New directions in word representations*

Ponomarev Vyacheslav
HSE
2019

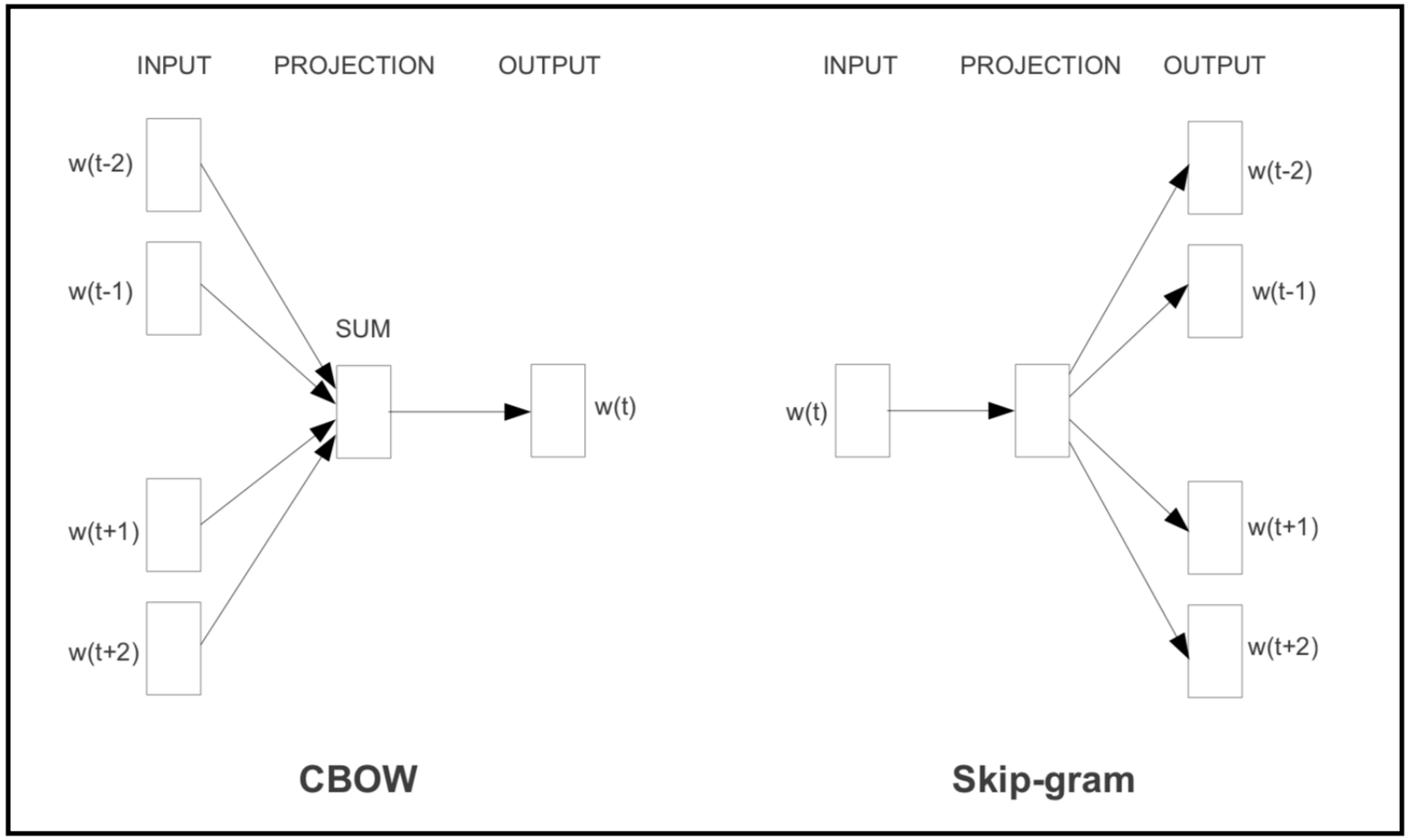
*And natural language understanding tasks

Plan

- Old-school embeddings: word2vec, GloVe, FastText
- (New!) Deep contextualized embeddings
- (New!) Pre-training and fine-tuning in NLP tasks
- (Top!) Pre-training deep bidirectional transformers

First approaches in word representations

- word2vec - Tomas Mikolov et al. 2013



Source: Efficient Estimation of Word Representations in Vector Space [Mikolov et al.,2013]

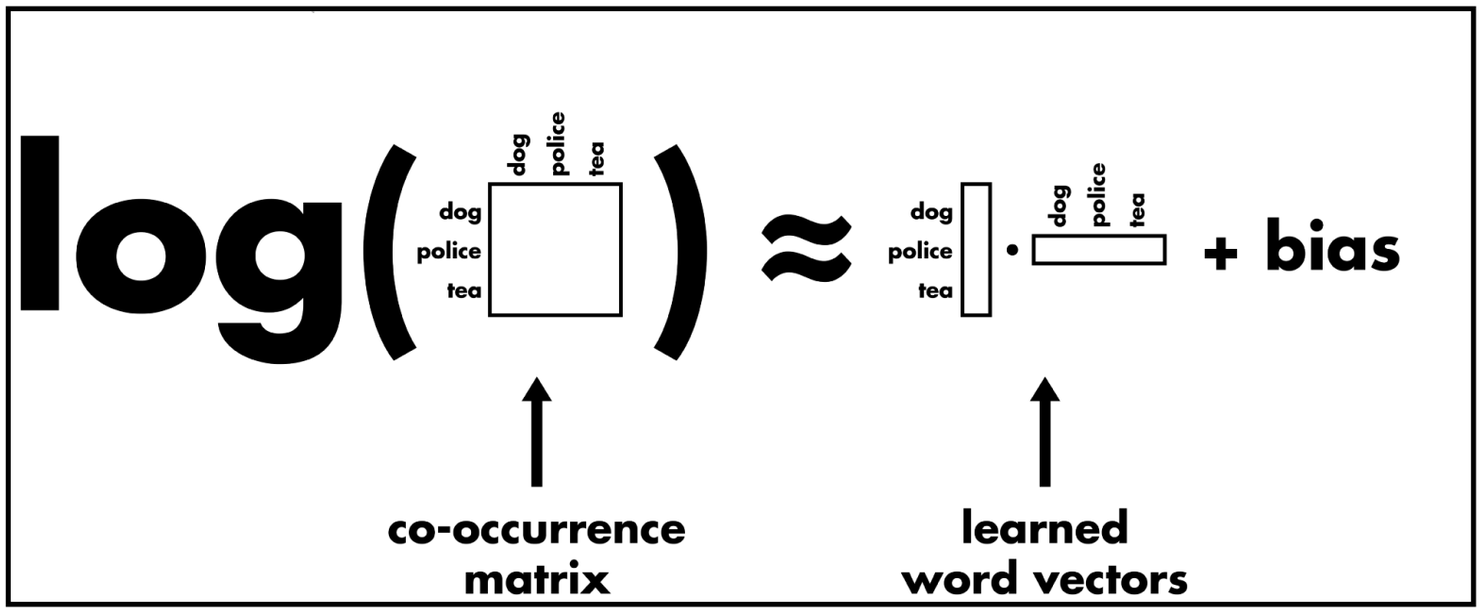
First approaches in word representations

- word2vec - Tomas Mikolov et al. 2013

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Source: https://nlp.stanford.edu/projects/glove/

- GloVe - Jeffrey Pennington et al. 2014



Source: https://towardsdatascience.com/emnlp-what-is-glove-part-v-fa888272c290

First approaches in word representations

- word2vec - Tomas Mikolov et al. 2013
- GloVe - Jeffrey Pennington et al. 2014
- fastText - Piotr Bojanowski, Edouard Grave et al. 2017

FastText =
SkipGram model + Subword model (n-grams)

First approaches in word representations: What is the problem?

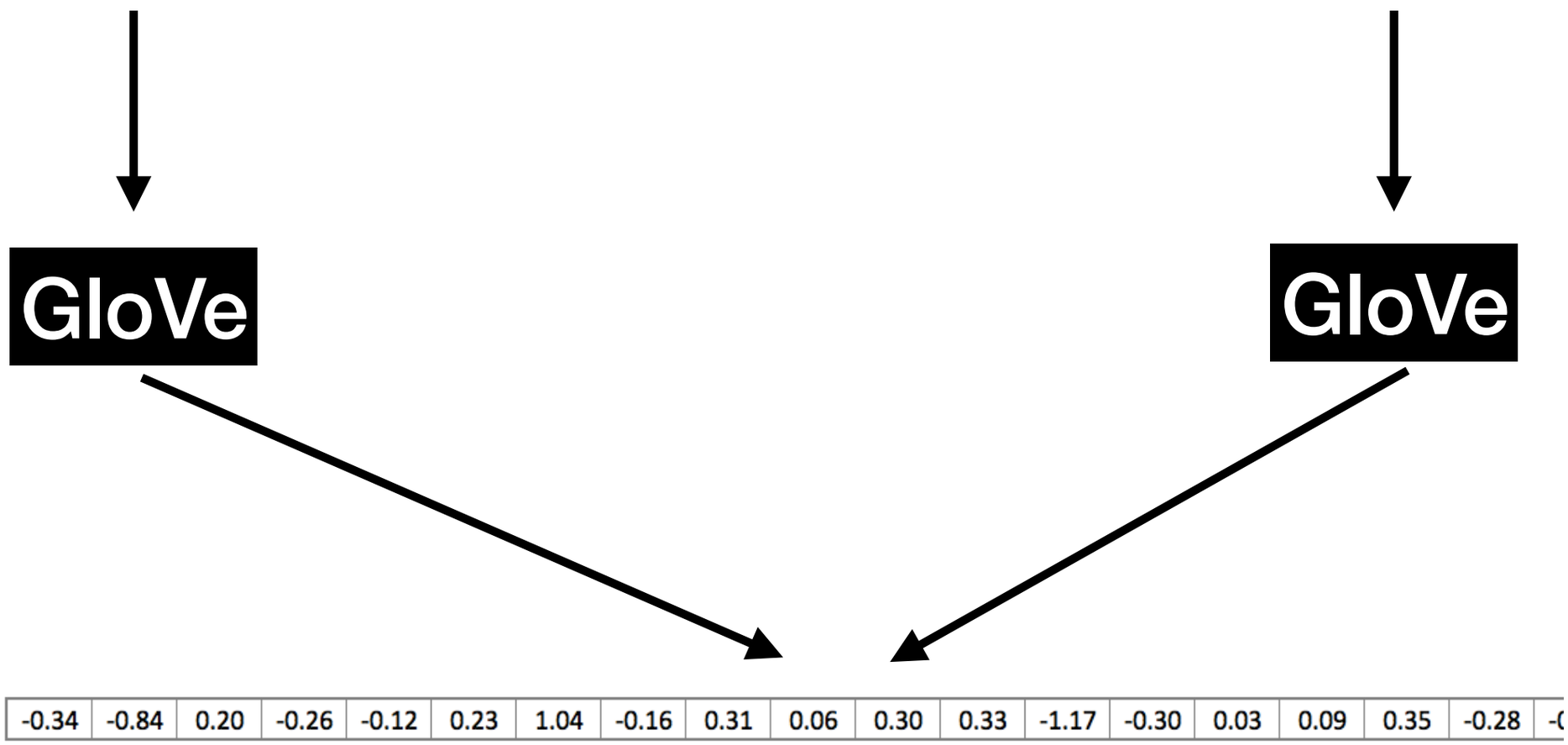
Challenges for word embeddings:

1) To model **complex characteristics** of word use (syntax, semantics)
2) To model how these uses vary across linguistic contexts == To model **polysemy**

First approaches in word representations: What is the problem?

*«Let's **Stick** Together»*

*«I'm going to purchase myself a walking **stick**»*

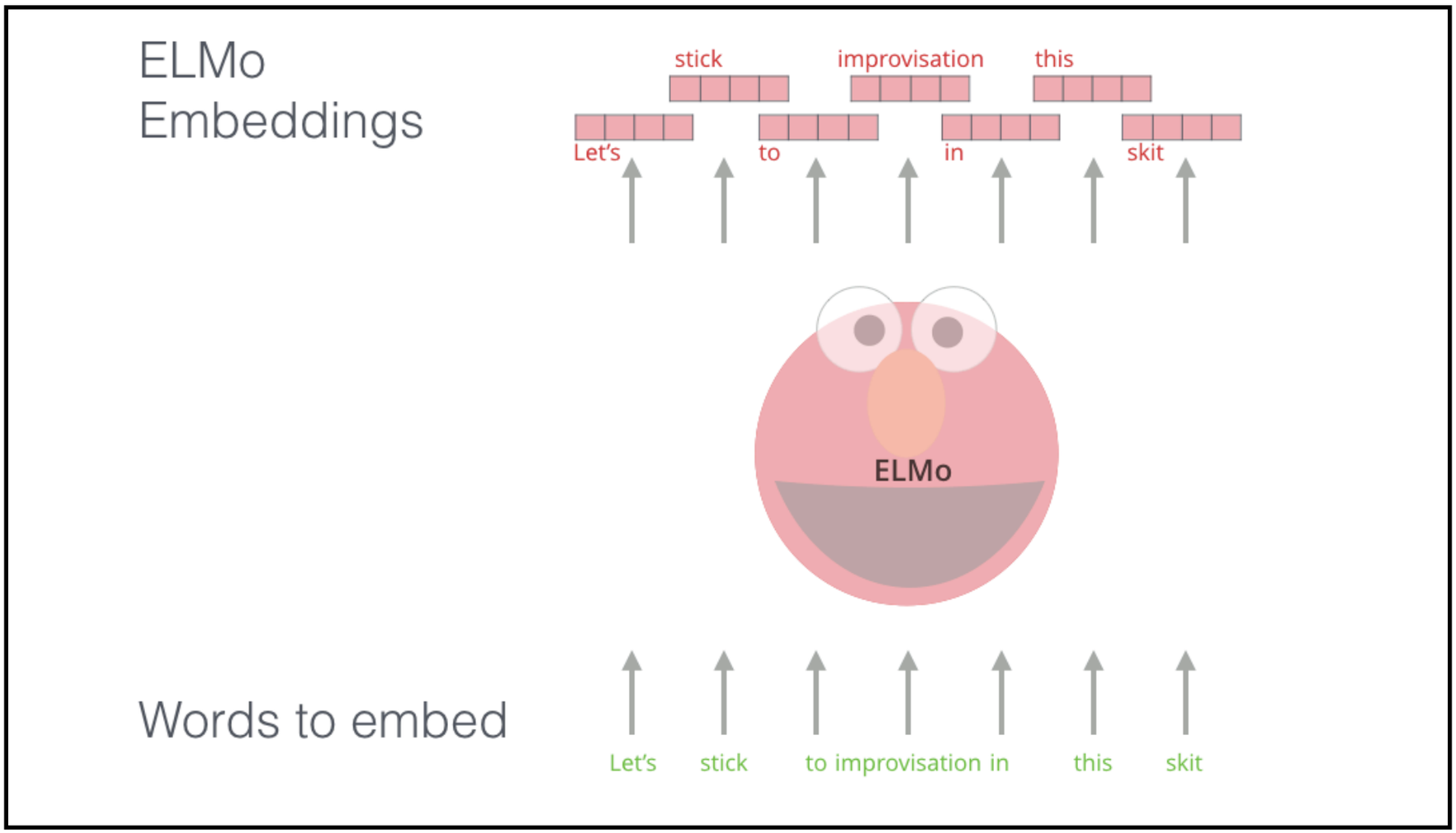


ELMo: Context matters

Source: https://jalammar.github.io/illustrated-bert/

ELMo: Embeddings from Language Models

1. Representation is a **function** of the **entire input** sentence.
2. Representations are **deep** - they are a **function** of **all** of the internal **layers** of the underlying model.



Source: https://jalammar.github.io/illustrated-bert/

ELMo: Embeddings from Language Models

Forward LM:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$$

Backward LM:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N)$$

* LM:

x_k^{LM} – context-independent token representation (Token embedding or Char-CNN)

LSTM – L layers

$\overrightarrow{h}_{k,j}^{LM}$ – Hidden state at position k on level j, j=1…L

Softmax

ELMo: Embeddings from Language Models

Objective in bidirectional LM:

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

* LM:

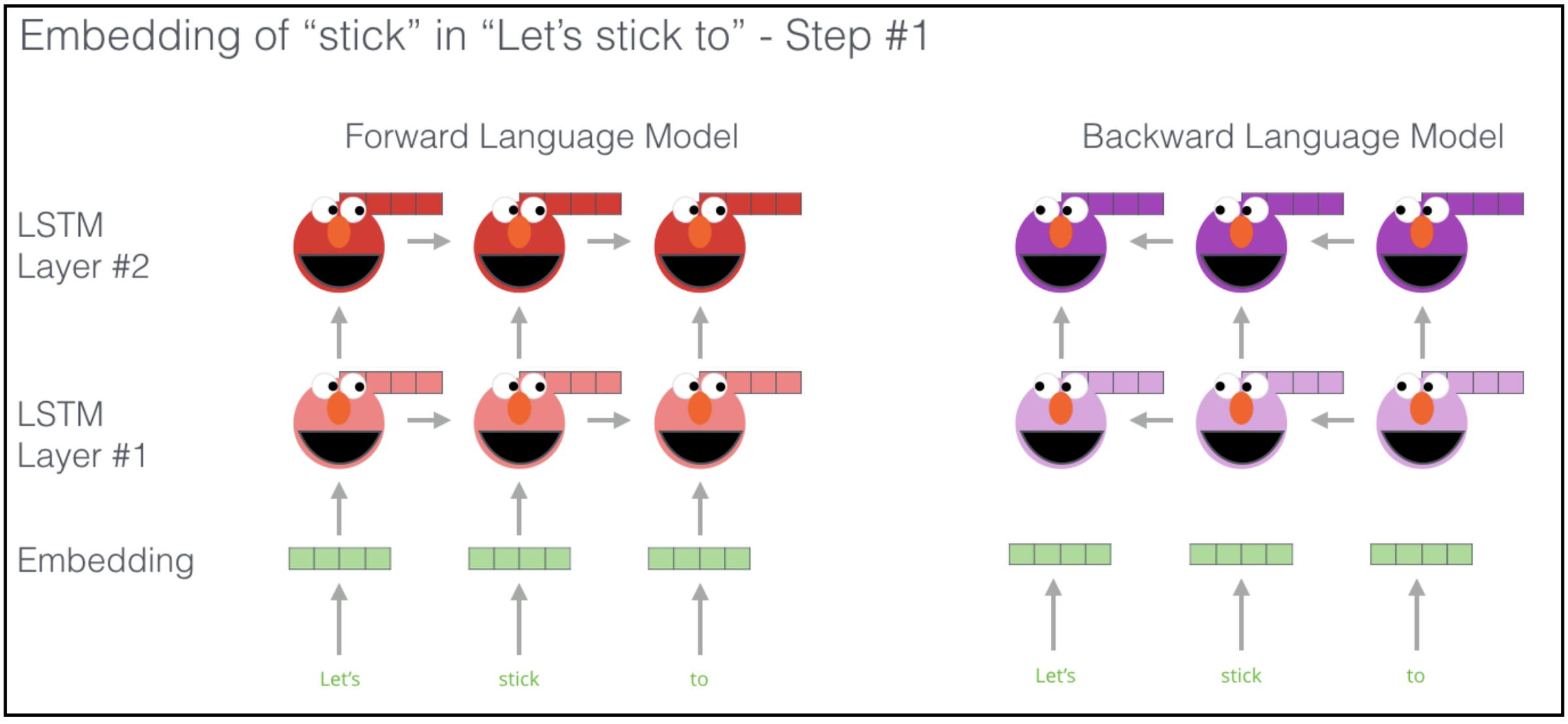
x_k^{LM} – context-independent token representation (Token embedding or Char-CNN)

LSTM – L layers

$\overrightarrow{h}_{k,j}^{LM}$ – Hidden state at position k on level j, j=1…L

Softmax

ELMo: Embeddings from Language Models



Source: https://jalammar.github.io/illustrated-bert/

ELMo: Embeddings from Language Models

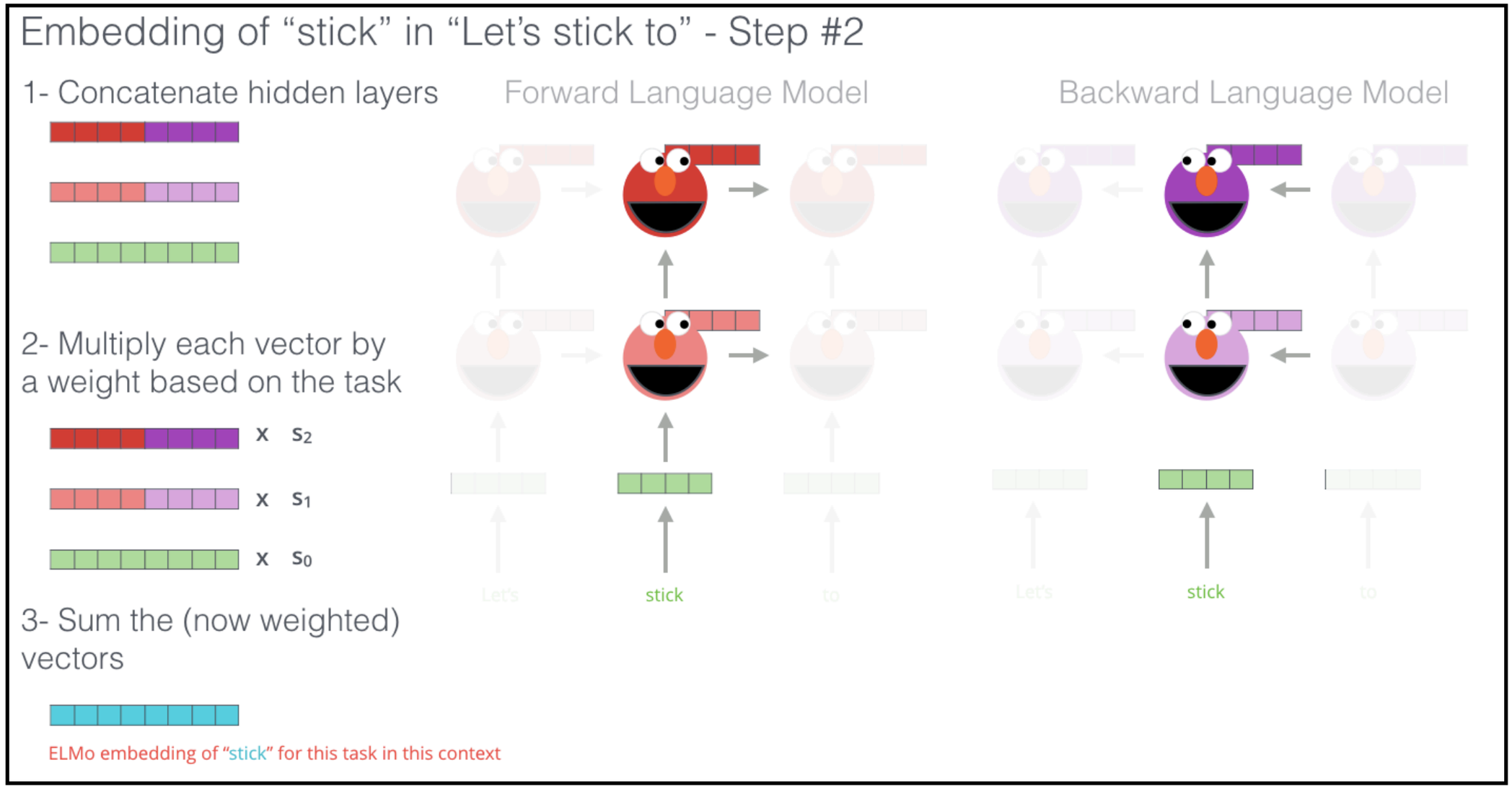
For each token t_k , L-layer biLM computes a set of 2L + 1 representations:

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

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

ELMo: Embeddings from Language Models

Source: https://jalammar.github.io/illustrated-bert/



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

ELMo: Embeddings from Language Models

Model Architecture:

- 2 bi-LSTM layers with 4096 units, 512 dimension projections
- Char-CNN with 2048 filters, linear projection down to a 512 dimension
- Applying layer normalization
- Apply dropout and (optionally) weight decay
- 10 epochs on 1B Word Benchmark

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%

Source: Deep contextualized word representations [Matthew Peters et al., 2018]

The Transformer: Going beyond LSTMs

Transformer == replacement for LSTM **?**

Perfect for machine translation **+**

Better deal with long-term dependencies **+**

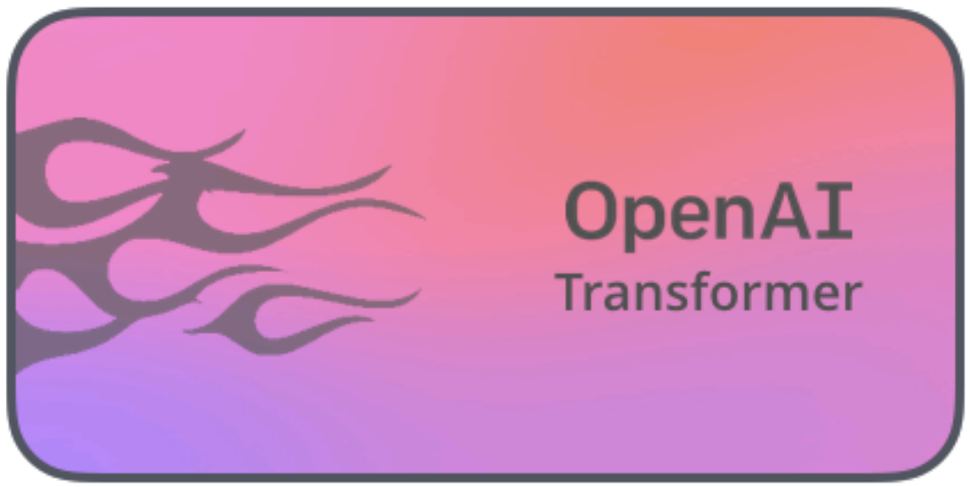
How to pre-train and fine-tune on downstream task **?**

Is there a way to do transfer learning like in CV **?**

OpenAI Transformer

We don’t need entire Transformer! Just use the decoder.

The decoder is a natural choice for predicting the next word since it’s built to mask future tokens



Source: https://jalammar.github.io/illustrated-bert/

OpenAI Transformer: Stage 1 (unsupervised pre-training)

Standard LM objective:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Transformer decoder:

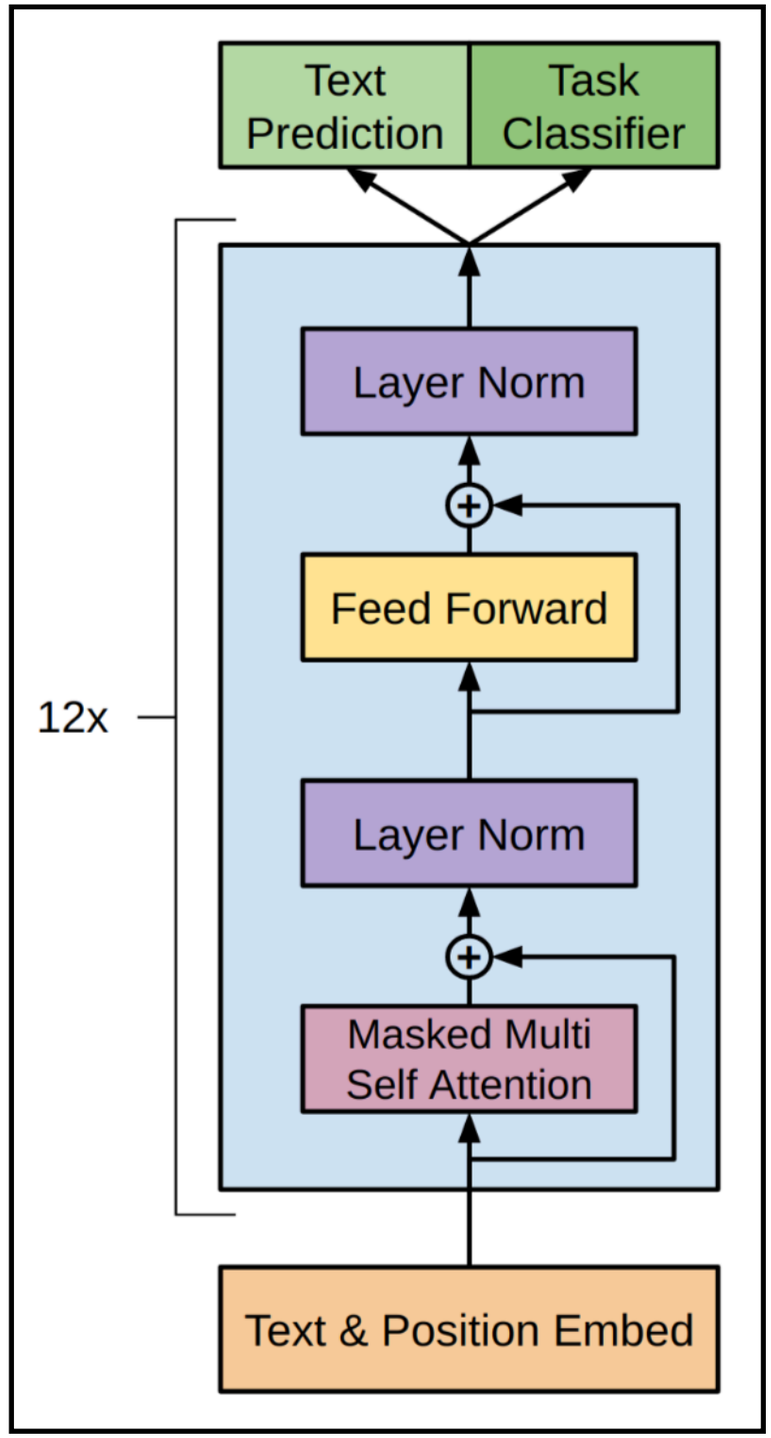
$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$U = (u_{-k}, \ldots, u_{-1})$ – context vector of tokens

n – number of layers

W_e – token embedding matrix

W_p – position embedding matrix



OpenAI Transformer: Stage 2 (supervised fine-tuning)

$$\text{Dataset } \mathcal{C} : \{(x_i^1, \ldots, x_i^m), y_i\}_{i=1}^n$$

h_l^m – final transformer block's activation

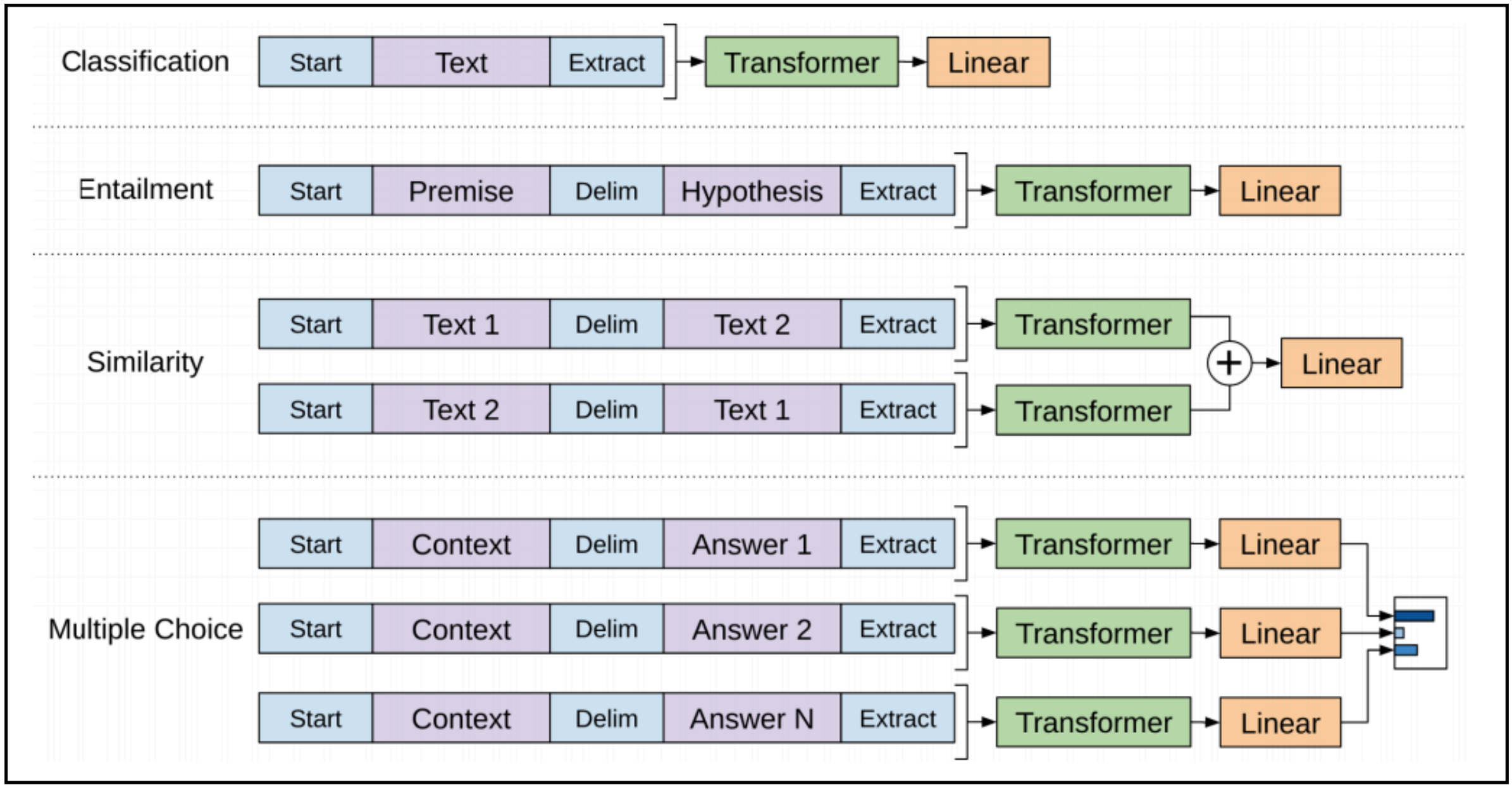
W_y – learnable prediction matrix

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y)$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

$$\boxed{L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})}$$

OpenAI Transformer: Stage 2.5 (Task-specific input transformations)



Source: Improving language understanding with unsupervised learning [A. Radford et al., 2018]

OpenAI Transformer: results

Long story short: up to 10% performance increase

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Source: Improving language understanding with unsupervised learning [A. Radford et al., 2018]

OpenAI Transformer: results

Source: https://blog.openai.com/language-unsupervised/

Can we do better?

ELMo : Bidirectional language model

OpenAI : Transformer decoder == forward language model

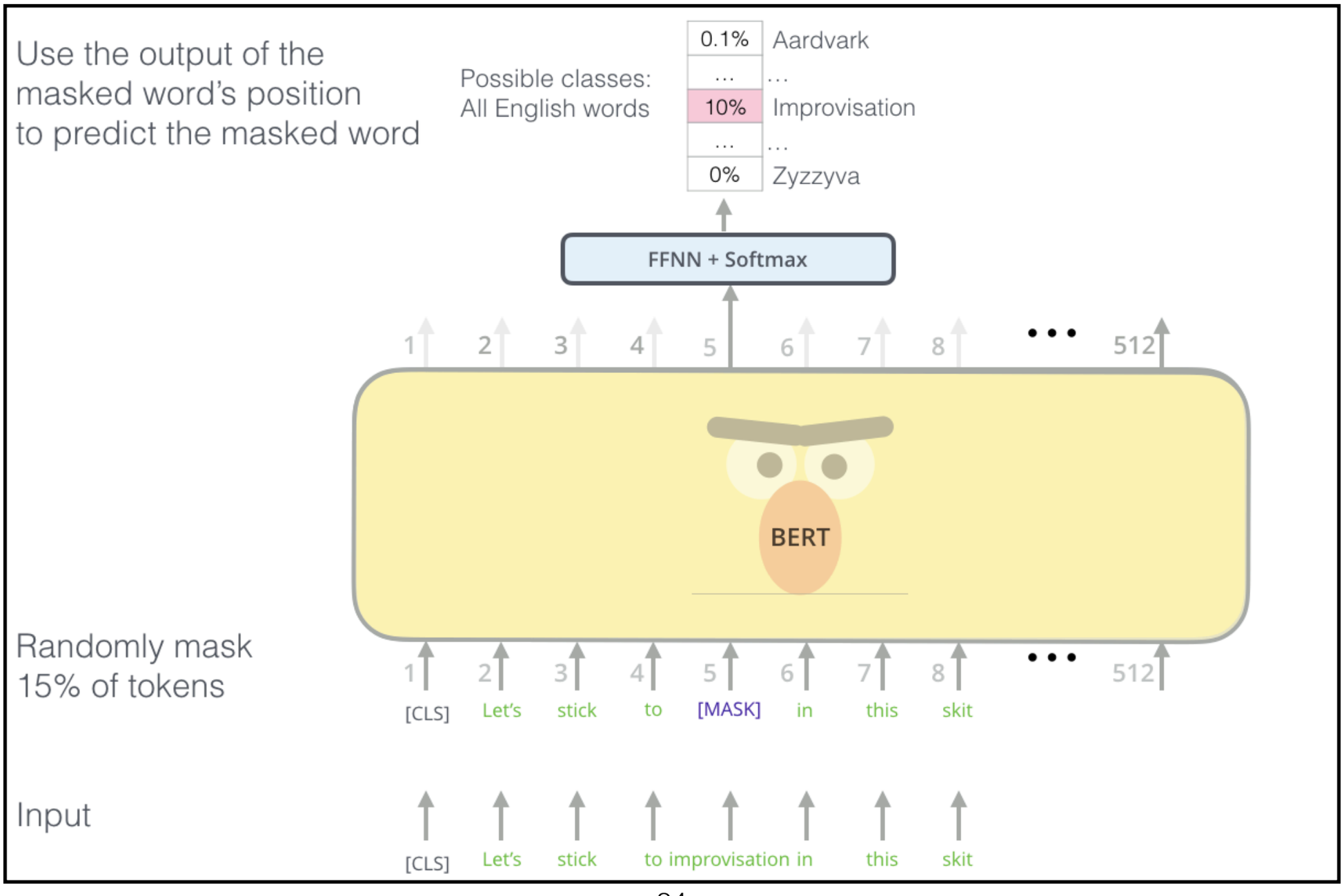
?!

Could we build a transformer-based model whose language model is conditioned on both left and right context?

BERT: Bidirectional Encoder Representations from Transformers

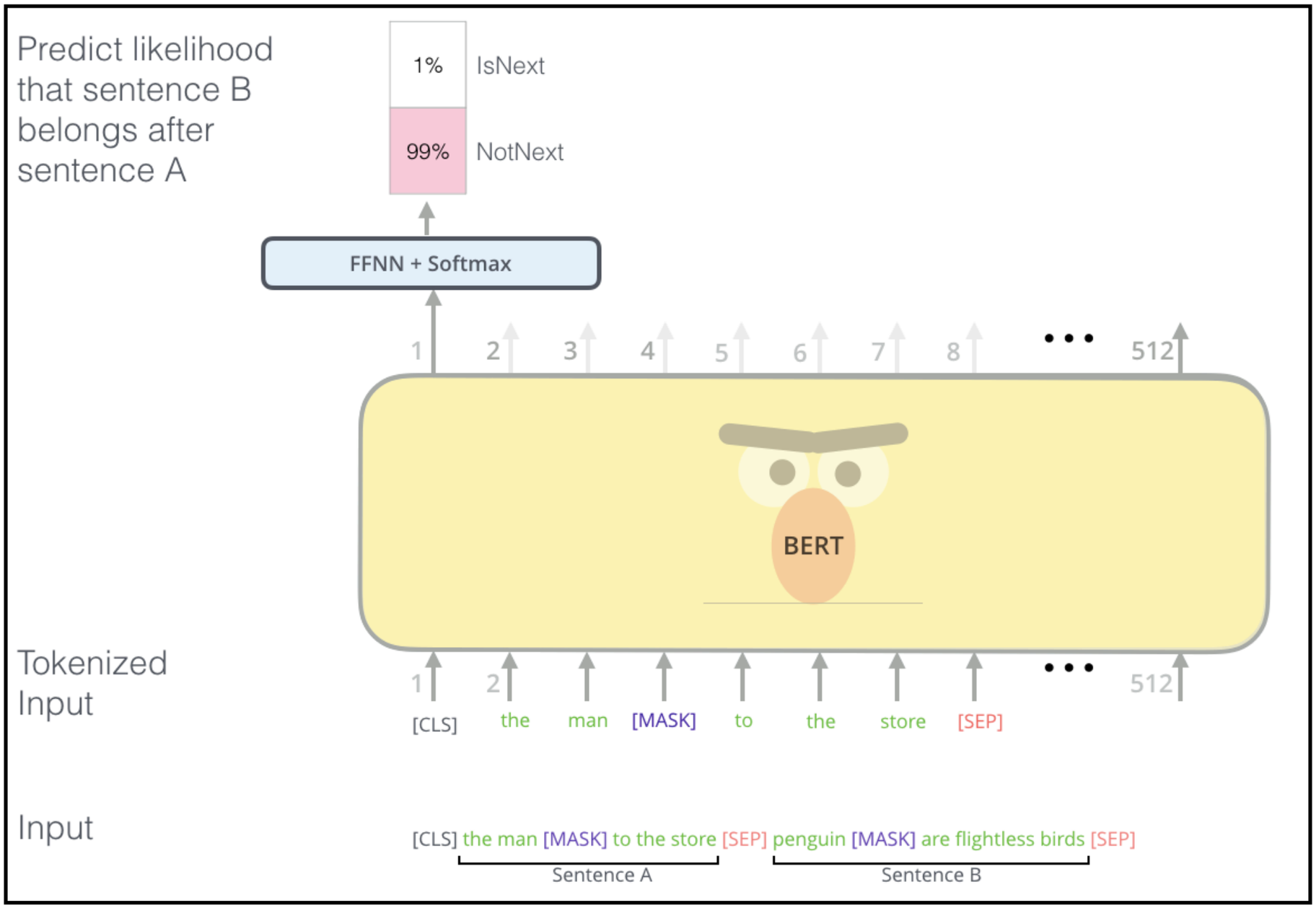
Task #1: Masked LM

Source: https://jalammar.github.io/illustrated-bert/



BERT: Bidirectional Encoder Representations from Transformers

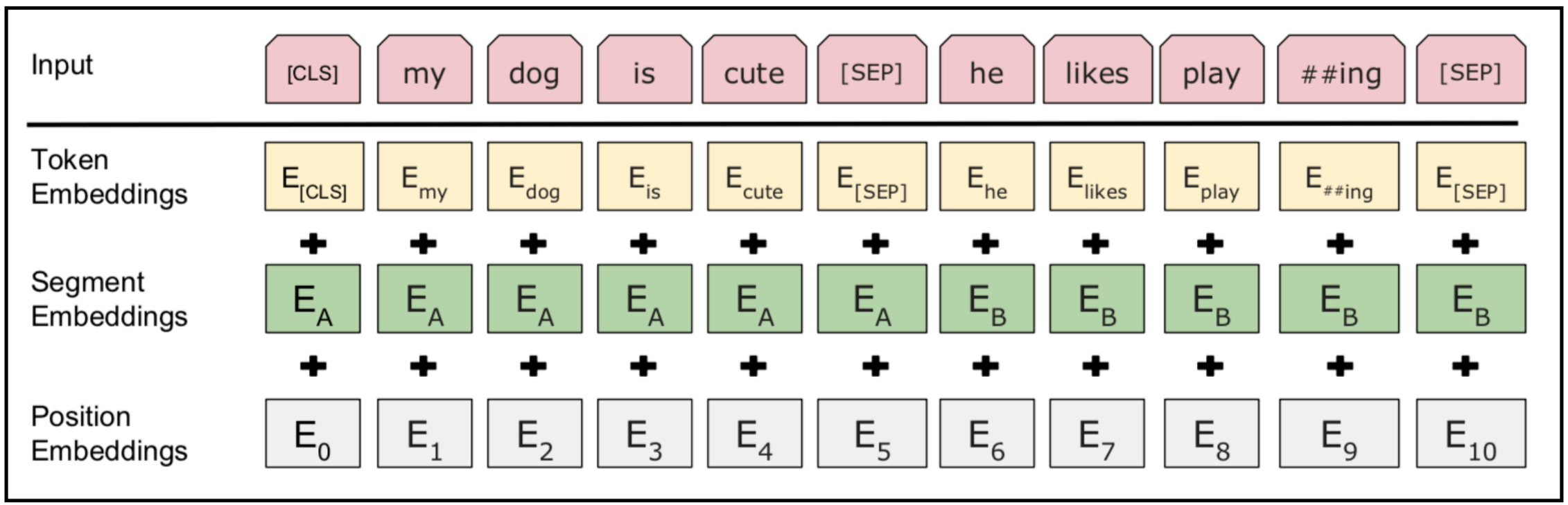
Task #2: Next Sentence Prediction



Source: https://jalammar.github.io/illustrated-bert/

BERT: Bidirectional Encoder Representations from Transformers

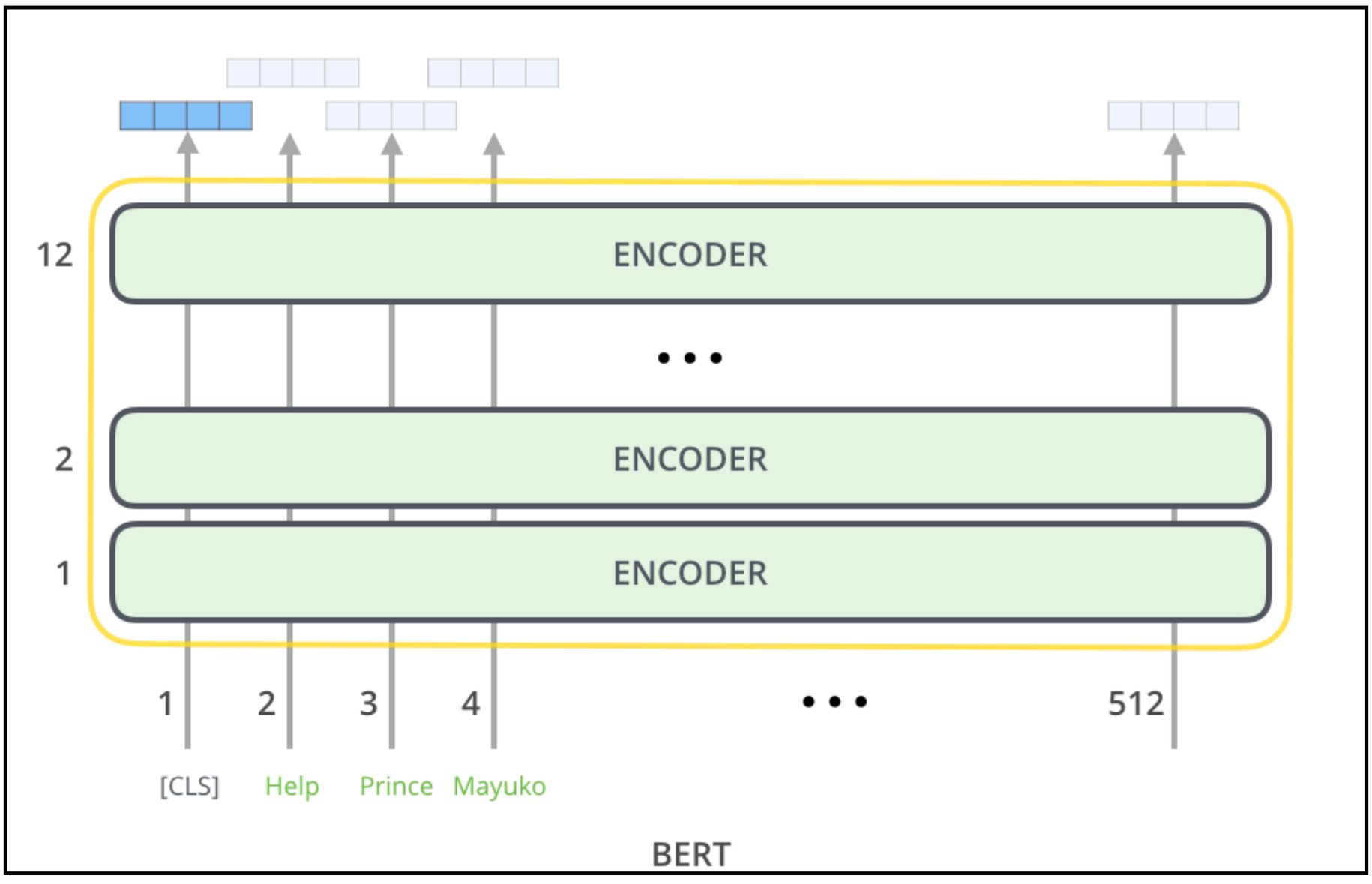
Input Representation



Source: Bert: Pre-training of deep bidirectional transformers for language understanding. [Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

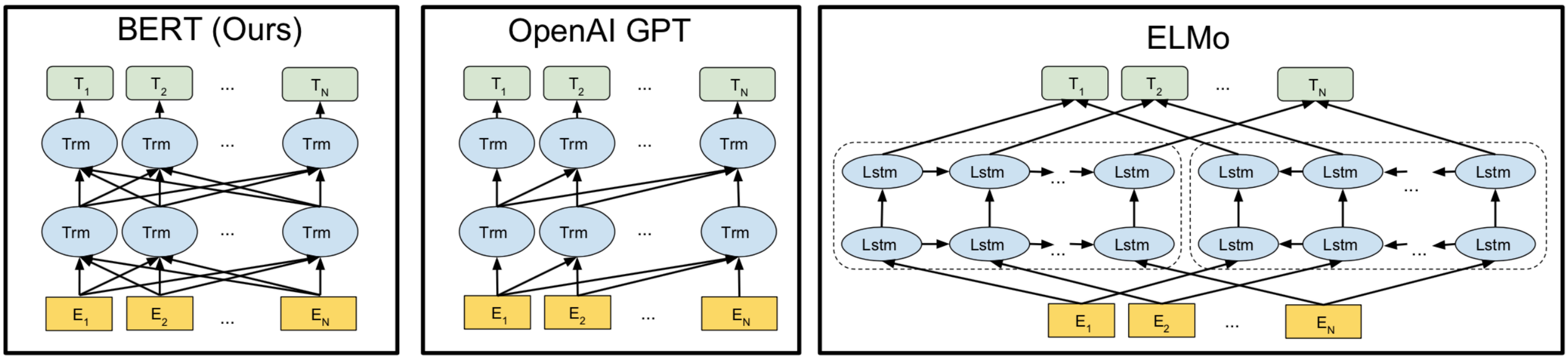
Model (Base)



Source: https://jalammar.github.io/illustrated-bert/

BERT: Bidirectional Encoder Representations from Transformers

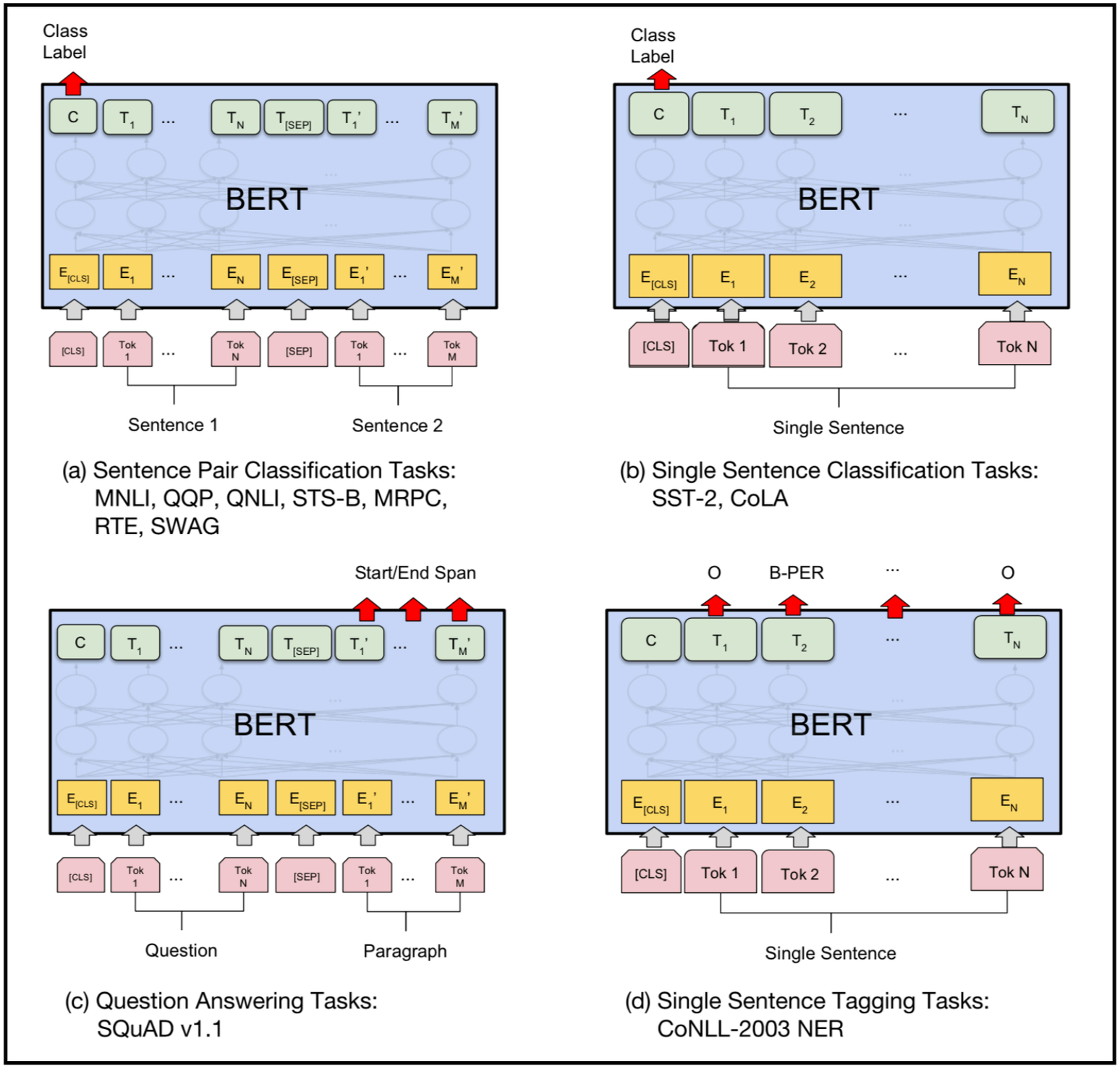
Comparison with previous approaches



Source: Bert: Pre-training of deep bidirectional transformers for language understanding. [Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

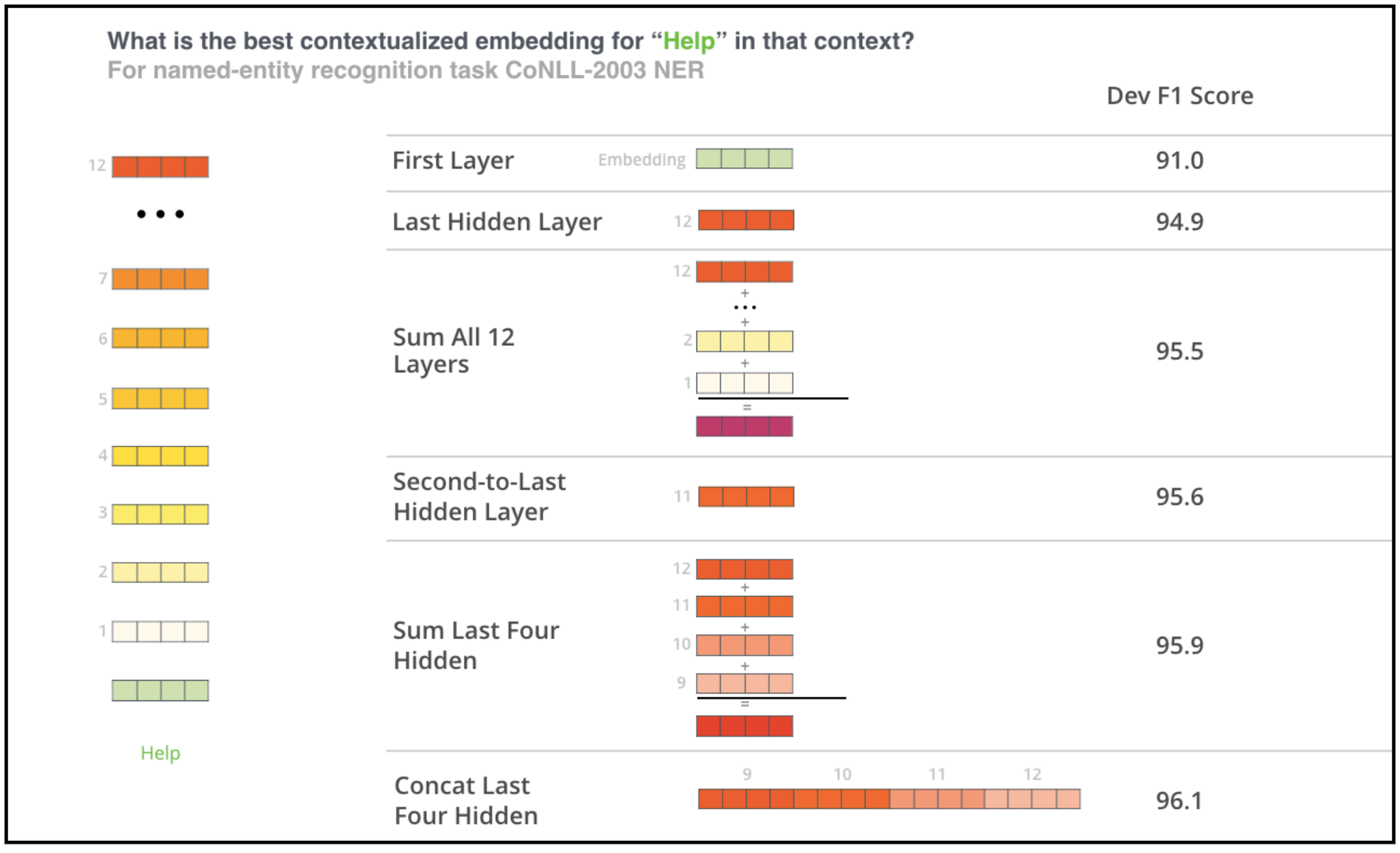
Fine-tuning



Source: Bert: Pre-training of deep bidirectional transformers for language understanding. [Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

Feature-based approach



Source: https://jalammar.github.io/illustrated-bert/

BERT: Bidirectional Encoder Representations from Transformers

GLUE Benchmark

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Source: Bert: Pre-training of deep bidirectional transformers for language understanding. [Devlin et al., 2018]

Summary

- Deep contextualized embeddings boost performance when compared to conventional ones.
- Unsupervised pre-training in NLP tasks is highly beneficial.
- BERT is current state-of-the-art, applicable to variety of tasks without architecture changes.
- NLP's ImageNet moment has arrived

References

1. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality.

2. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

3. Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

4. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

5. Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

6. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) // URL: https://jalammar.github.io/illustrated-bert/