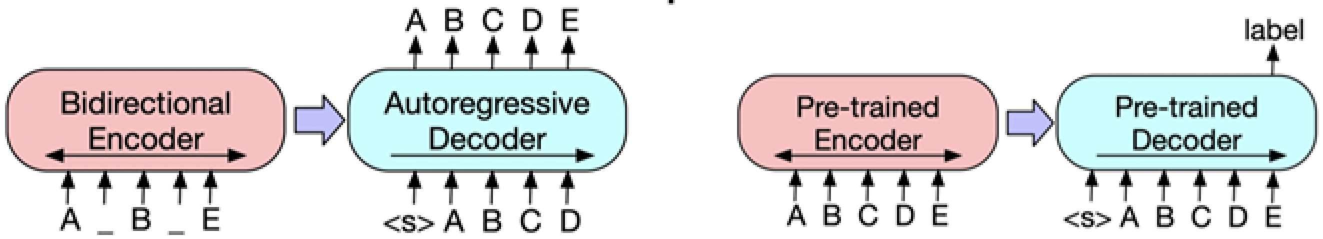
- Investigates **better pre-training** for BERT
- Found that BERT was undertrained.
- Optimizes hyperparameter choice.
- Evaluates next-sentence prediction task
- RoBERTA outperforms BERT on several tasks.
- **Pre-training improvements:**
  - **Dynamic masking:** randomly change which tokens in a sentence get masked (BERT: same tokens in each epoch)
  - **Much larger batch sizes** (2K sentences instead of 256)
  - **Use byte-level BPE, not character level BPE**

BART (LEWIS ET AL., ACL 2020)

Combines **bidirectional encoder** (like BERT) with **auto-regressive** (unidirectional) **decoder** (like GPT)

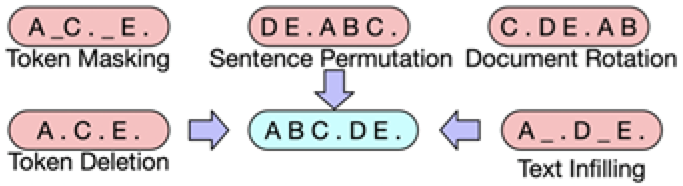
Used for **classification**, **generation**, **translation**

Uses final token of decoder sequence for classification tasks.



Pre-training: corrupts (encoder) input with **masking**, **deletion, rotation, permutation, infilling.**

Decoder needs to recover original input



SENTENCEBERT (REIMERS & GUREVYCH, EMNLP 2019)

- For tasks that require scoring of **sentence pairs**
  - (e.g. semantic textual similarity, or entailment recognition)
  - Motivation: BERT treats sequence pairs as one (long) sequence, but cross-attention across O(2n) words is very slow.
- SentenceBERT Solution: **Siamese network**
  - Run BERT over each sentence independently
  - Compute **one vector** (**u** and **v**) **for each sentence** by (mean or max) pooling over word embeddings or by using CLS token
- **Classification tasks:**
  - concatenate **u**, **v**, and **u–v**,
  - use as input to softmax
  - **Similarity tasks:**
  - use the cosine similarity
  - of **u** and **v** as similarity score
  - **Training:** start with BERT, fine-tune Siamese model on task-specific data