

344.063/163 KV Special Topic: Natural Language Processing with Deep Learning Large Language Models



Navid Rekab-Saz

navid.rekabsaz@jku.at

Institute of Computational Perception

Agenda

- Large language models
 - Decoders
 - Encoders
 - Encoder-Decoders

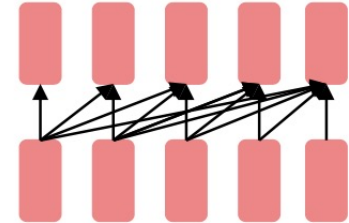
Agenda

- **Large language models**
 - **Decoders**
 - Encoders
 - Encoder-Decoders

Three types of large-scale pretrained LMs

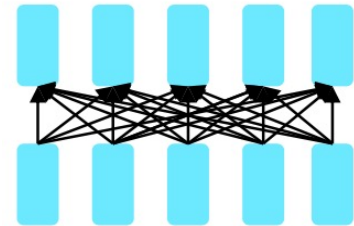
■ Decoders

- “Normal” LM objective: predict the next token conditioned on the previous tokens (unidirectional)
- Training and inference is auto-regressive (one after each other)
- Particularly suited for generating text



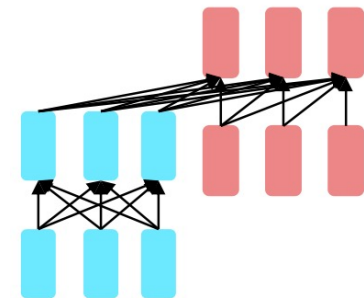
■ Encoders

- Input is encoded into contextualized embeddings
- Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
- Training is bidirectional – model sees whole the sequence (past and future)



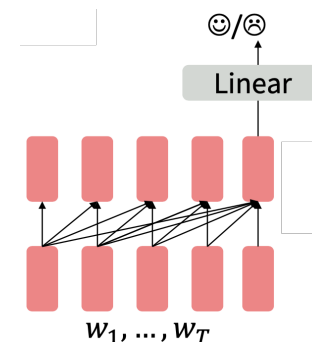
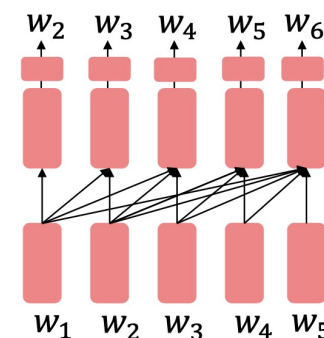
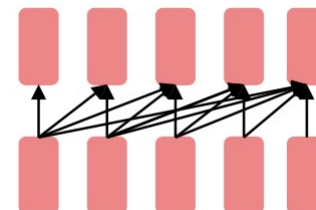
■ Encoder-Decoders

- The encoder encodes whole the input (bidirectional)
- The decoder generates the output in auto-regressive fashion



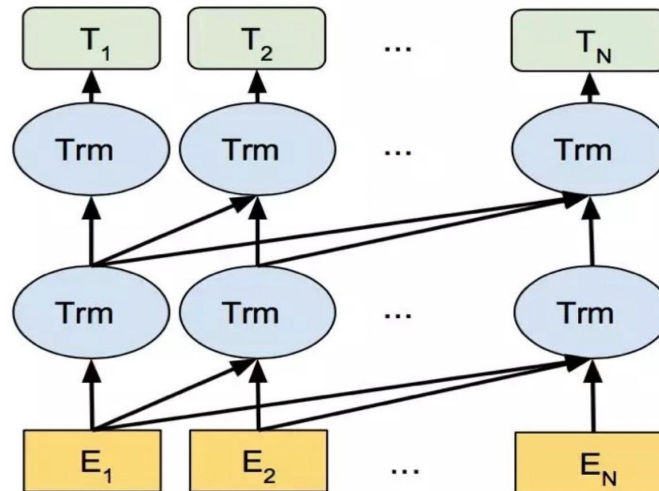
Decoders LM

- The architecture can be composed of (multi-layers of) RNNs or Transformers
- Training
 - “Normal” LM objective predicts the probability distribution of the next word and optimize the network based on the actual word
- Text generation
 - Next word is generated by sampling from the predicted probability distribution
- Downstream tasks
 - Fine-tuning and prediction in downstream task is commonly done using the last output embedding



GPT: Generative Pretrained Transformer

- 12 layers of Transformer decoder
- Tokenization using Byte-pair encoding
- Trained on BooksCorpus
- GPT is followed by much larger models: GPT-2 and GPT-3



Text Generation

- GPT-2 and later GPT-3 show very “convincing” results on text generation
 - Try here: <https://transformer.huggingface.co>

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

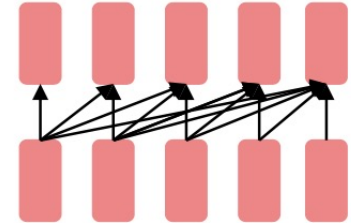
Agenda

- **Large language models**
 - Decoders
 - **Encoders**
 - Encoder-Decoders

Three types of large-scale pretrained LMs

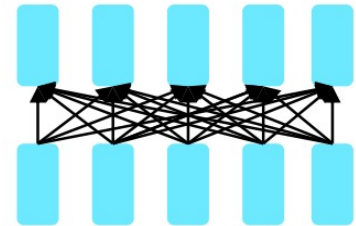
■ Decoders

- “Normal” LM objective: predict the next token conditioned on the previous tokens (unidirectional)
- Training and inference is auto-regressive (one after each other)
- Particularly suited for generating text



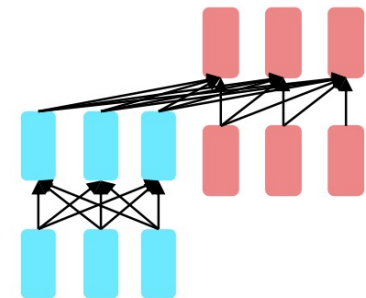
■ Encoders

- Input is encoded into contextualized embeddings
- Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
- Training is bidirectional – model sees whole the sequence (past and future)



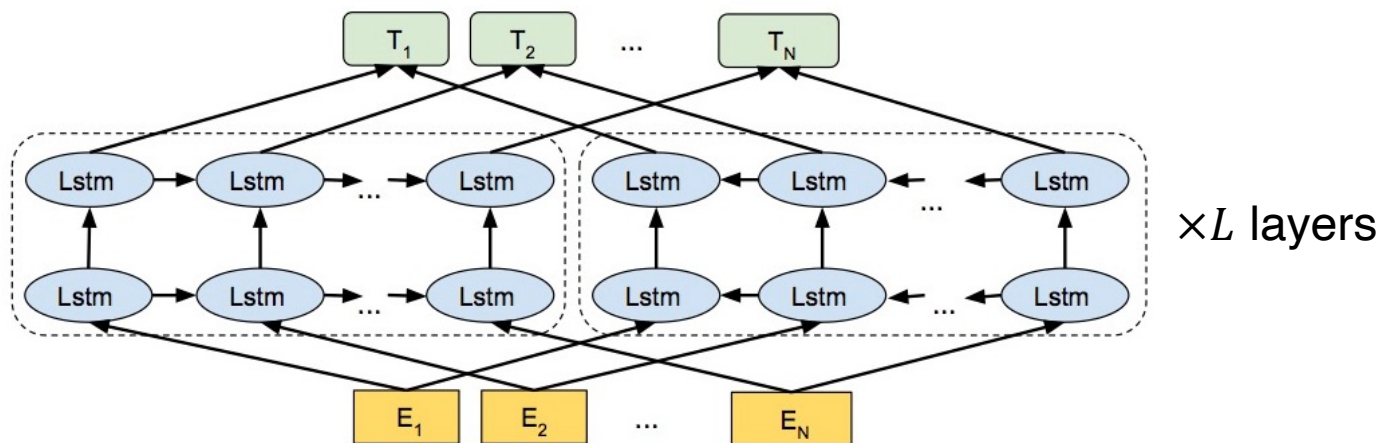
■ Encoder-Decoders

- The encoder encodes whole the input (bidirectional)
- The decoder generates the output in auto-regressive fashion



ELMo: Embeddings from Language Models

- ELMo's architecture: **Multi-layer Bidirectional LSTM**
- The model outputs the contextualized embeddings of the words of the input sequence
- Trained by predicting the next word, done in both directions
 - One time from left to right and one time from right to left
- Input word embeddings are created using character-based CNN
- Time complexity of both training and inference is a factor of sequence length
 - Due to the “step-by-step” nature of RNNs



Masked Language Model (MLM)

- Limitation of “normal” language modeling objective: it is constrained to using only the left (or right) context
 - Though language understanding requires the full context!
- **Masked Language Model** masks out $k\%$ of the input sequence and predicts the masked words in output

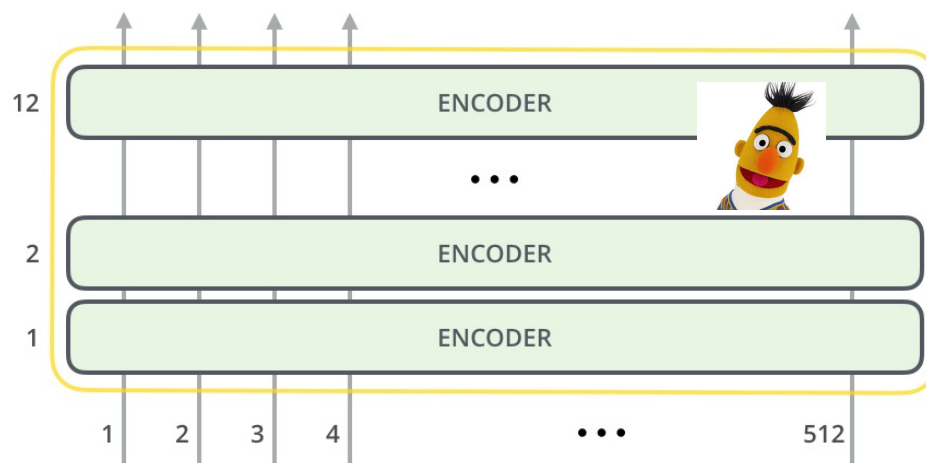
Example

sequence: Jim made spaghetti for his girlfriend and he was very proud!

Input: Jim made [MASK] for his girlfriend and [MASK] was very proud!

	↓	↓
predict:	spaghetti	he

BERT : Bidirectional Encoder Representation from Transformers



- BERT is a pre-trained language model, composed of multi-layers of Transformer encoder
- BERT ...
 - provides contextualized word embeddings
 - uses WordPiece for tokenization
 - uses sentence (sequence) pair encoding which provides ...
 - sequence embedding
 - an embedding for relation estimation between two sequences

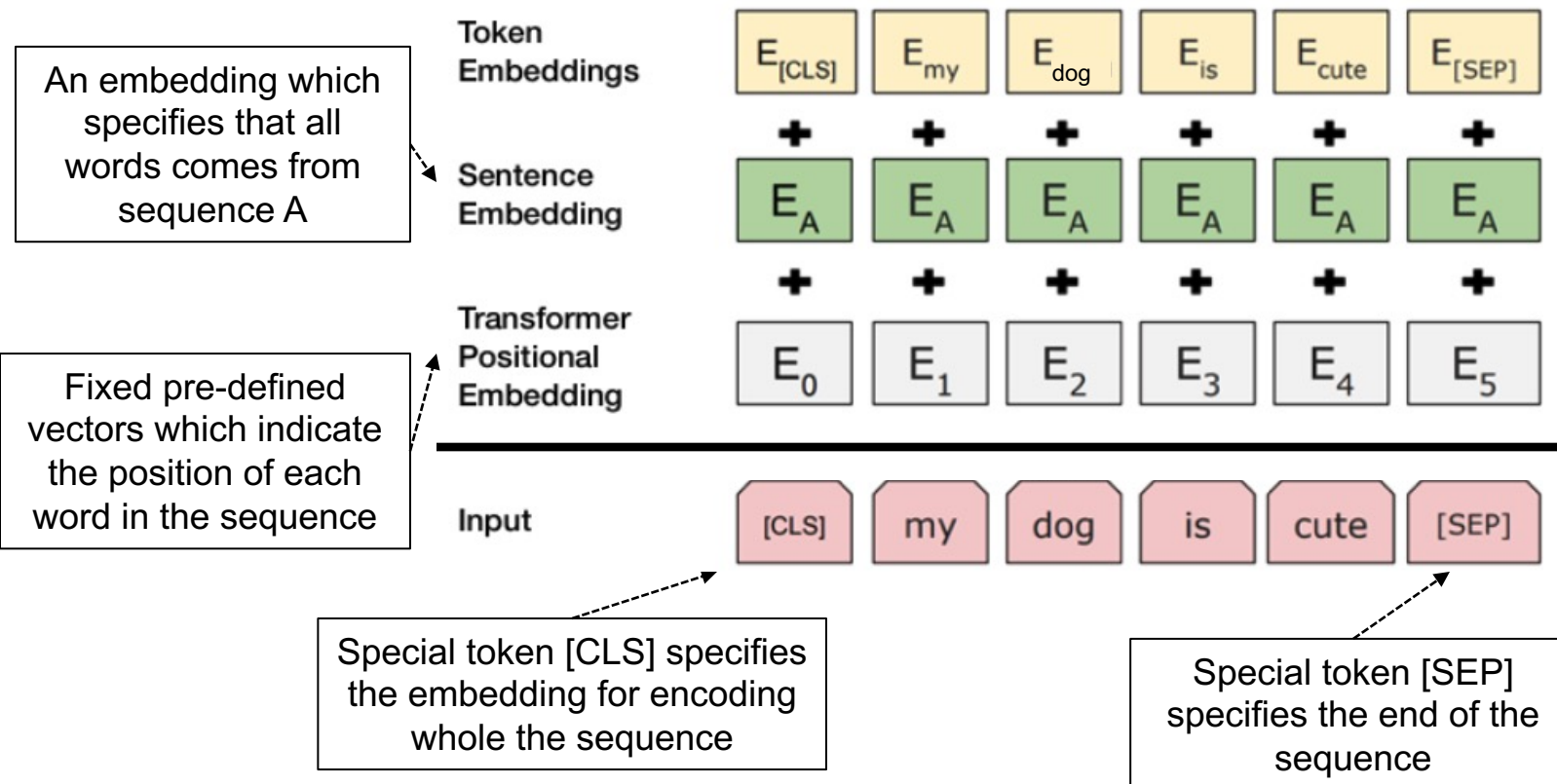
Training

- Trained using MLM on Wikipedia + BookCorpus
- Dictionary size is ~30K tokens (due to WordPiece subword tokenization)
- Specs of some provided pre-trained models:
 - BERT-Tiny: 2-layer, 128-hidden, 2-head, ~4M parameters*
 - BERT-Mini: 4-layer, 256-hidden, 4-head, ~11M parameters*
 - BERT-Base: 12-layer, 768-hidden, 12-head, ~110M parameters*
 - BERT-Large: 24-layer, 1024-hidden, 16-head, ~340M parameters*
- Some resources:
 - <https://github.com/google-research/bert>
 - Library to have BERT models in PyTorch: <https://huggingface.co/transformers/>

* For comparison, a (static) word embedding like word2vec with vocabulary size 200K and vector size 768 has 153M parameters

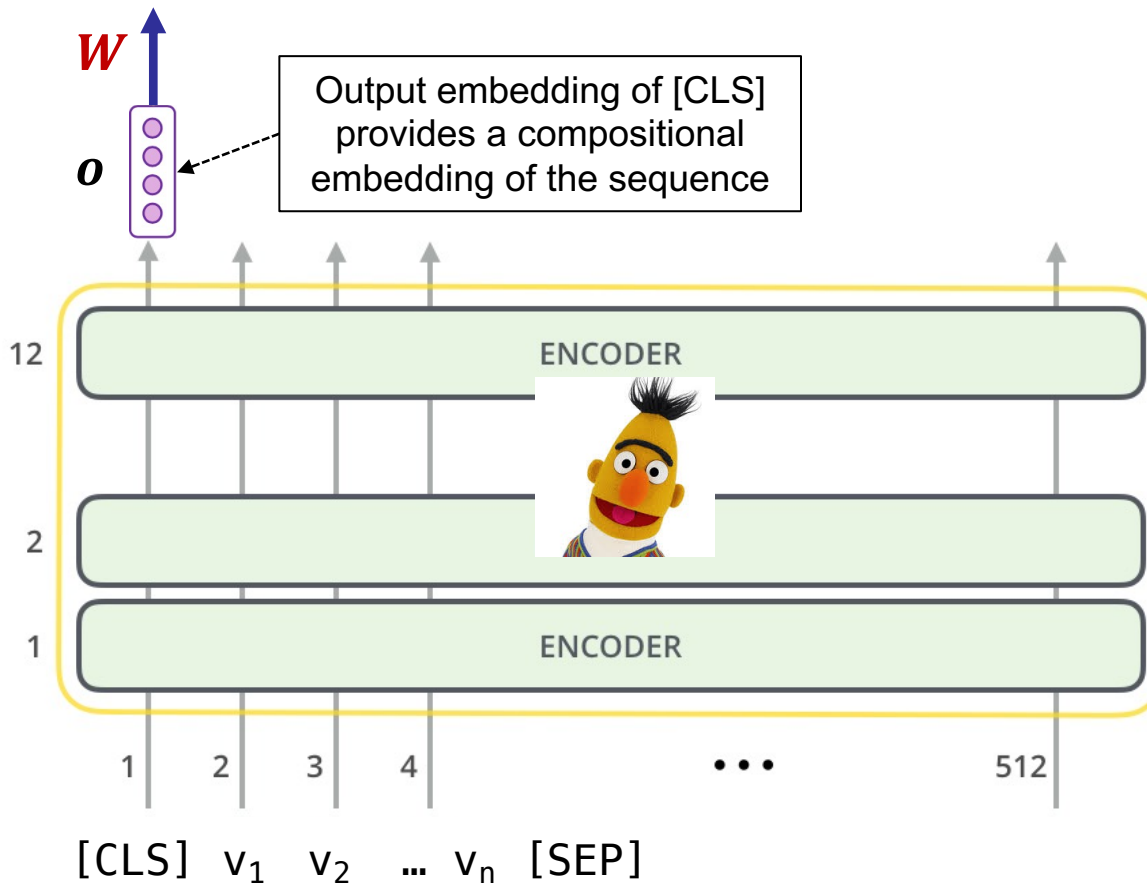
Input to BERT – one sequence

- The input embeddings to BERT are in fact the sum of three types of embeddings



BERT Fine-tuning for one sequence

$$\hat{y} = P(Y|D) = \text{softmax}(\mathbf{o}W + \mathbf{b})$$

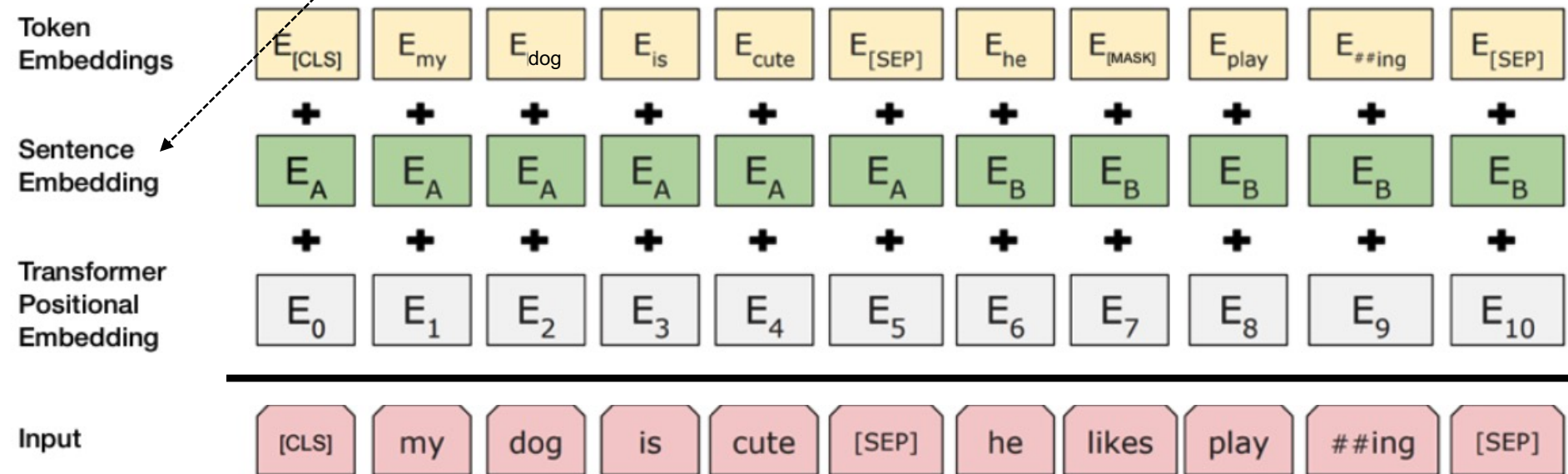


Sentence (Sequence) pair encoding

- Many NLP tasks need to calculate the relation between two sequences
 - E.g., question answering, information retrieval, natural language inference, paraphrasing, etc.
- During training, BERT also learns the **relationships between two sequences** using an additional binary classifier objective
 - The binary classifier take the output embedding of [CLS]
 - It predicts whether Sequence B is the actual sequence that proceeds Sequence A or a random sentence
 - This classifier is jointly optimized with the MLM objective
- If one sequence is given, the output of [CLS] is sequence embedding
- If two sequences are given, the output of [CLS] is the feature vector of the relation between the sequences

Input to BERT – two sequences

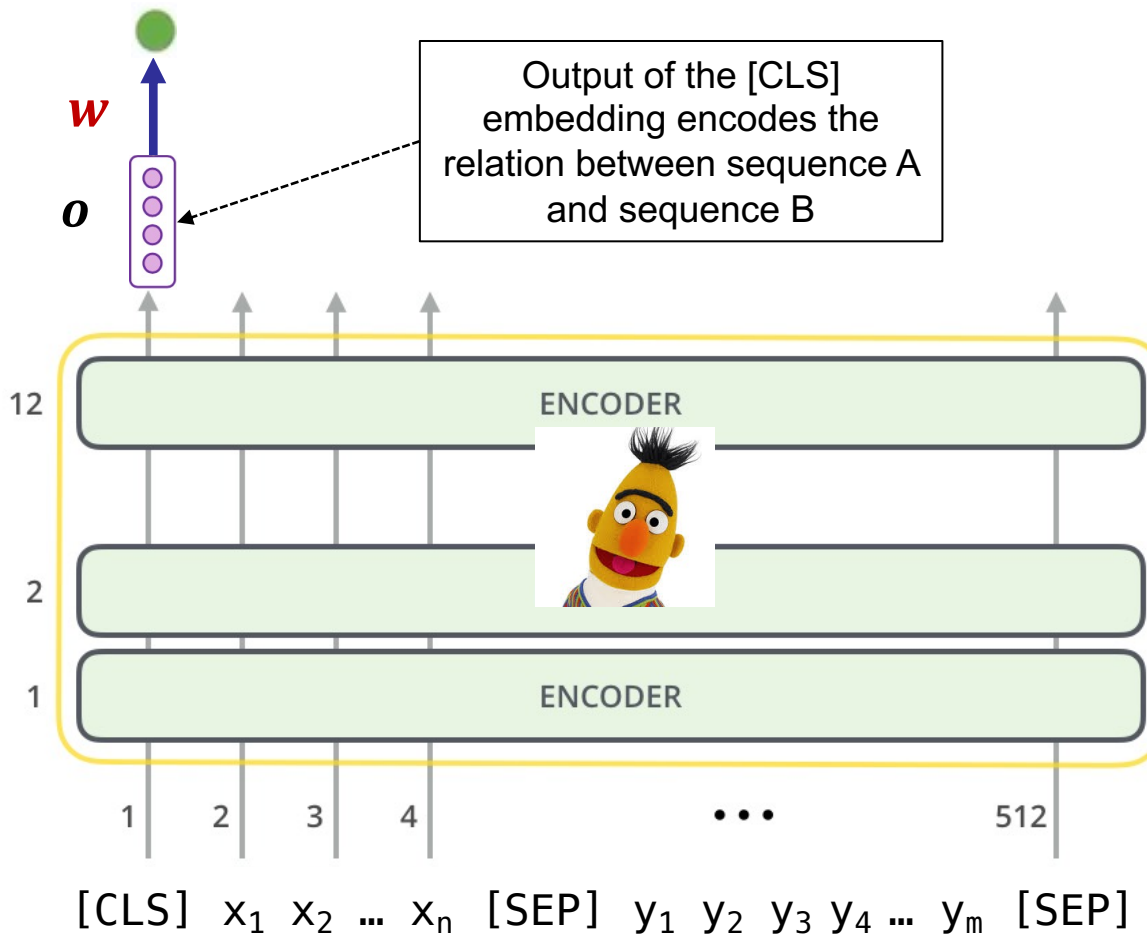
Sentence embeddings make a distinction between the embeddings of sequence A and sequence B



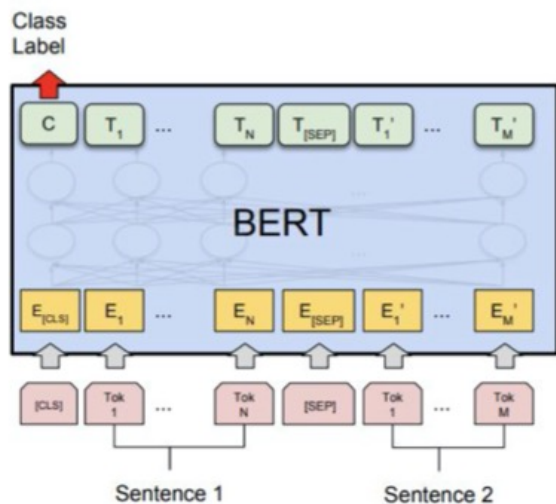
BERT Fine-tuning for two sequences

or any other matching task

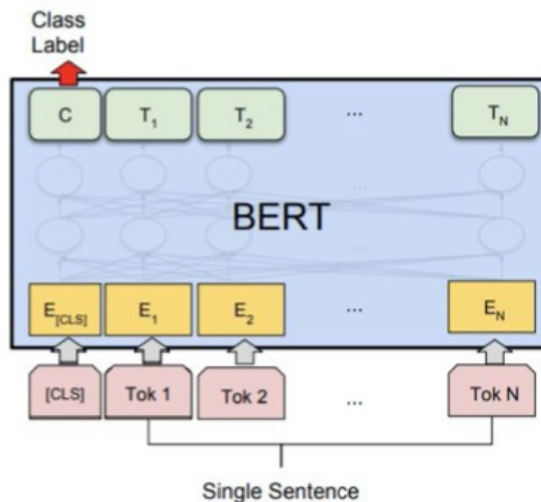
$$\text{score}(X, Y) = \sigma(o \cdot w)$$



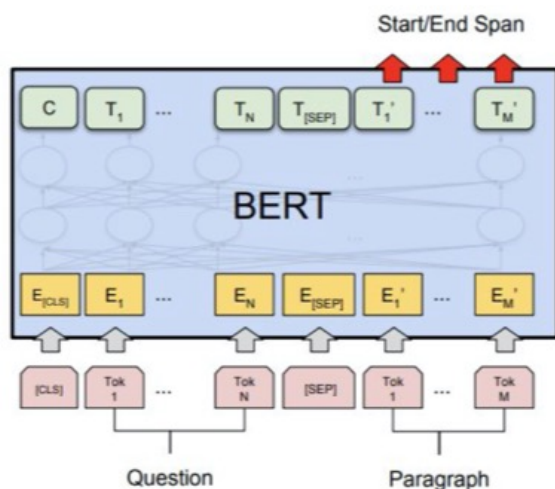
Fine tuning – inputs in different scenarios



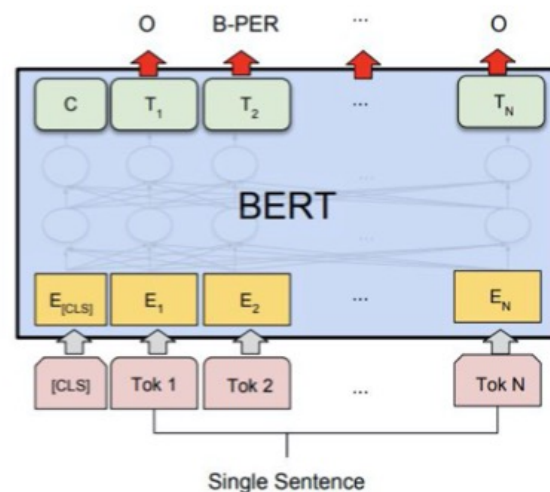
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



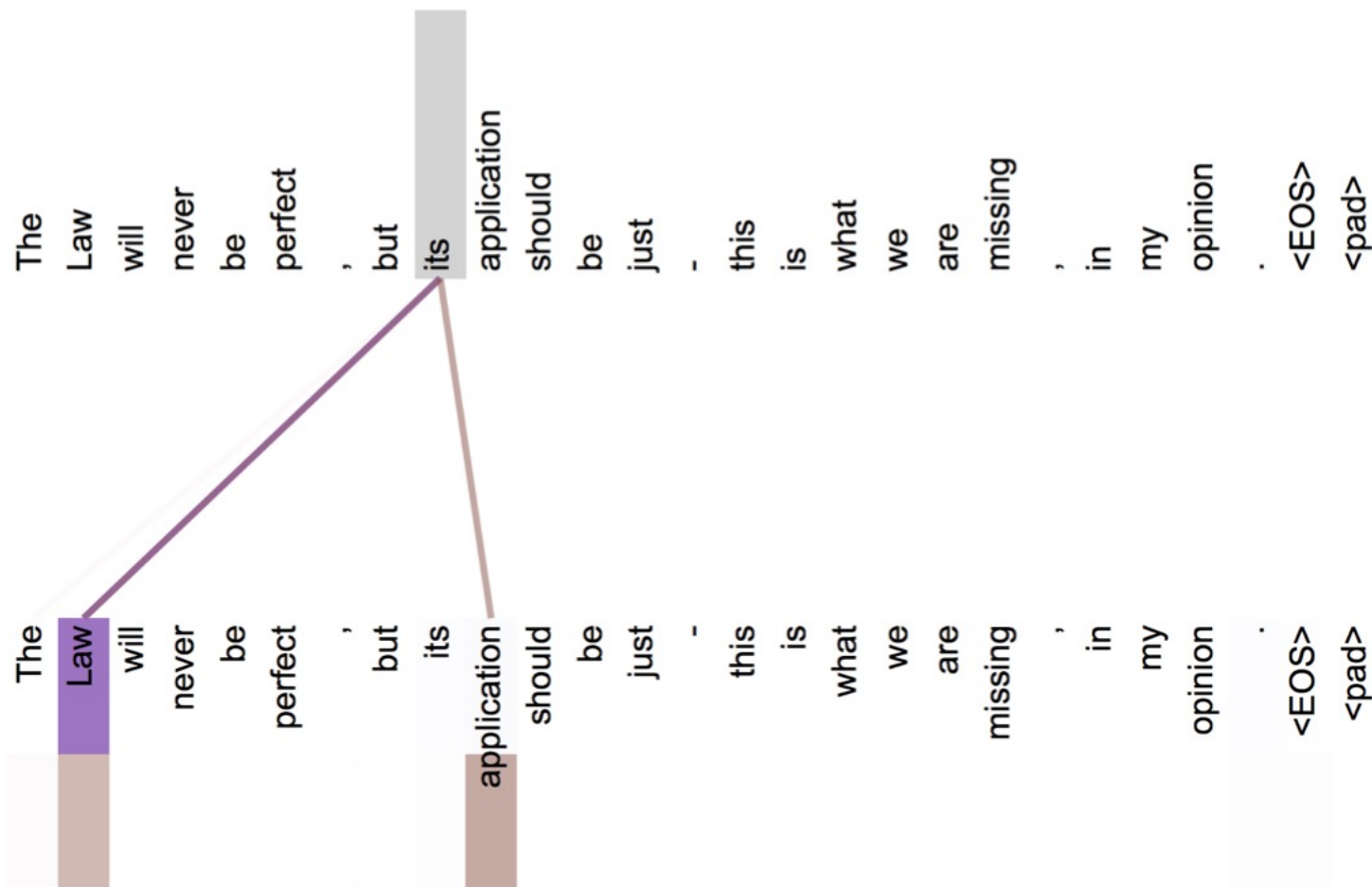
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Multi-head attention visualization

- A selected example:



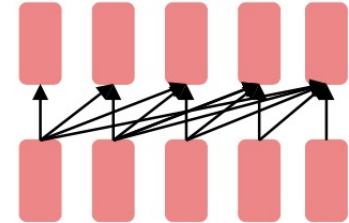
Agenda

- **Large language models**
 - Decoders
 - Encoders
 - **Encoder-Decoders**

Three types of large-scale pretrained LMs

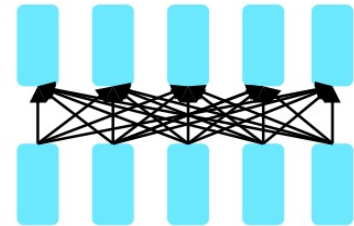
■ Decoders

- “Normal” LM objective: predict the next token conditioned on the previous tokens (unidirectional)
- Training and inference is auto-regressive (one after each other)
- Particularly suited for generating text



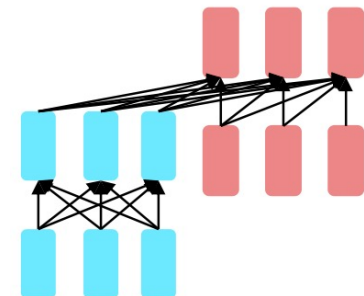
■ Encoders

- Input is encoded into contextualized embeddings
- Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
- Training is bidirectional – model sees whole the sequence (past and future)



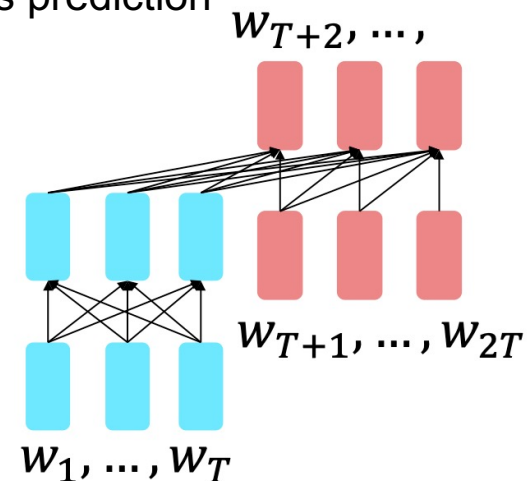
■ Encoder-Decoders

- The encoder encodes whole the input (bidirectional)
- The decoder generates the output in auto-regressive fashion



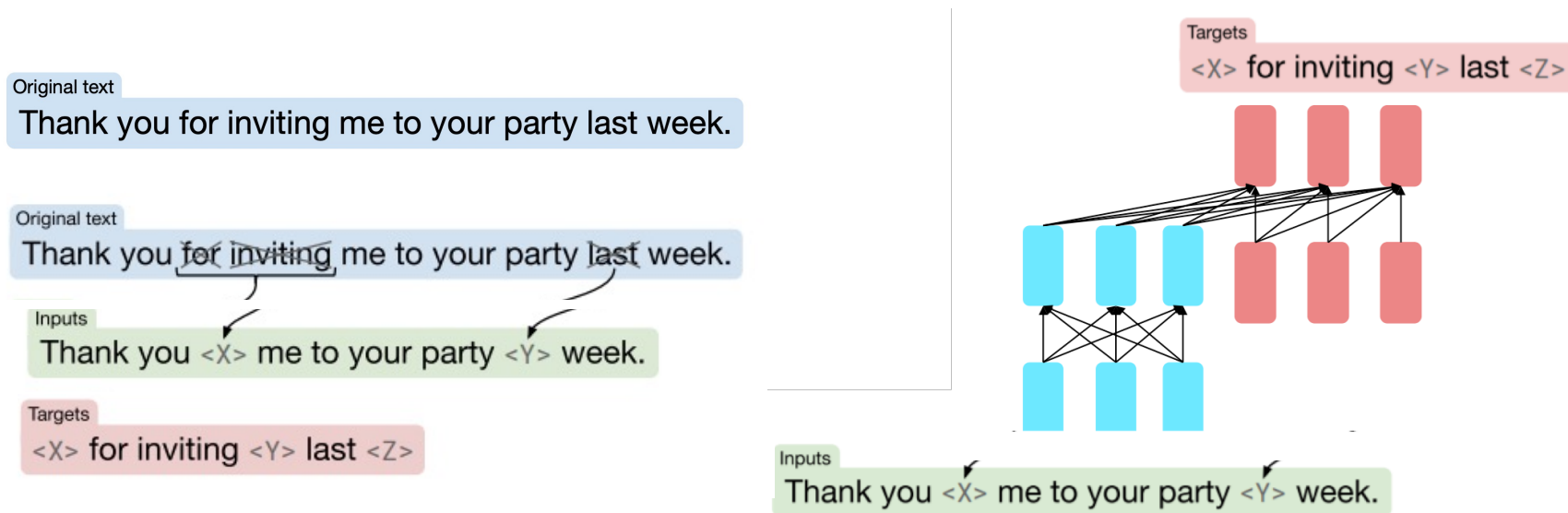
Training encoder-decoders

- How should we train an encoder-decoder LM?
- First approach – Using two consecutive sequences and next word prediction of the second sequence:
 - Take two sequences that follow each other in the corpus
 - Like w_1, \dots, w_T and w_{T+1}, \dots, w_{2T}
 - Pass the first sequence to the encoder
 - Apply “Normal” LM objective at decoder:
 - Generate the words of the second subsequence at decoder’s output one after each other
 - Optimize whole the model based on decoder’s prediction
- This approach resembles decoders LM
 - Though here the decoder has also access to the information coming from the encoder (a larger context)



T5: Text-to-Text Transfer Transformer

- Second approach – **span corruption**
 - For a given sequence, randomly select a few spans with different lengths
 - Replace the selected spans with unique placeholders.
 - Pass the edited sequence to the encoder
 - Generate the removed spans in the decoder
 - Optimize whole the model based on decoder's prediction



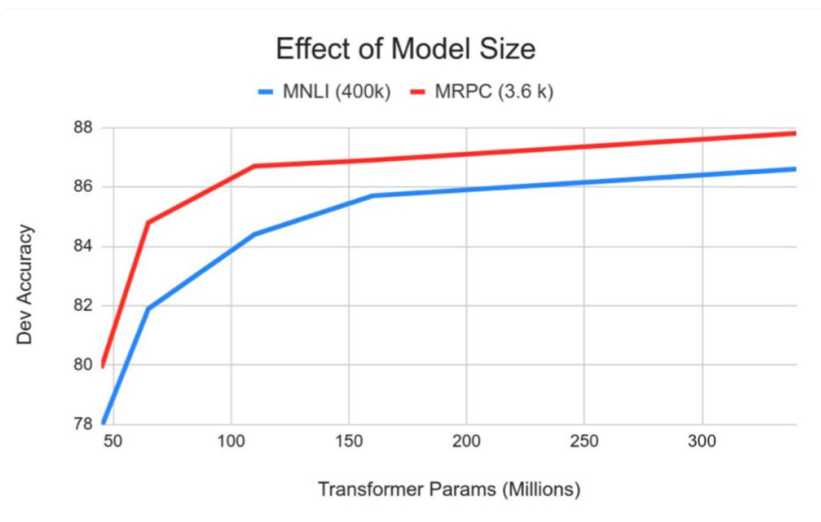
T5 – results on different seq2seq tasks

- Results show that span corruption works better than language modeling

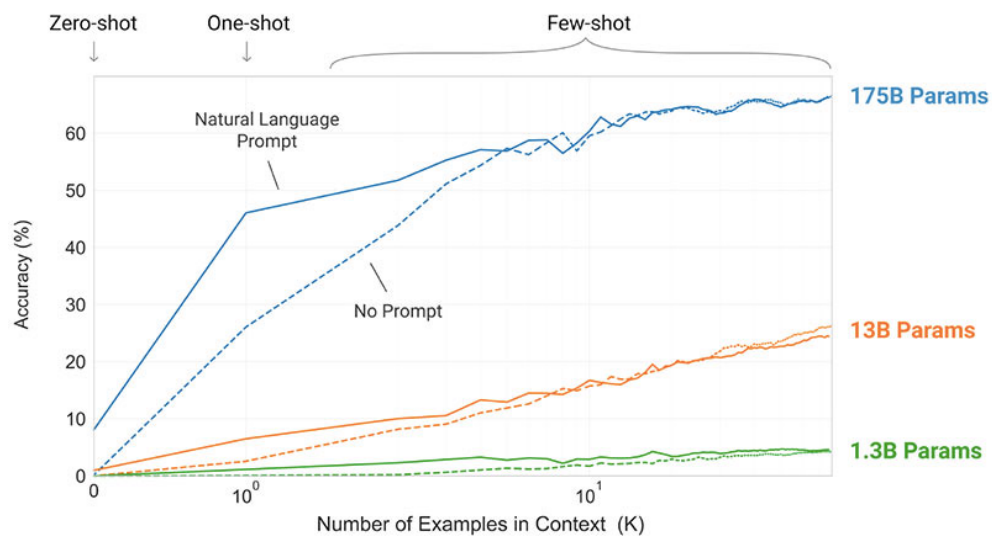
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

- Another pretrained encode-decoder LM: BART
 - Read more <https://arxiv.org/pdf/1910.13461.pdf>

Bigger models work better!



BERT variations



GPT variations