Pre-training Strategies

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

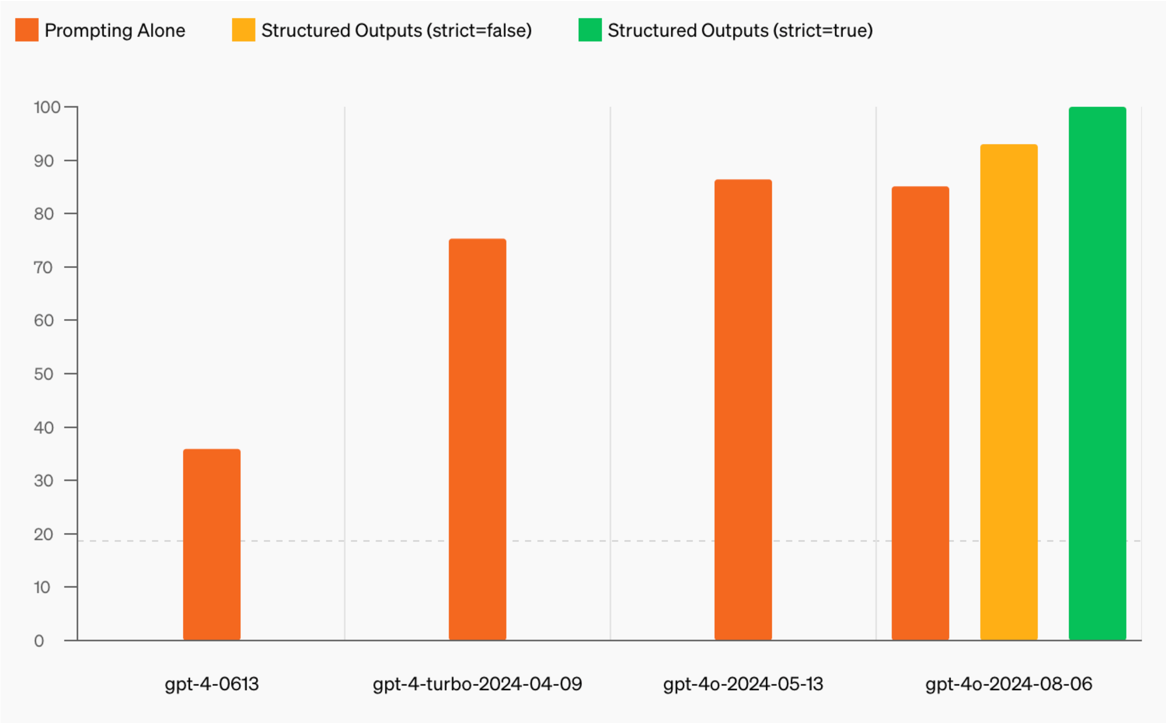
OpenAI introduces **Structured Outputs in the API** for GPT-4o !

Announced on
August 6, 2024

[OpenAI Blog]

Model outputs can now be made to reliably adhere to developer-supplied JSON Schemas

This will be very useful for developers looking to **build reliable applications** with GPT-4o API in the backend. With Structured Outputs in the API, model-generated outputs will **exactly match the JSON Schemas** provided by developers



With Structured Outputs, GPT-4o scores a perfect 100% in JSON schema following, while with just prompting GPT-4 scores less than 40% in output format following.

“You shall know a word by the company it keeps”

This quote is a summary of **distributional semantics**, and motivated **word2vec**. But:

“… the complete meaning of a word is always contextual,
and no study of meaning apart from a complete context
can be taken seriously.” (J. R. Firth 1935)

*I **record** the **record***

the two instances of ***record*** mean different things.

Background - Contextual Representations

- Word embeddings serve as the foundation for deep learning models in natural language processing.
- **Problem :** Word embeddings (word2vec, GloVe) are used without considering the context in which the words appear.

A bat flew out of the cave.

He hit the ball with a bat.

[0.286 , 0.792 , -0.177 , ...]

- **Solution :** Train contextual representations on text corpus

A bat flew out of the cave.

[-0.107 , 0.109 , -0.542 , ...]

He hit the ball with a bat.

[0.349 , 0.271 , 0.130 , ...]

The representation of the word should depend on the context in which it appears.

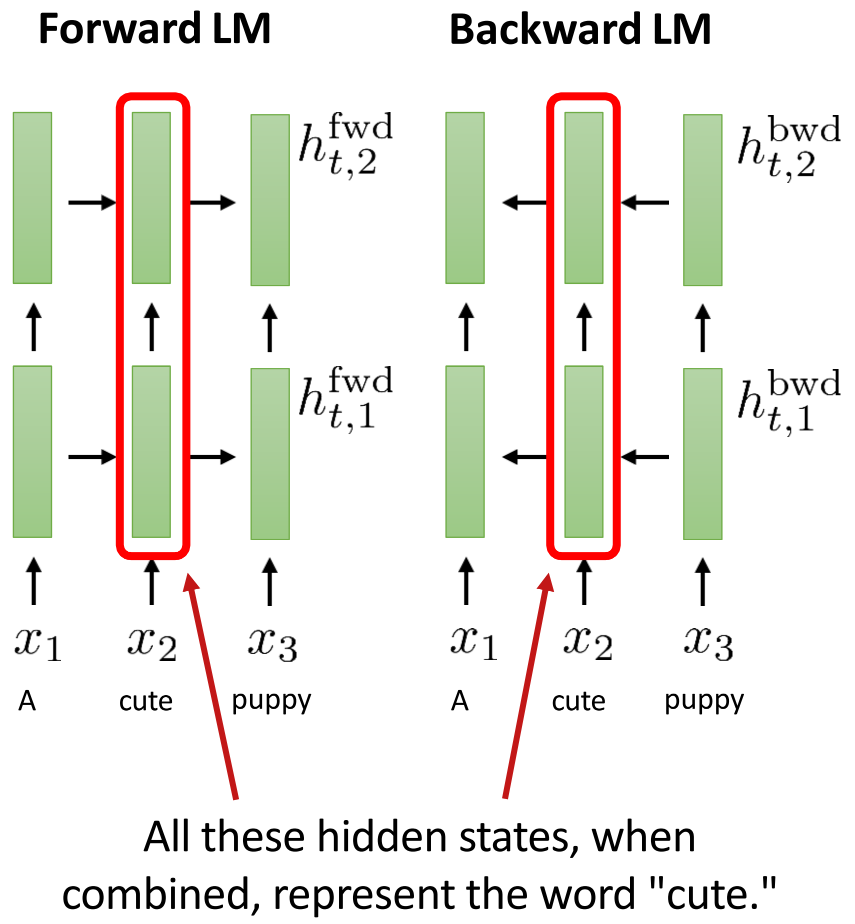
Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

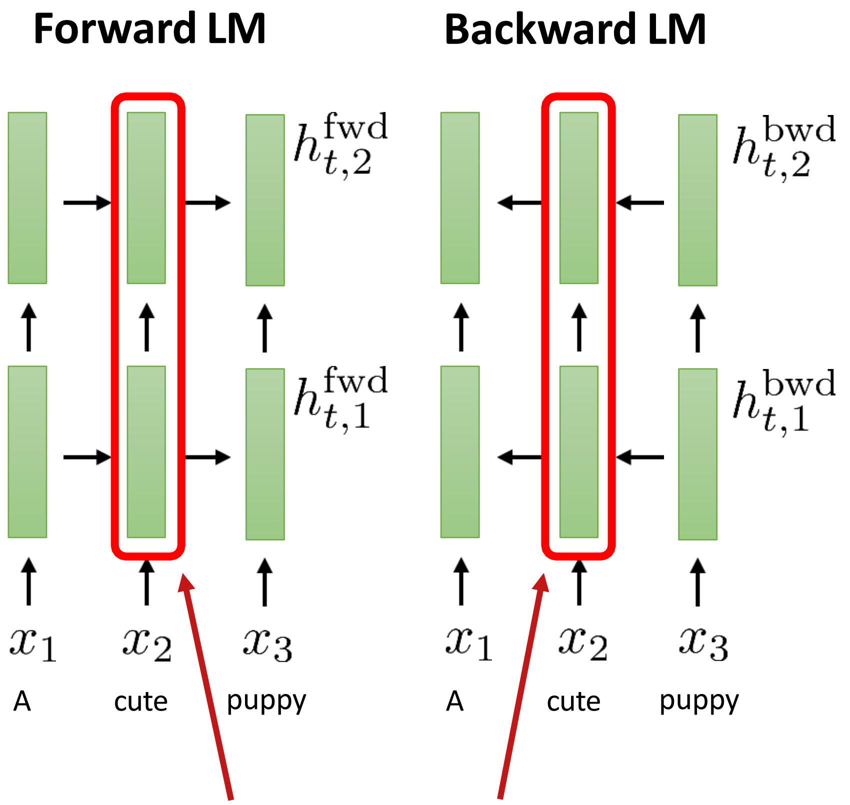
ELMo (**E**mbedding from **L**anguage **Mo**dels)



Replace static embeddings (lexicon lookup) with context-dependent embeddings (produced by a deep neural language model)

- Each token's representation is a function of the entire input sentence, computed by a deep (multi-layer) bidirectional language model
- Return for each token a (task-dependent) linear combination of its representation across layers.
- Different layers capture different information

ELMo Architecture



All these hidden states, when combined, represent the word "cute."

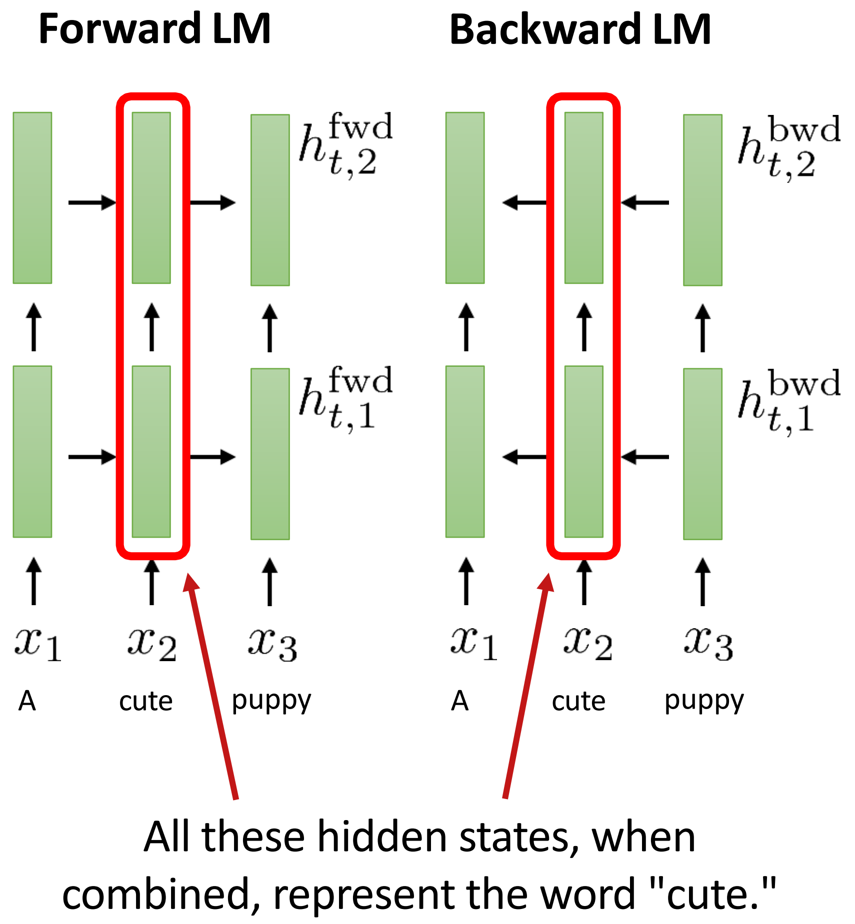
—Train a multi-layer bidirectional language model with character convolutions on raw text

—Each layer of this language model network computes a vector representation for each token.

— Freeze the parameters of the language model.

—For each task: train task-dependent softmax weights to combine the layer-wise representations into a single vector for each token *jointly* with a task- specific model that uses those vectors

ELMo Architecture



The forward LM is a deep LSTM that goes over the sequence from start to end to predict token t_k based on the prefix $t_1 \ldots t_{k-1}$:

$$p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s)$$

Parameters: token embeddings Θ_x LSTM $\overrightarrow{\Theta}_{LSTM}$ softmax Θ_s

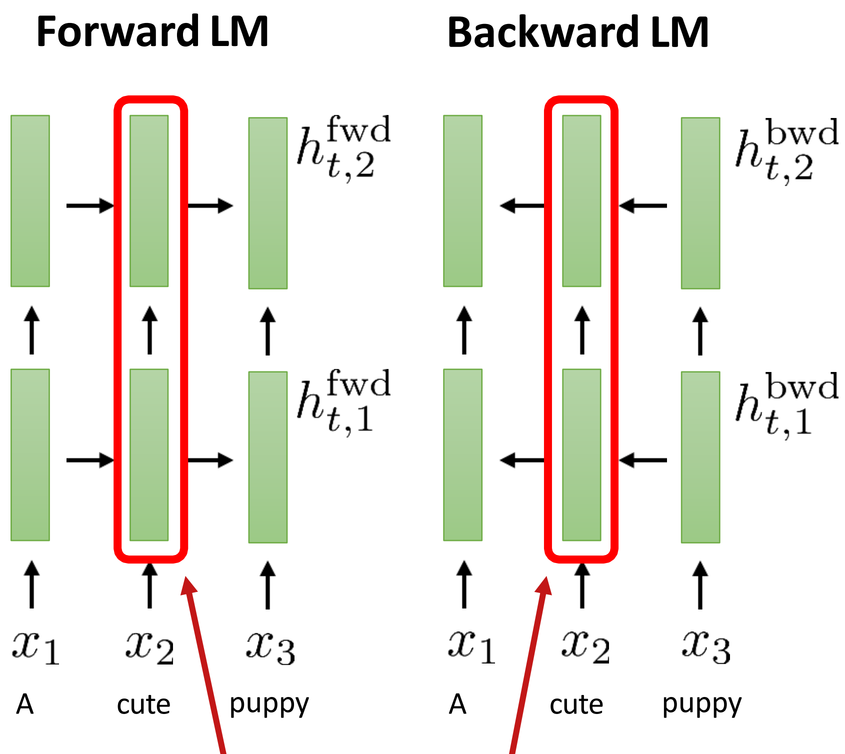
The backward LM is a deep LSTM that goes over the sequence from end to start to predict token t_k based on the suffix $t_{k+1} \ldots t_N$:

$$p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)$$

Train these LMs jointly, with the same parameters for the token representations and the softmax layer (but not for the LSTMs)

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

ELMo's Token Representation



All these hidden states, when combined, represent the word "cute."

Given a token representation $\mathbf{x}_k$, each layer j of the LSTM language models computes a vector representation $\mathbf{h}_{k,j}$ for every token k.

With L layers, ELMo represents each token as

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$ and $\mathbf{h}_{k,0}^{LM} = \mathbf{x}_k$

ELMo learns softmax weights s_j^{task} to collapse these vectors into a single vector and a task-specific scalar γ^{task}:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

$\text{simple version: } \text{ELMO}_t = [h_{t,2}^{\text{fwd}}, h_{t,2}^{\text{bwd}}]$ top layer hidden states

ELMo’s Token Representation

- The input token representations are purely **character-based**: a character CNN, followed by linear projection to reduce dimensionality

- 2048 character n-gram convolutional filters with two highway layers, followed by a linear projection to 512 dimensions”

- Advantage over using fixed embeddings: no UNK tokens, any word can be represented

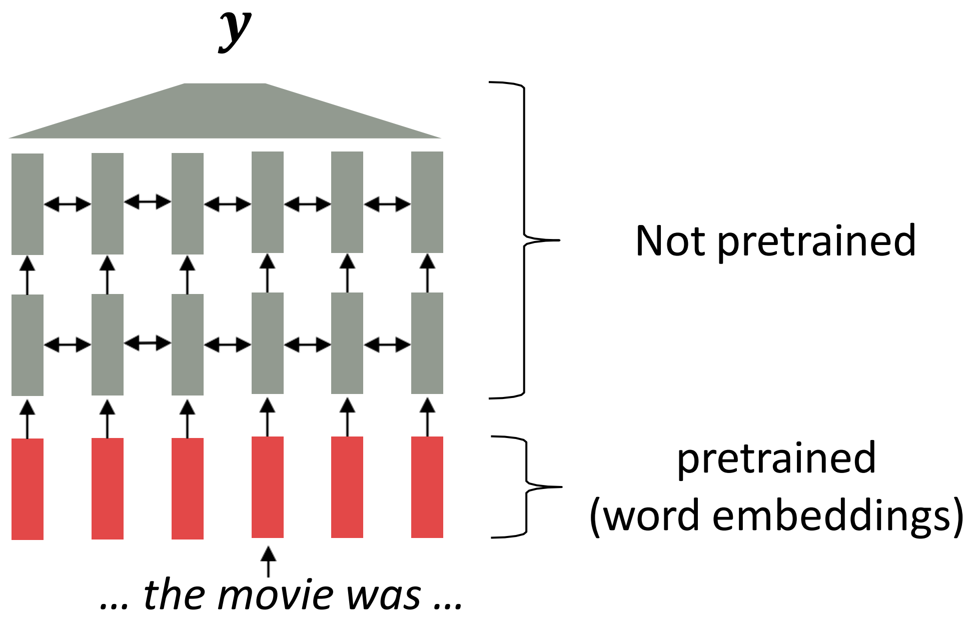
Evaluation

ELMo gave improvements on a variety of tasks:
— question answering (SQuAD)
— entailment/natural language inference (SNLI)
— semantic role labeling (SRL)
— coreference resolution (Coref)
— named entity recognition (NER)
— sentiment analysis (SST-5)

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Where We Were: Pre-trained Word Vectors

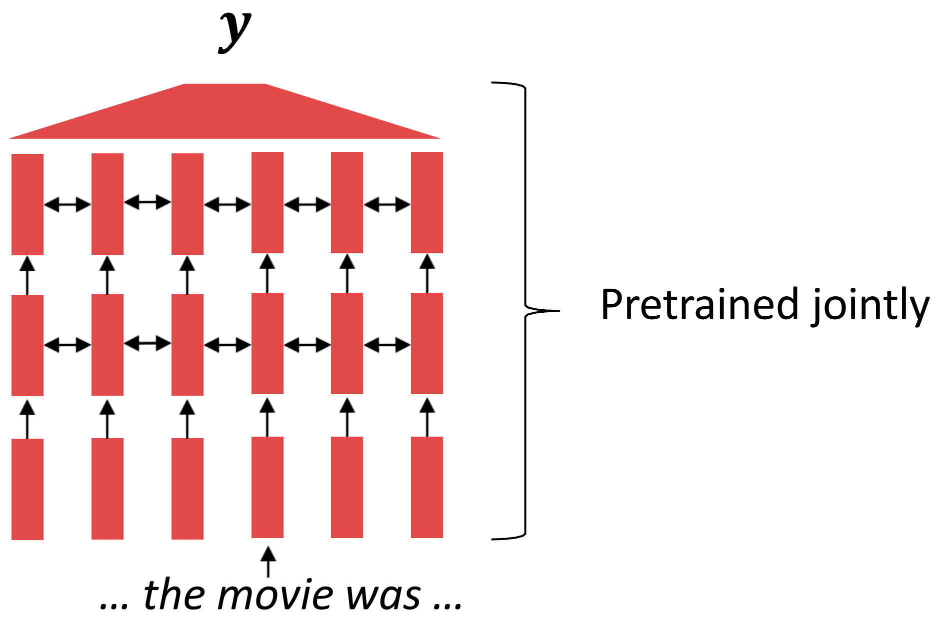
- Start with pretrained word embeddings (no context!)
- Learn how to incorporate context in an LSTM or Transformer while training on the task.
- The training data we have for our **downstream task** (like question answering) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network **are randomly initialized**!



Pre-trained Word Vectors -> Pre-trained Models

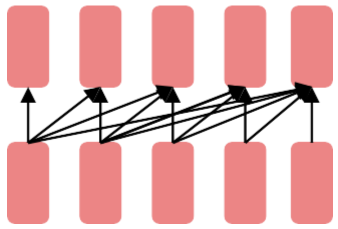
- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.

- This has been exceptionally effective at building strong:
  - **representations of language**
  - **parameter initializations** for strong NLP models.
  - **Probability distributions** over language that we can sample from



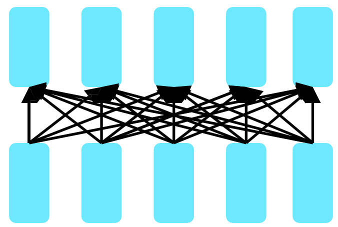
Pretraining for Three Types of Architectures

The neural architecture influences the type of pretraining, and natural use cases.



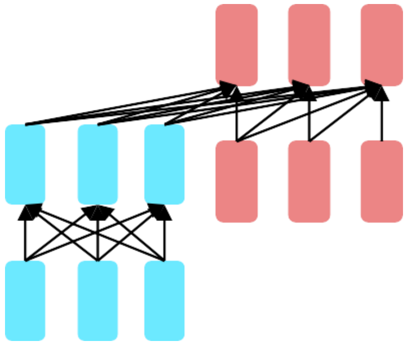
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words



Encoders

- Gets bidirectional context – can condition on future!
- How do we pretrain them?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Slides are adopted from Jacob Devlin

Background - Bidirectional Context

- Bidirectional context, unlike unidirectional context, takes into account both the left and right contexts.

Target Word

Apple is my favourite fruit and I eat it all the time.

Left Context

Right Context

Target Word

Apple is my favourite brand for buying laptop and other gadgets.

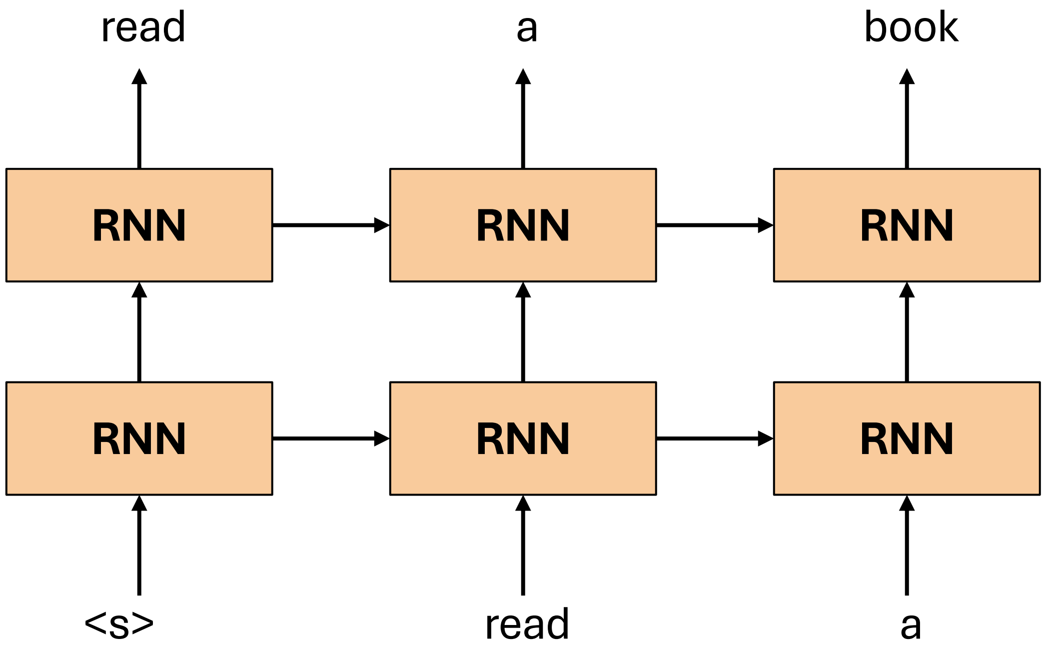
Left Context

Right Context

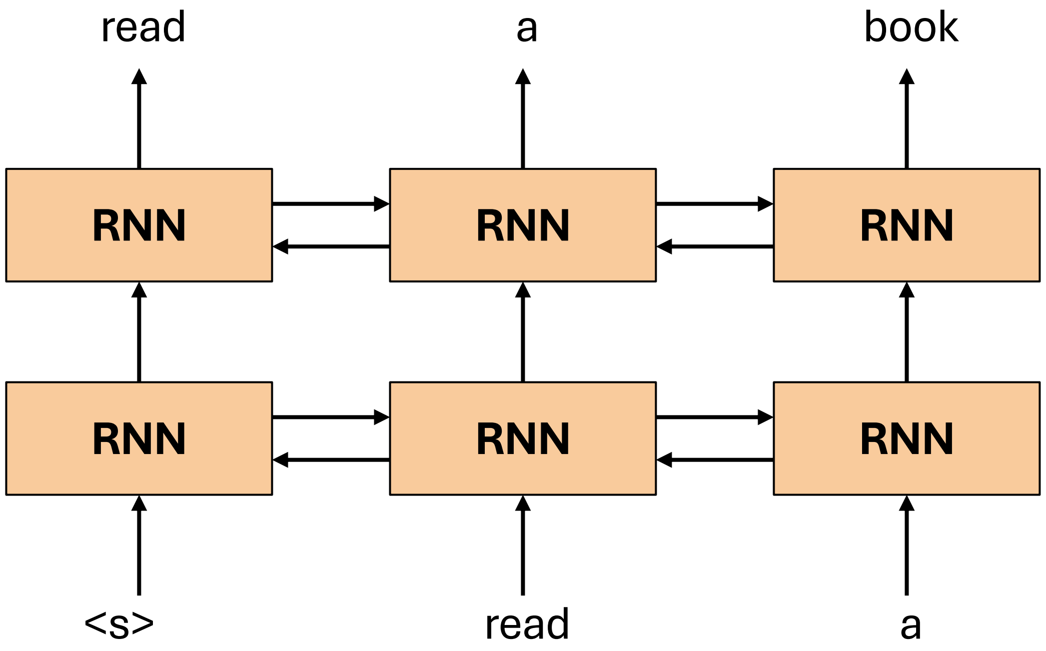
Motivation

- **Problem with previous methods:**
  - Language models only use left context or right context.
  - But language understanding is **bidirectional**.
- **Possible Issue:**
  - Directionality is needed to generate a well-formed probability distribution.
  - Words can see themselves in a bidirectional model.

Unidirectional vs. Bidirectional Models



Unidirectional



Bidirectional

Masked Language Modeling

- Mask out k% of the input words, and then predict the masked words (Usually k = 15%). Example :

I like going to the [MASK] in the evening
↓
park

  - Too little masking: Too expensive to train
  - Too much masking: Not enough context
- The model needs to predict 15% of the words, but we don't replace with [MASK] 100% of the time. Instead:
  - 80% of the time, **replace with [MASK]**
    - Example : like going to the park → like going to the [MASK]
  - 10% of the time, **replace random word**
    - Example : like going to the park → like going to the store
  - 10% of the time, **keep same**
    - Example : like going to the park → like going to the park

Next Sentence Prediction

- To learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence.

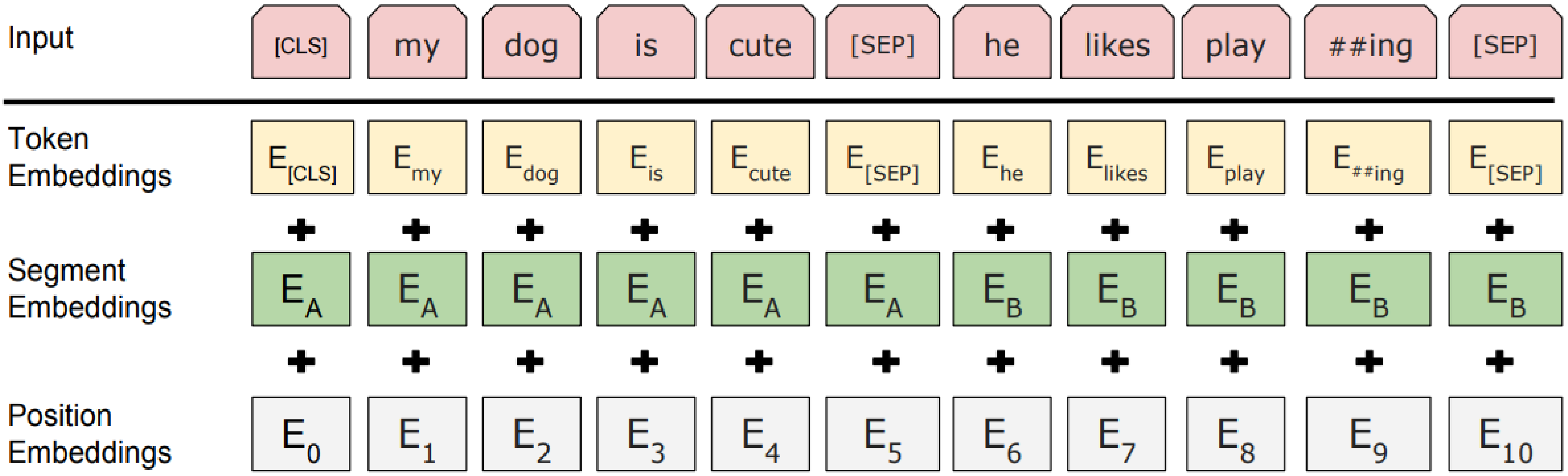
Input = [CLS] I enjoy read [MASK] book ##s [SEP]
I finish ##ed a [MASK] novel [SEP]
Label = IsNext

Input = [CLS] I enjoy read ##ing book [MASK] [SEP]
The dog ran [MASK] the street [SEP]
Label = NotNext

- Important for many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI)
- How to choose sentences A and B for pretraining?
  - 50% of the time B is the actual next sentence that follows A (labeled as IsNext)
  - 50% of the time it is a random sentence from the corpus (labeled as NotNext)

Input Representation

- Use 30,000 WordPiece vocabulary on input.
- For a given token, its input representation is constructed by summing the token embeddings, the segmentation embeddings and the position embeddings.

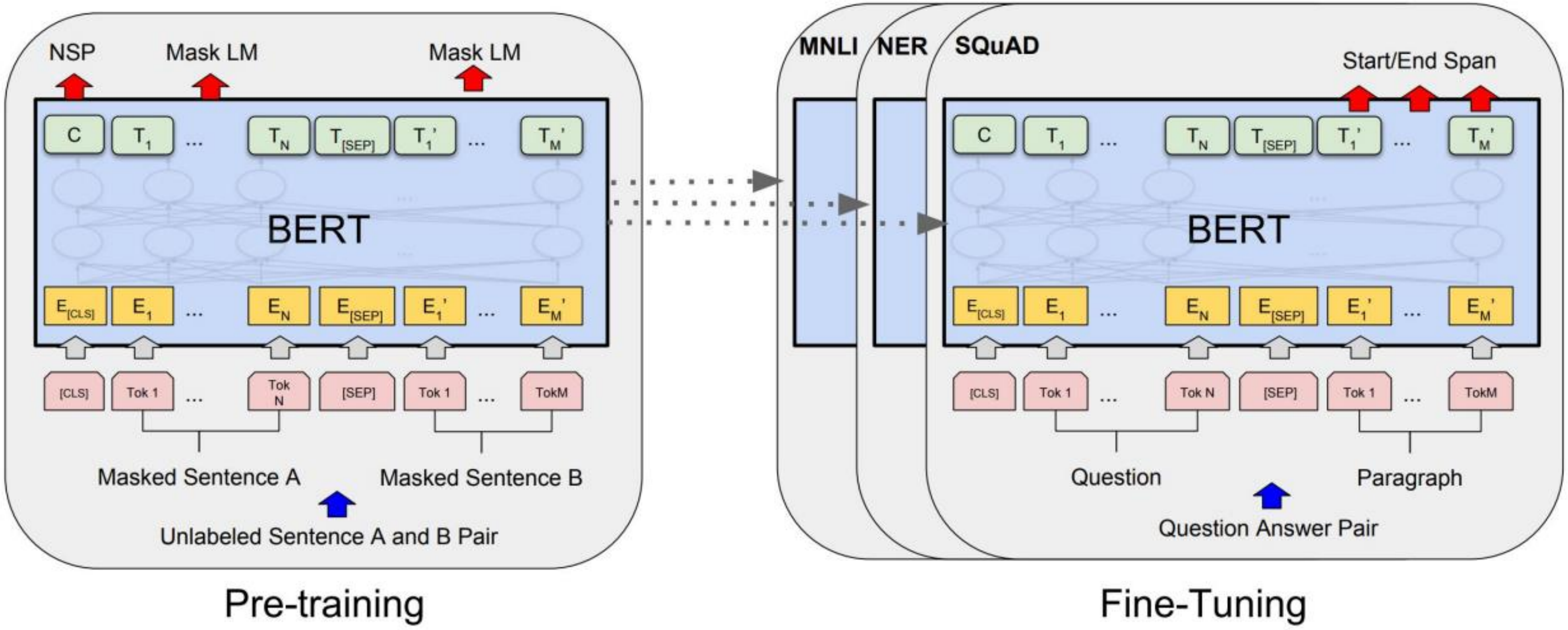


Source of Image : BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NAACL 2019)

Training Details

- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- BERT-Base: 12-layer, 768-hidden, 12-head
- BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

Fine-Tuning Procedure



QA Task based Fine-tuning

QA Task based Fine-tuning

QA Task based Fine-tuning

QA Task based Fine-tuning

QA Task based Fine-tuning

BERT: Evaluation

BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**