# BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Bidirectional Encoder Representations from Transformers)

**Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, *ACL'19***

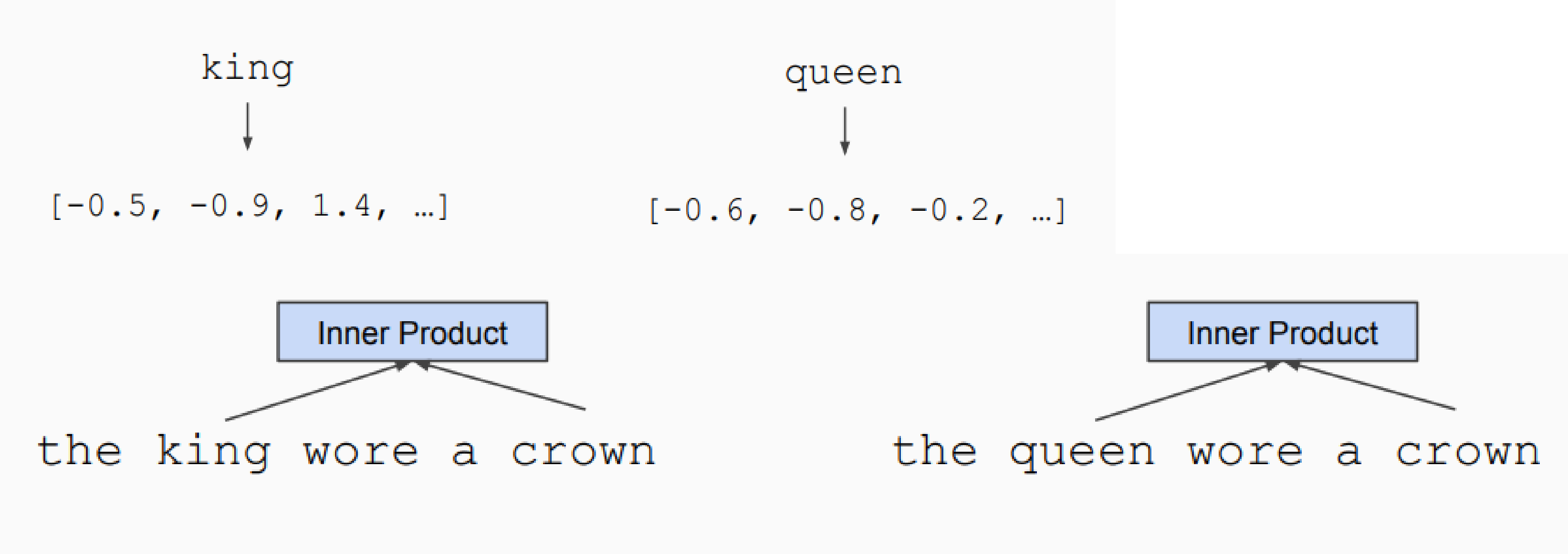
***Presented by – Md. Mahadi Hassan***

# Overview

- Like transfer learning is used in vision, pretrained model will enable NLP tasks to have a basic understanding about the language and then fine tune the model for specific tasks.
- They define two tasks for pretraining: masked language model(MLM) and next sentence prediction(NSP)
- They showed tuning the BERT model for 11 different tasks and showed that it gives best result in all of them.

# Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP
- Word embeddings (word2vec, GloVe) are often pre-trained on text corpus from co-occurrence statistics



# Contextual Representations

» **Problem**: Word embeddings are applied in a context free manner

```
open a bank account          on the river bank

            [0.3, 0.2, -0.8, …]
```

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

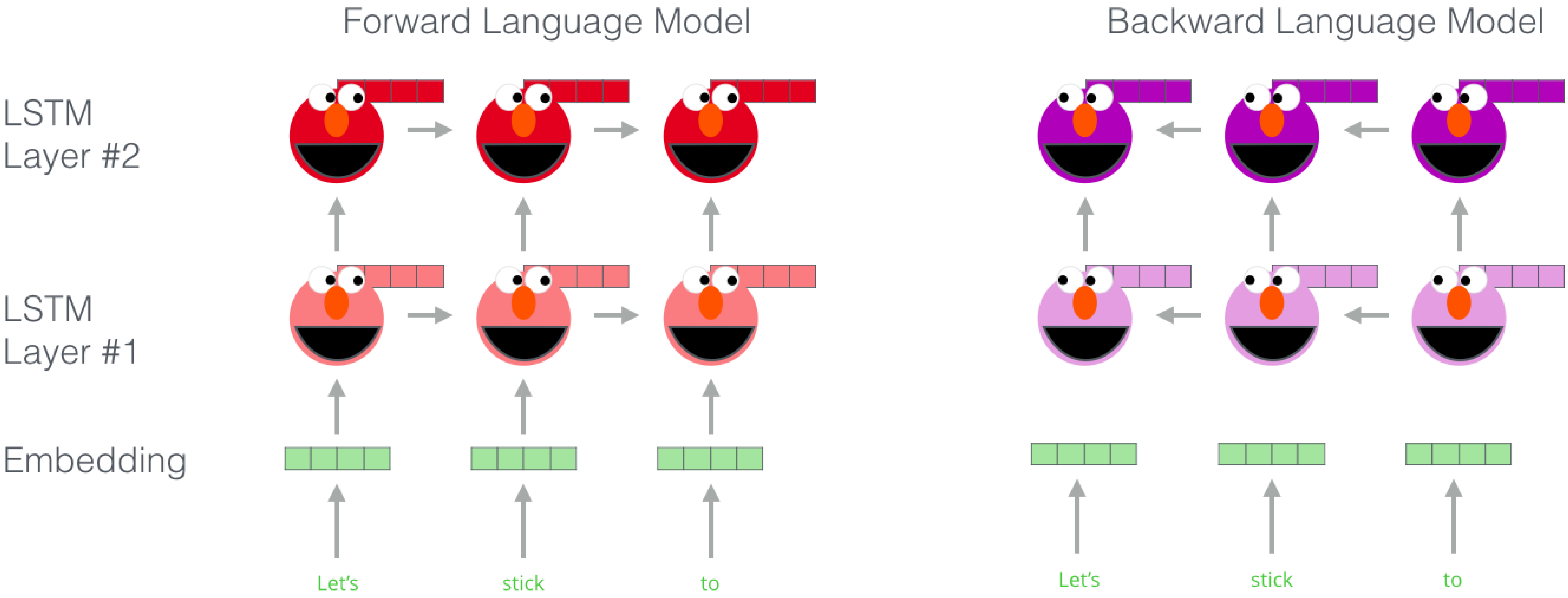
```
[0.9, -0.2, 1.6, …]          [-1.9, -0.4, 0.1, …]

open a bank account          on the river bank
```

# Elmo: Context Matters

- ELMo gained its language understanding from being trained to predict the next word in a sequence of words.
- trains a bi-directional LSTM – so that its language model doesn’t only have a sense of the next word, but also the previous word.

# Embedding of “stick” in “Let’s stick to” - Step #1



# Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

2- Multiply each vector by a weight based on the task

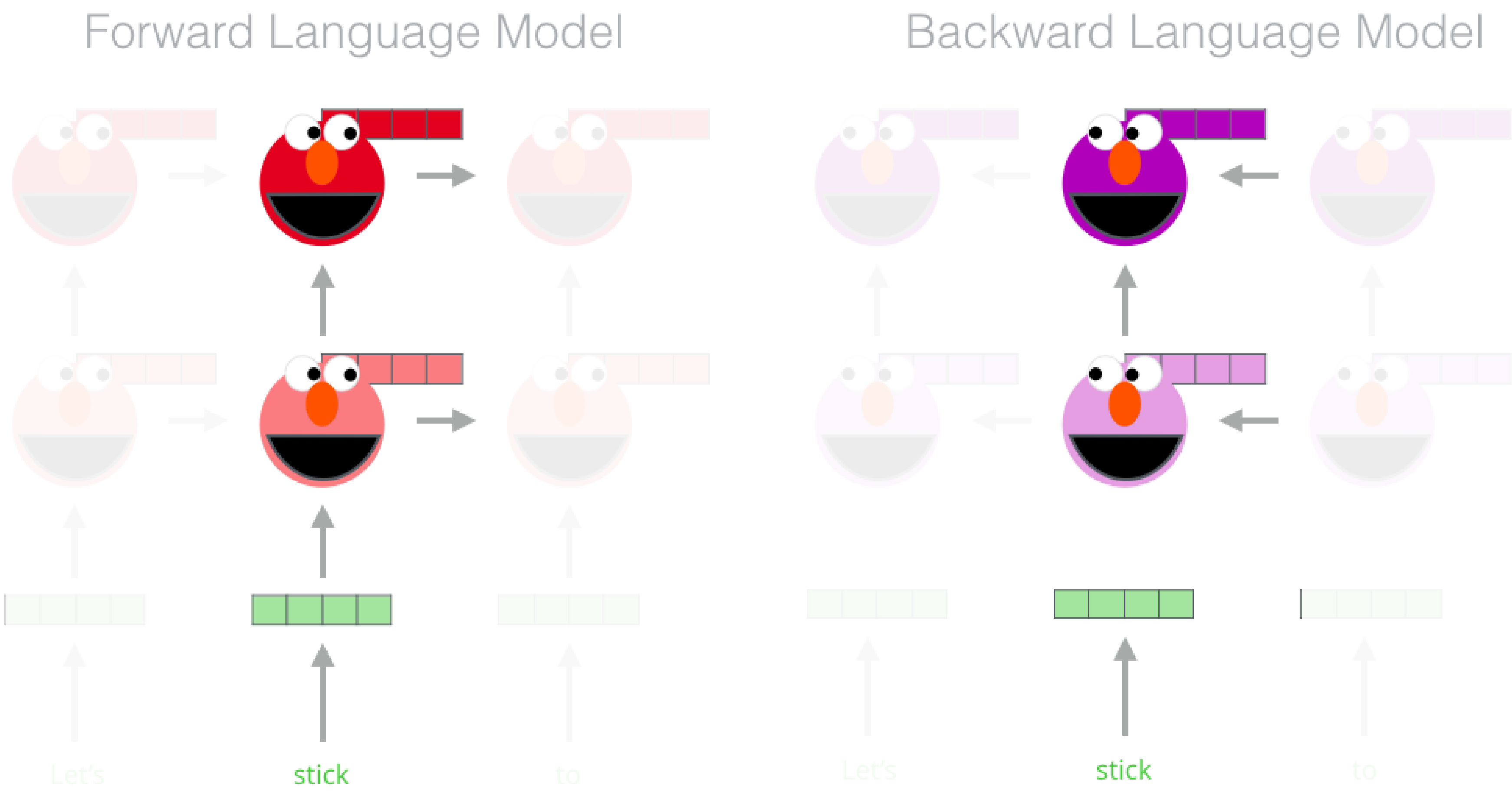
x $s_2$

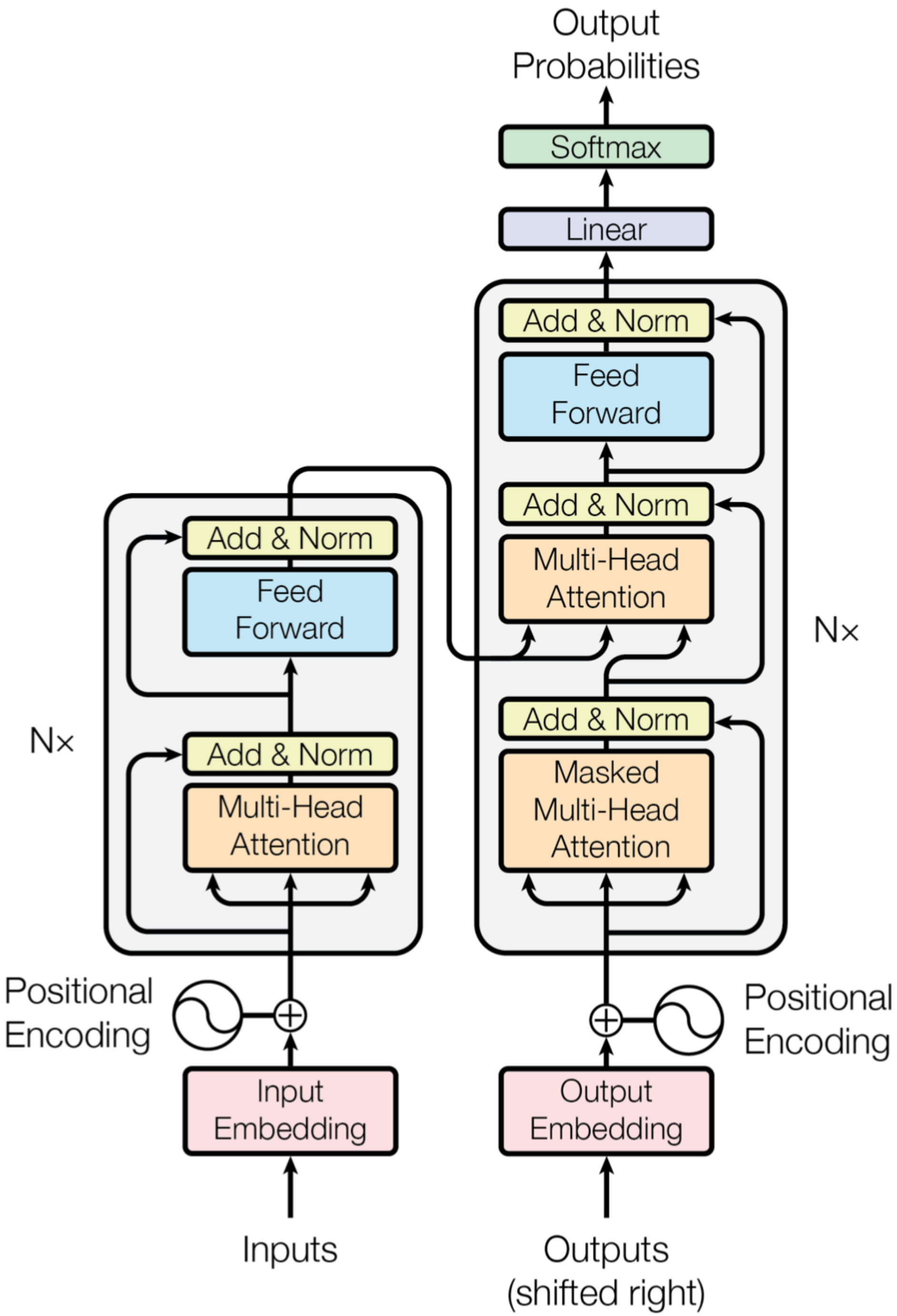
x $s_1$

x $s_0$

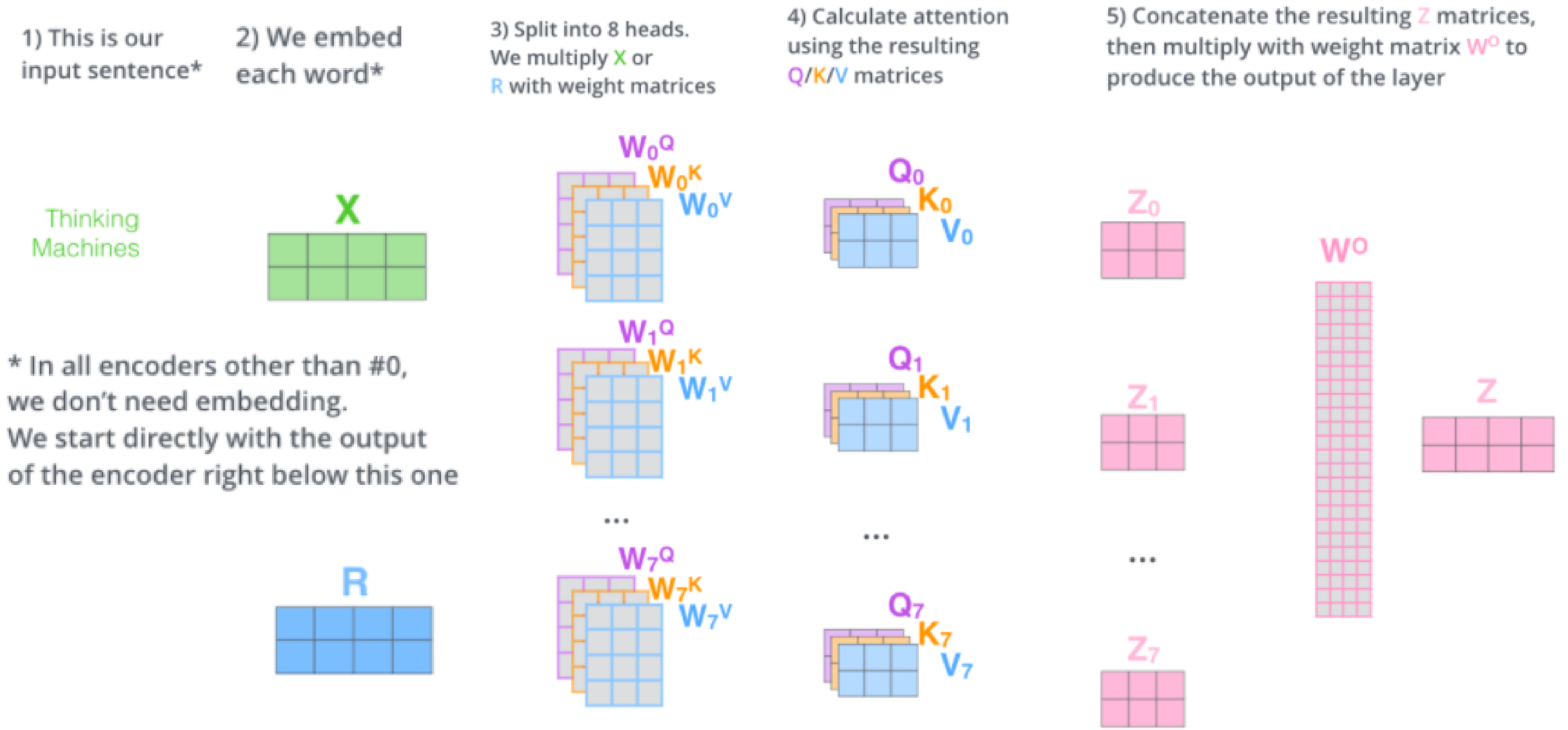
3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context



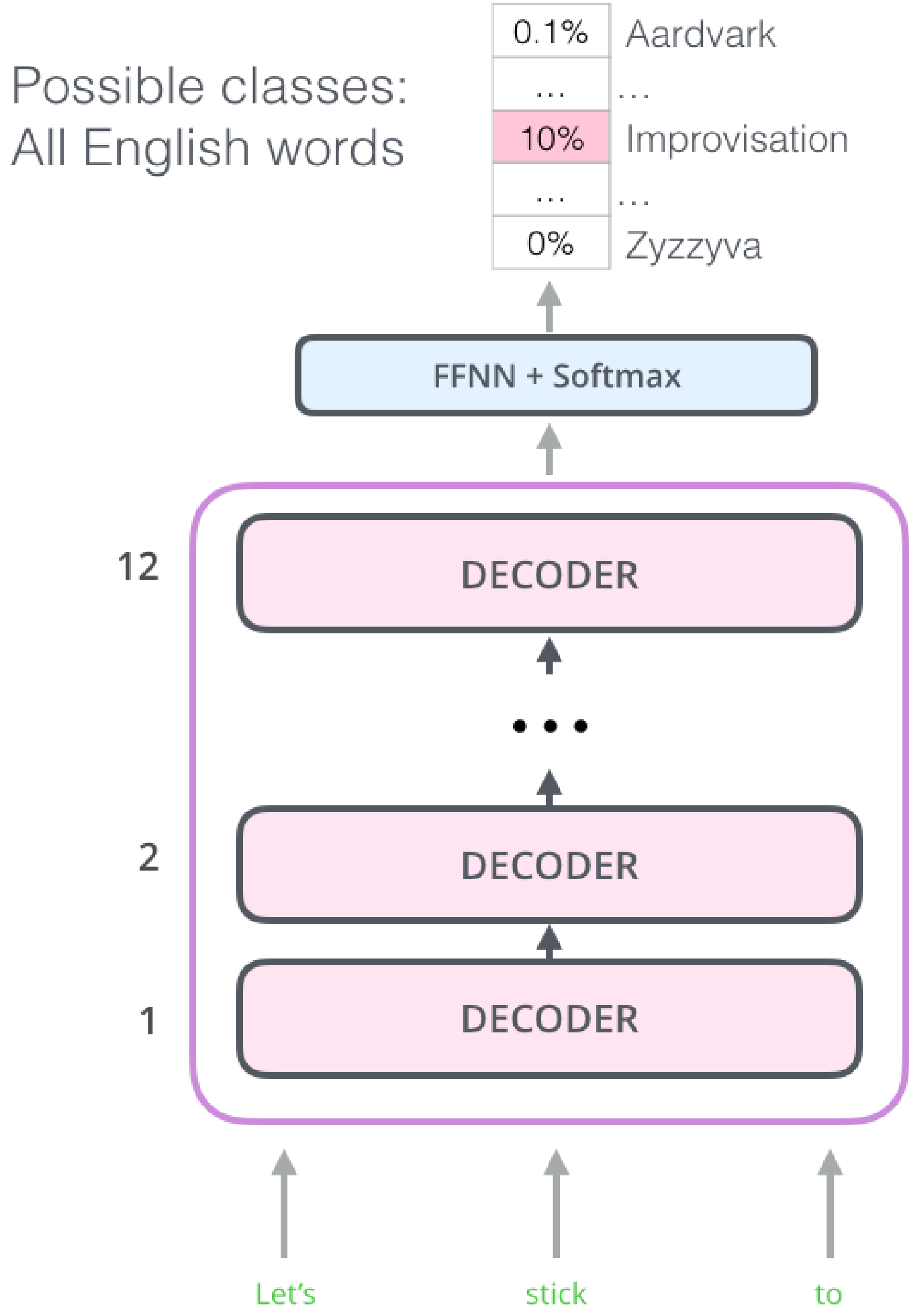

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

# Attention With Many Heads



# Open AI GPT

- OpenAI GPT use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer
- The decoder is a good choice because it’s a natural choice for language modeling (predicting the next word) since it’s built to mask future tokens


Possible classes:
All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
12
DECODER
2
DECODER
1
DECODER
Let's
stick
to

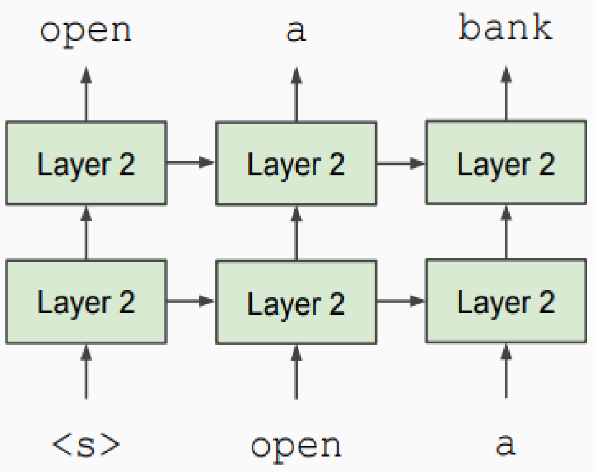
# Problem with Previous Methods

- Problem: Language models only use left context or right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
  - Reason 1: Directionality is needed to generate a well-formed probability distribution.
    - We don't care about this.
  - Reason 2: Words can "see themselves" in a bidirectional encoder.

# Unidirectional vs. Bidirectional Models

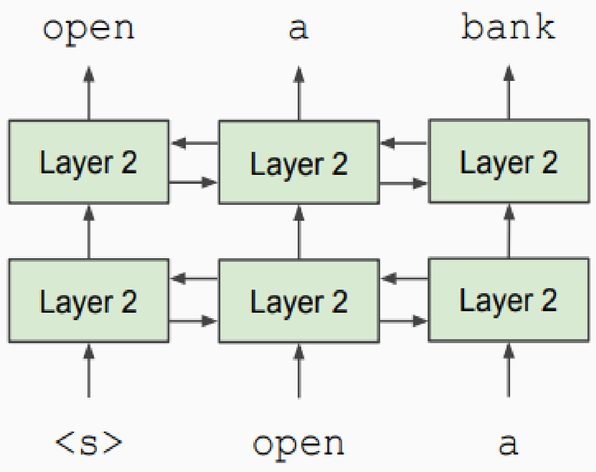
**Unidirectional context**

Build representation incrementally

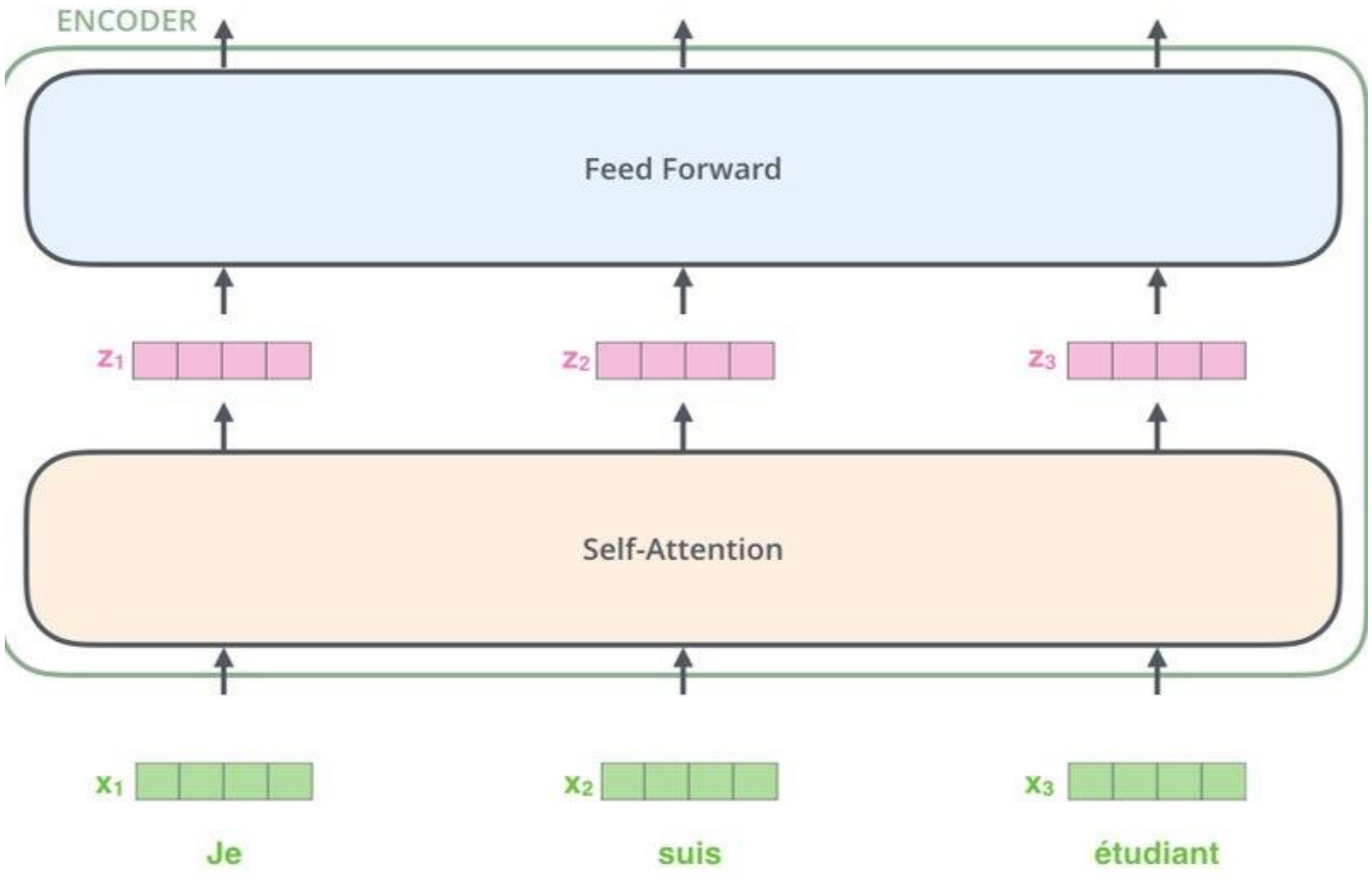



**Bidirectional context**

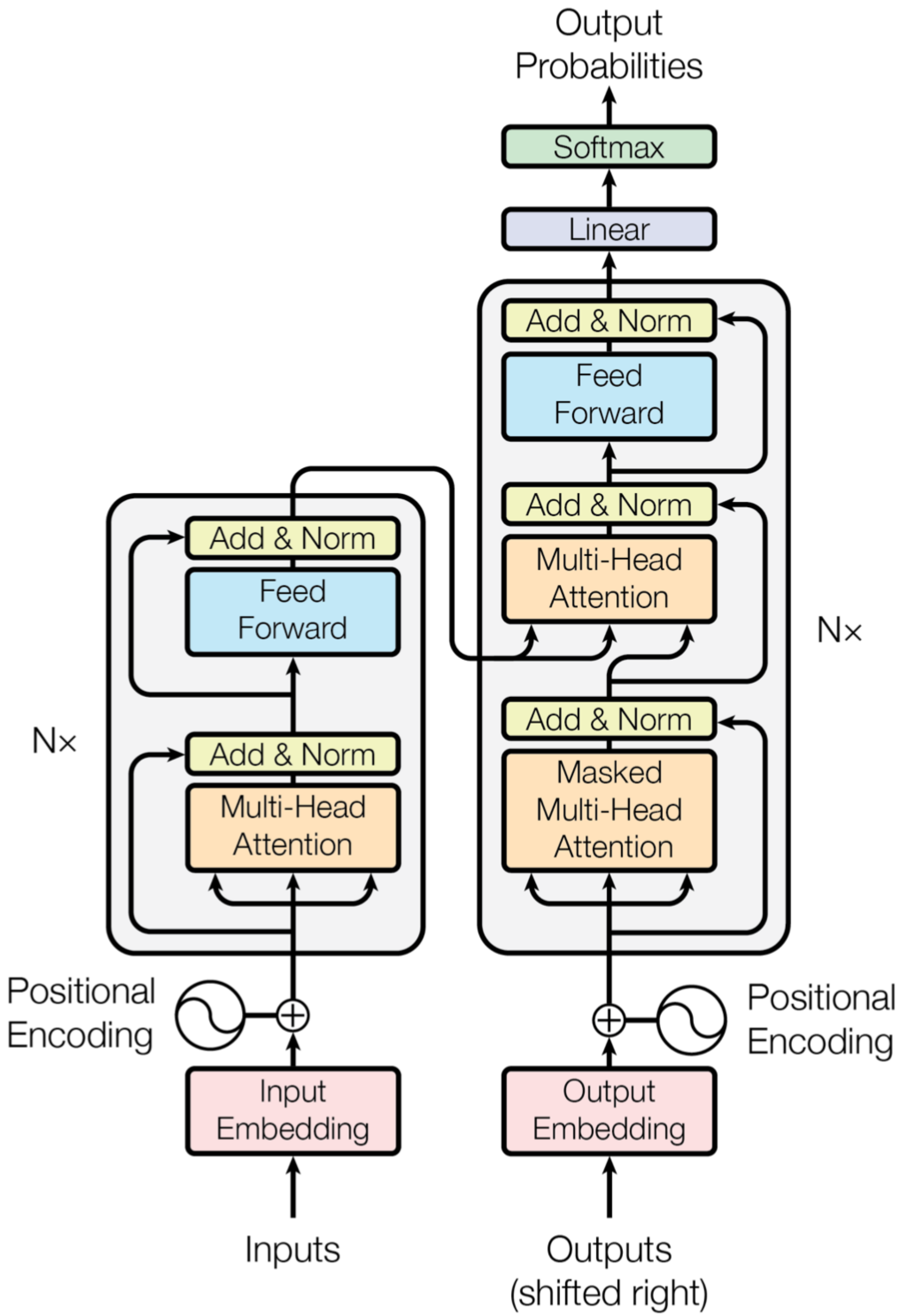
Words can “see themselves”

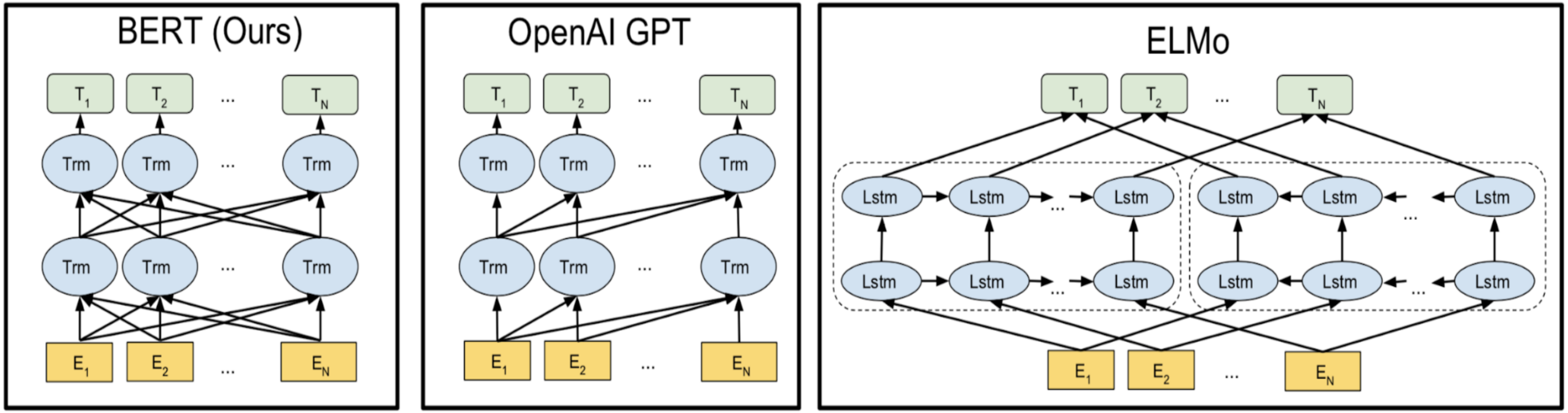


»



ENCODER
Feed Forward
$z_1$
$z_2$
$z_3$
Self-Attention
$x_1$
$x_2$
$x_3$
Je
suis
étudiant


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)


BERT (Ours)
T1 T2 ... TN
Trm Trm ... Trm
Trm Trm ... Trm
E1 E2 ... EN
OpenAI GPT
T1 T2 ... TN
Trm Trm ... Trm
Trm Trm ... Trm
E1 E2 ... EN
ELMo
T1 T2 ... TN
Lstm Lstm ... Lstm
Lstm Lstm ... Lstm
Lstm Lstm ... Lstm
Lstm Lstm ... Lstm
E1 E2 ... EN

# BERT: From Decoders to Encoders

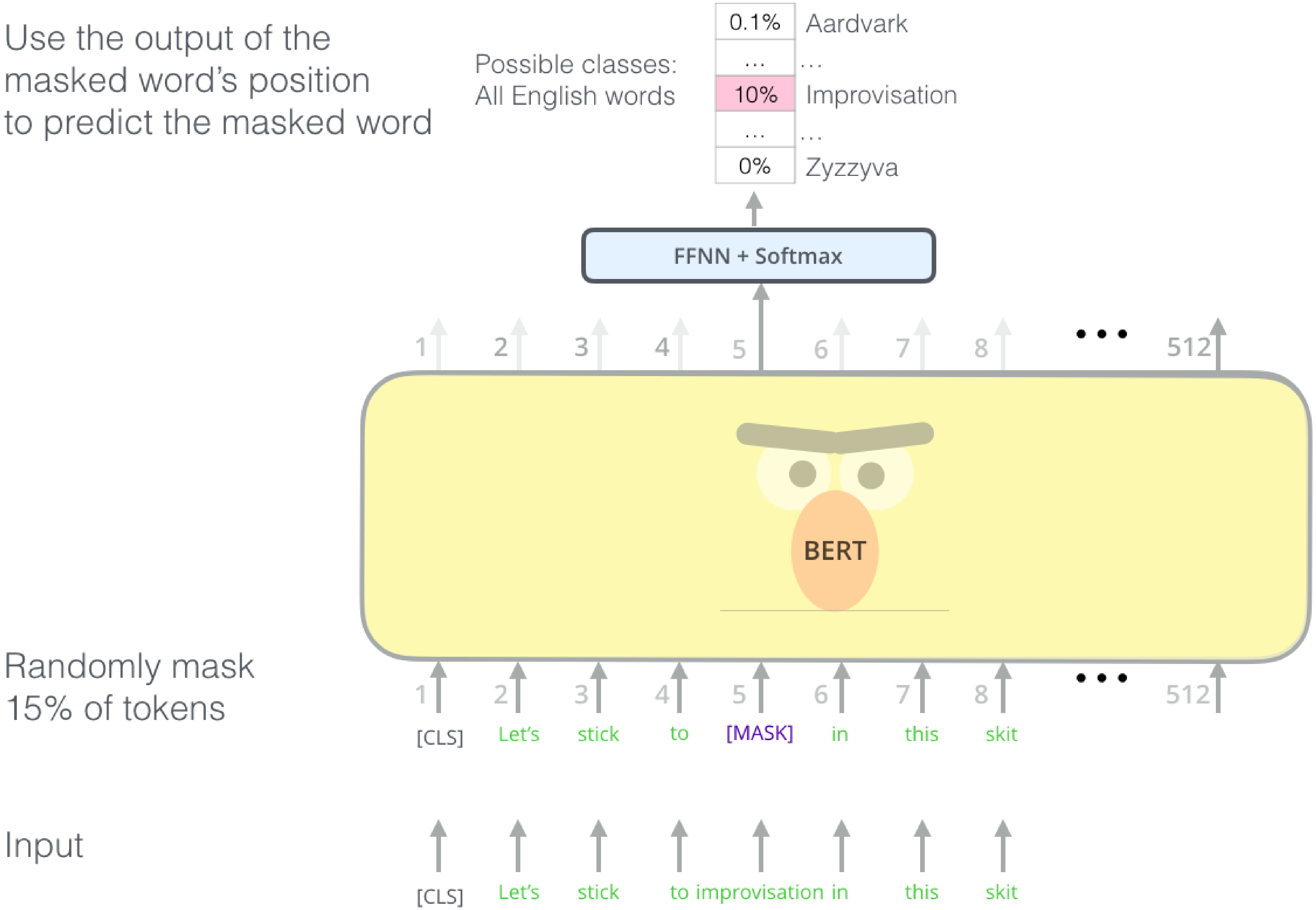
- **Problem**: Could we build a transformer-based model whose language model looks both forward and backwards?
  - “We’ll use transformer encoders”
- **Problem continued**: Everybody knows bidirectional conditioning would allow each word to indirectly see itself in a multi-layered context.

# Masked LM

- **Solution**: Mask out k% of the input words, and then predict the masked words
  - We always use k = 15%

```
                          store              gallon
                            ↑                  ↑
the man went to the [MASK] to buy a [MASK] of milk
```

- Too little masking: Too expensive to train
- Too much masking: Not enough context


Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
1 2 3 4 5 6 7 8 ... 512
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

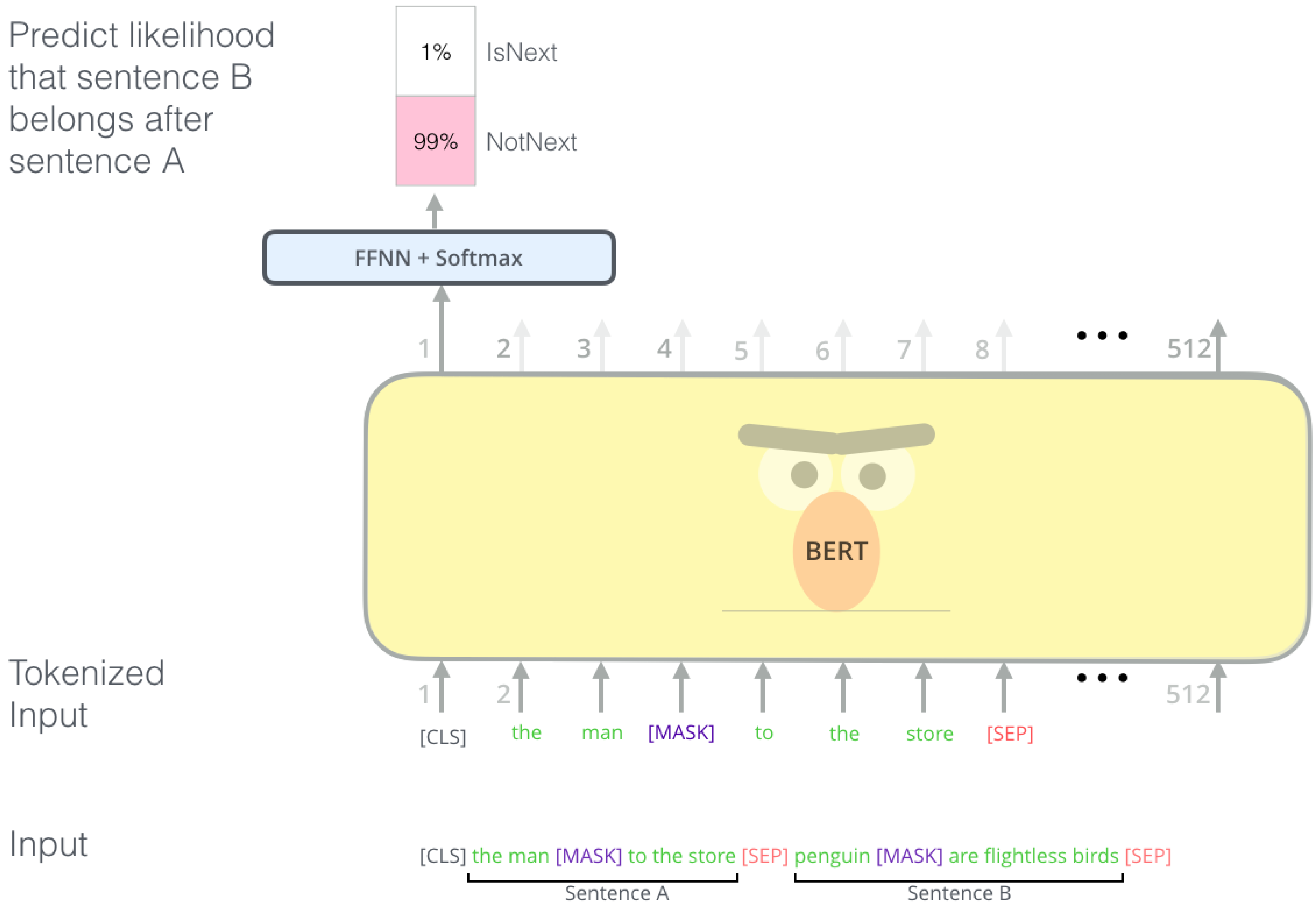
# Masked LM

- **Problem**: Mask token never seen at fine-tuning
- **Solution**: 15% of the words to predict, but don't replace with [MASK] 100% of the time. Instead:
- 80% of the time, replace with [MASK]
  - went to the store → went to the [MASK]
- 10% of the time, replace random word
  - went to the store → went to the running
- 10% of the time, keep same
  - went to the store → went to the store

# Next Sentence Prediction

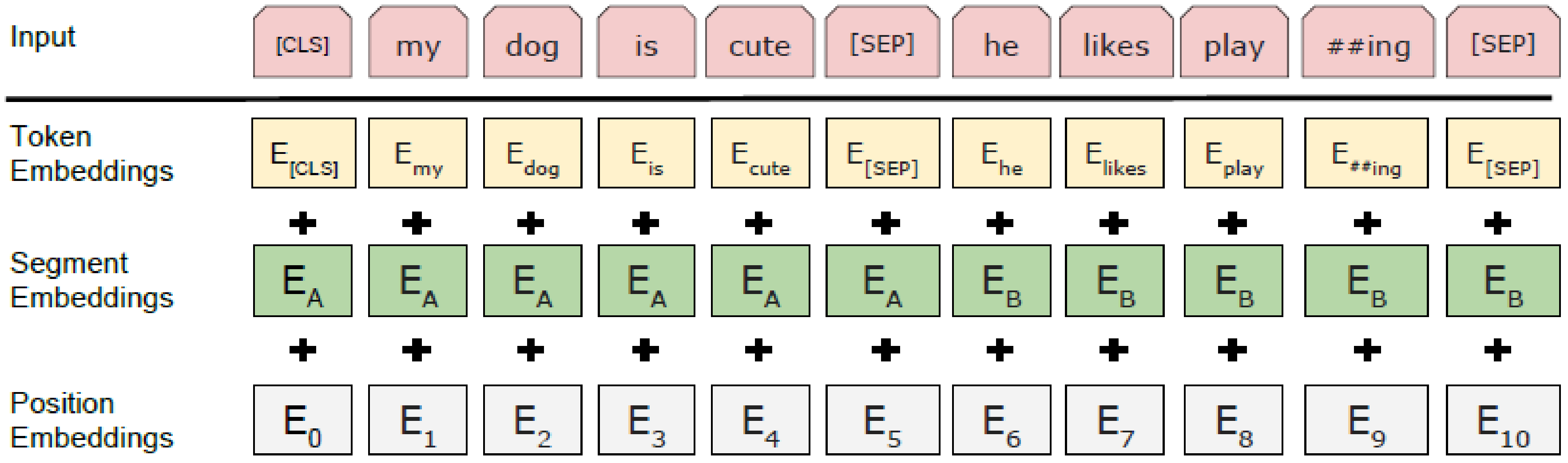
- To learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

| | |
|---|---|
| **Sentence A** = `The man went to the store.` | **Sentence A** = `The man went to the store.` |
| **Sentence B** = `He bought a gallon of milk.` | **Sentence B** = `Penguins are flightless.` |
| **Label** = `IsNextSentence` | **Label** = `NotNextSentence` |


Predict likelihood that sentence B belongs after sentence A
1%
IsNext
99%
NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
512
BERT
Tokenized Input
1
2
512
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

# Input Representation

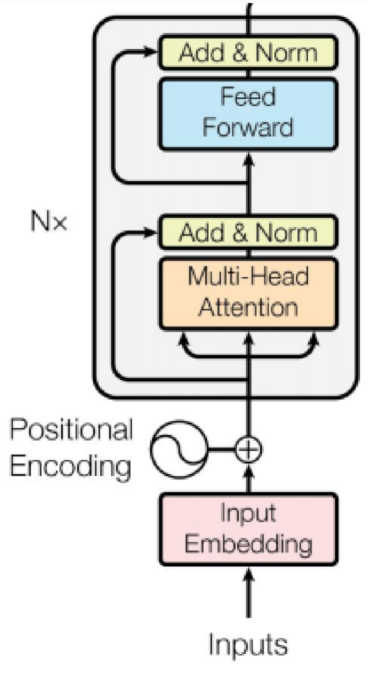
» Use 30,000 WordPiece vocabulary on input. Each token is sum of three embeddings. Single sequence is much more efficient.

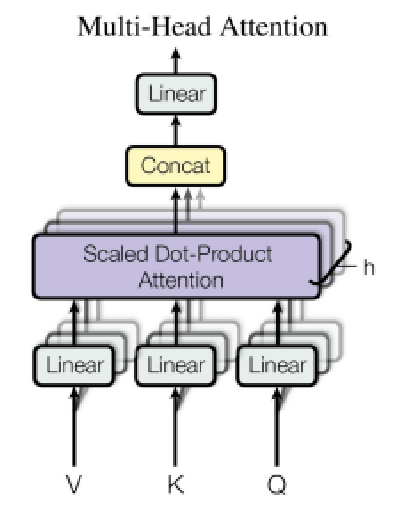


# Model Architecture

## Transformer encoder

- Multi-headed self attention
  - Models context
- Feed-forward layers
  - Computes non-linear hierarchical features
- Layer norm and residuals
  - Makes training deep networks healthy
- Positional embeddings
  - Allows model to learn relative positioning



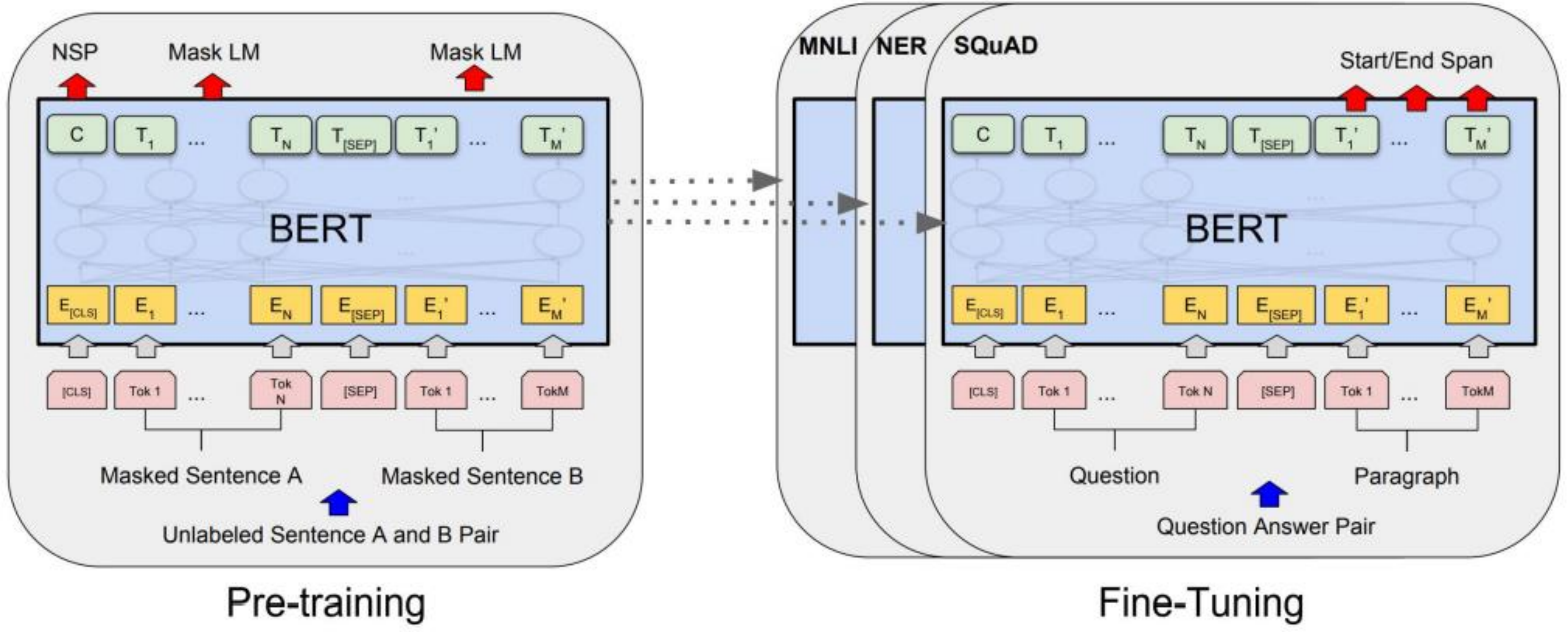



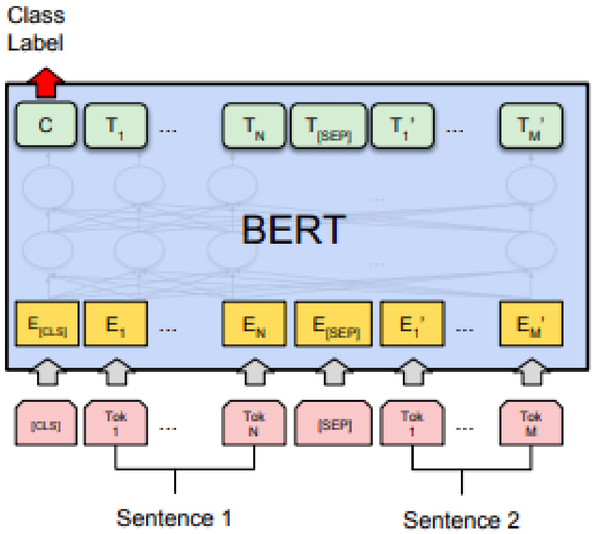
# Loss Function

- When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

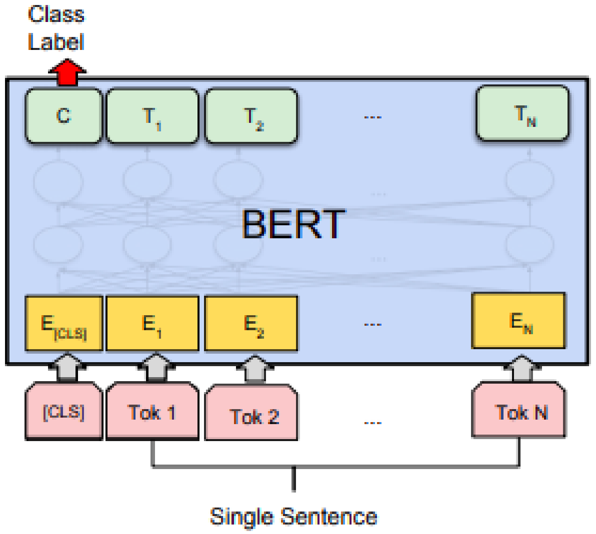
- BERT uses cross entropy loss as its loss function.

# BERT Fine-Tuning

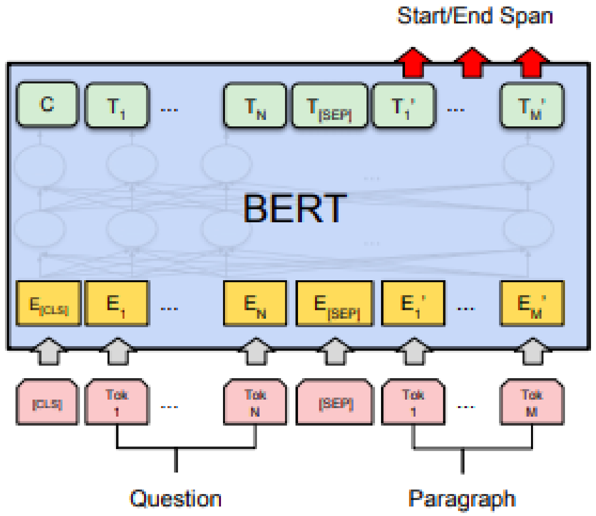

NSP
Mask LM
Mask LM
C
$T_1$
...
$T_N$
$T_{[SEP]}$
$T_1'$
...
$T_M'$
BERT
$E_{[CLS]}$
$E_1$
...
$E_N$
$E_{[SEP]}$
$E_1'$
...
$E_M'$
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
C
$T_1$
...
$T_N$
$T_{[SEP]}$
$T_1'$
...
$T_M'$
BERT
$E_{[CLS]}$
$E_1$
...
$E_N$
$E_{[SEP]}$
$E_1'$
...
$E_M'$
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Question
Paragraph
Question Answer Pair
Fine-Tuning



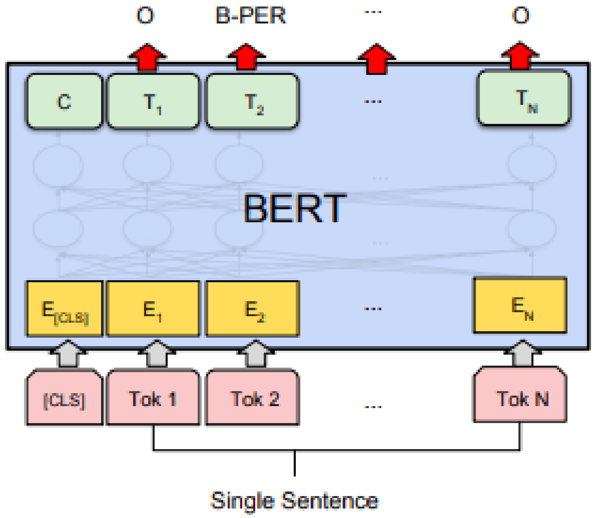

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG




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

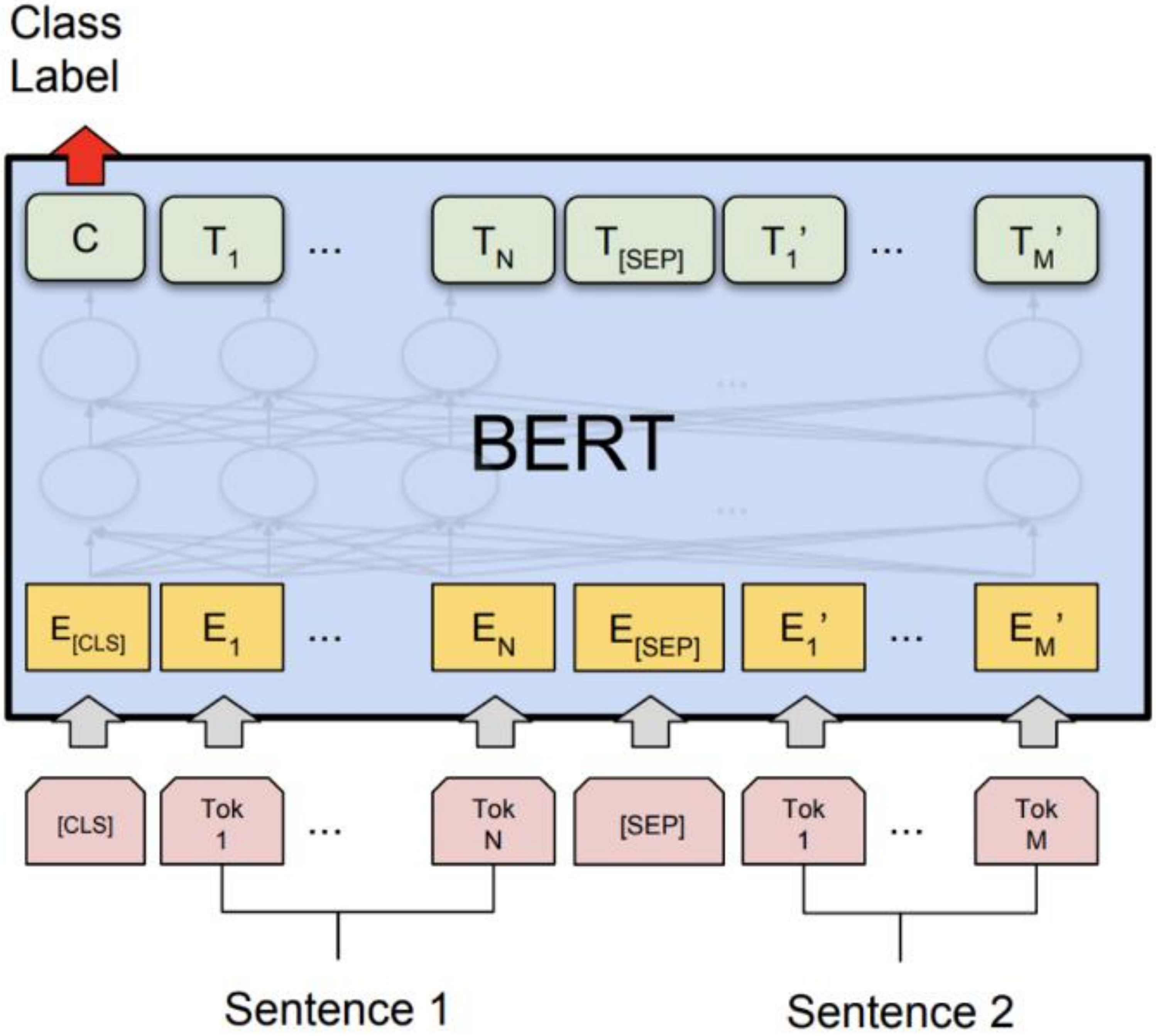



(c) Question Answering Tasks: SQuAD v1.1

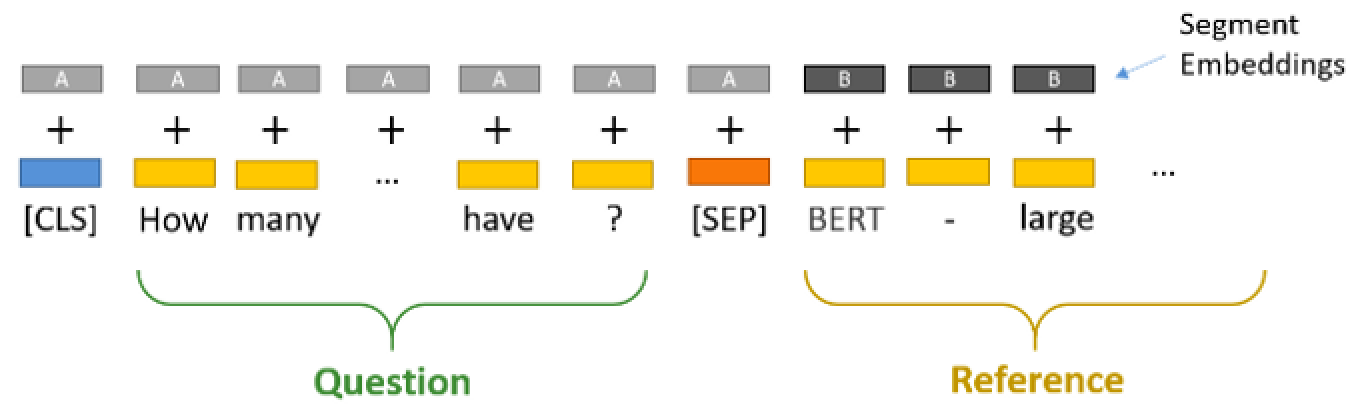



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

# Fine-Tune BERT for Classification


Class
Label
C
$T_1$
...
$T_N$
$T_{[SEP]}$
$T_1'$
...
$T_M'$
BERT
$E_{[CLS]}$
$E_1$
...
$E_N$
$E_{[SEP]}$
$E_1'$
...
$E_M'$
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
Tok M
Sentence 1
Sentence 2

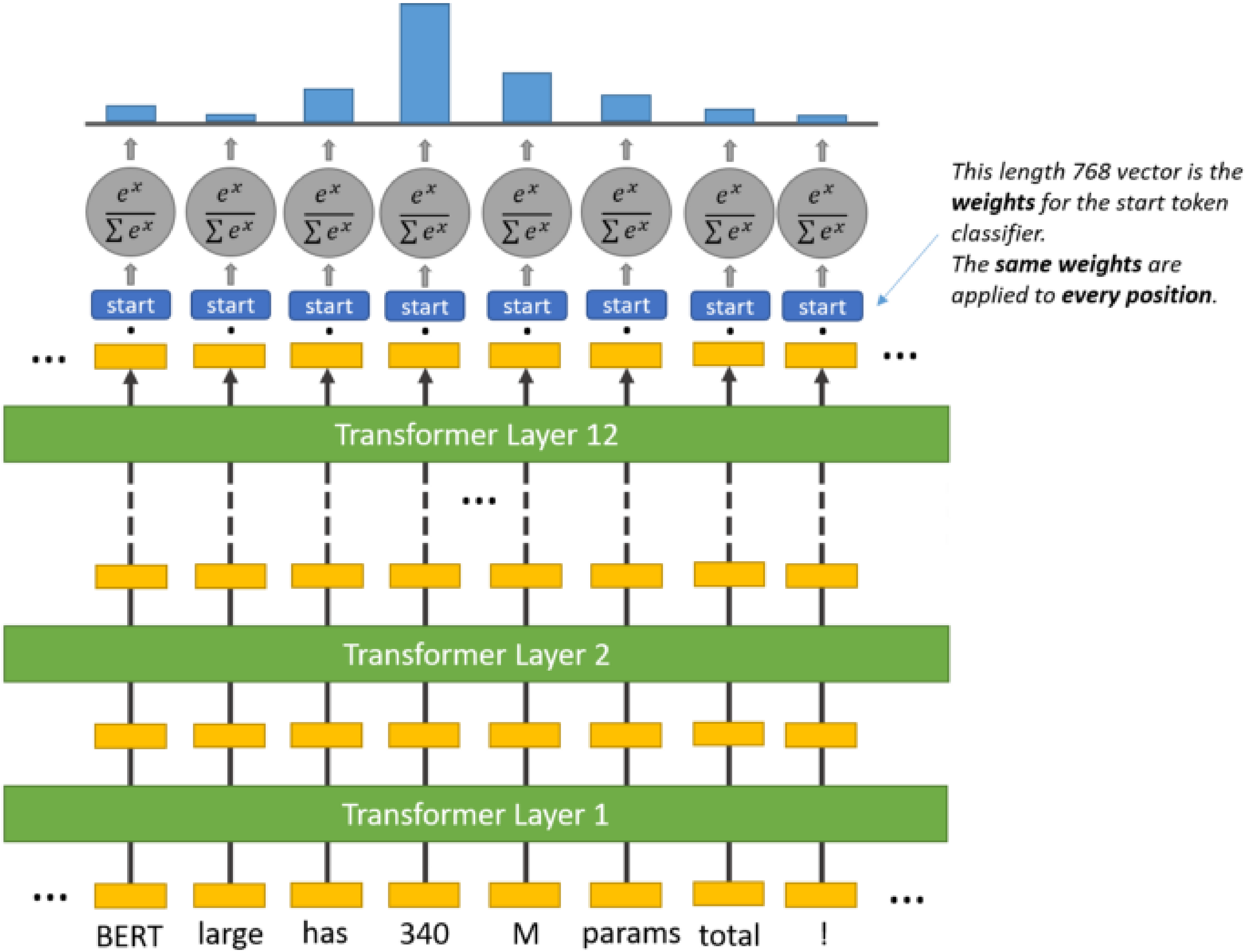
# Fine-Tune BERT for SQuaD


Segment Embeddings
A
A
A
A
A
A
A
B
B
B
+
...
...
[CLS]
How
many
have
?
[SEP]
BERT
-
large
Question
Reference

*BERT introduces a start vector and an end vector.*

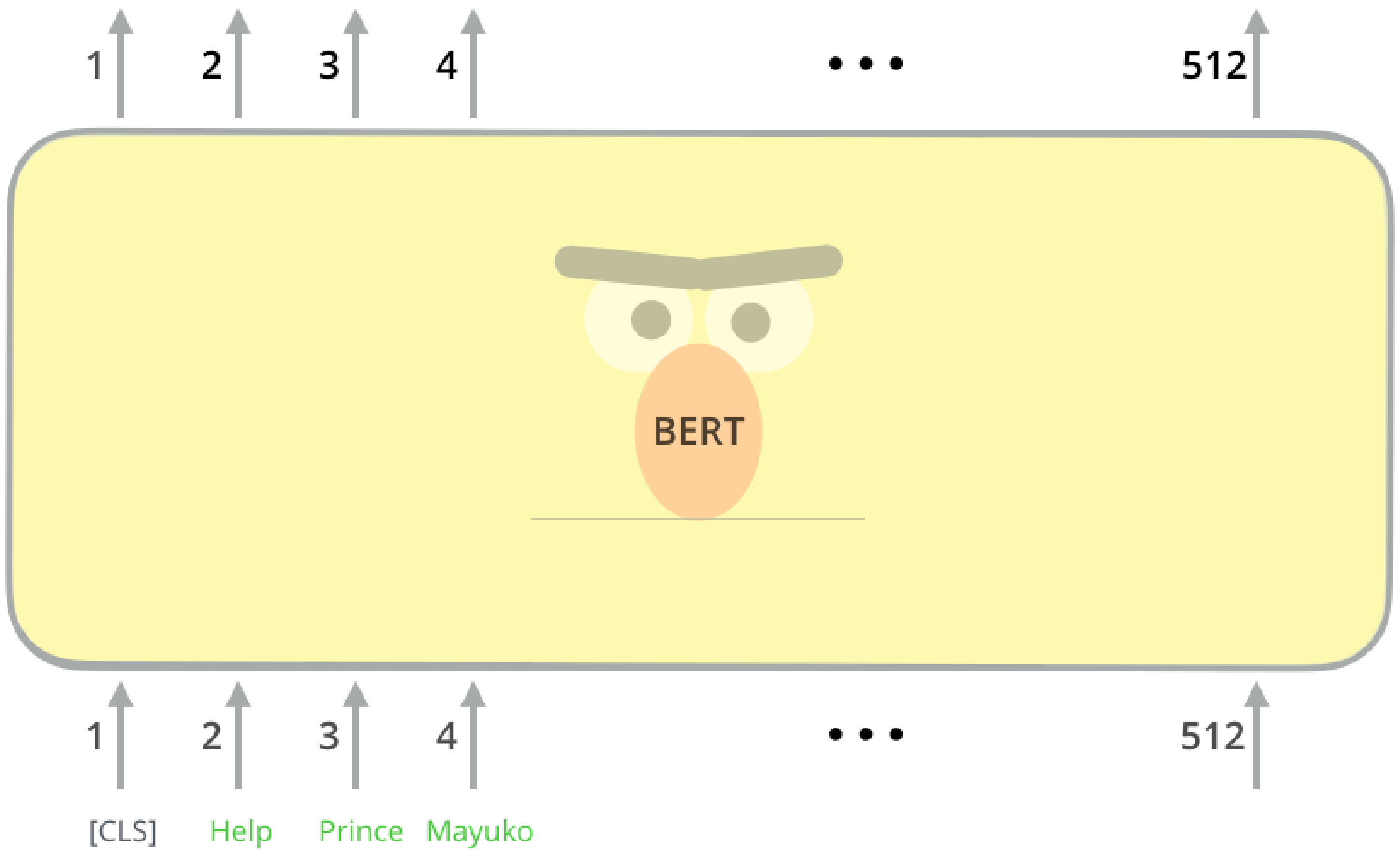
*The probability of each word being the start-word is calculated by taking a dot product between the final embedding of the word and the start vector, followed by a softmax over all the words.*

*The word with the highest probability value is considered.*

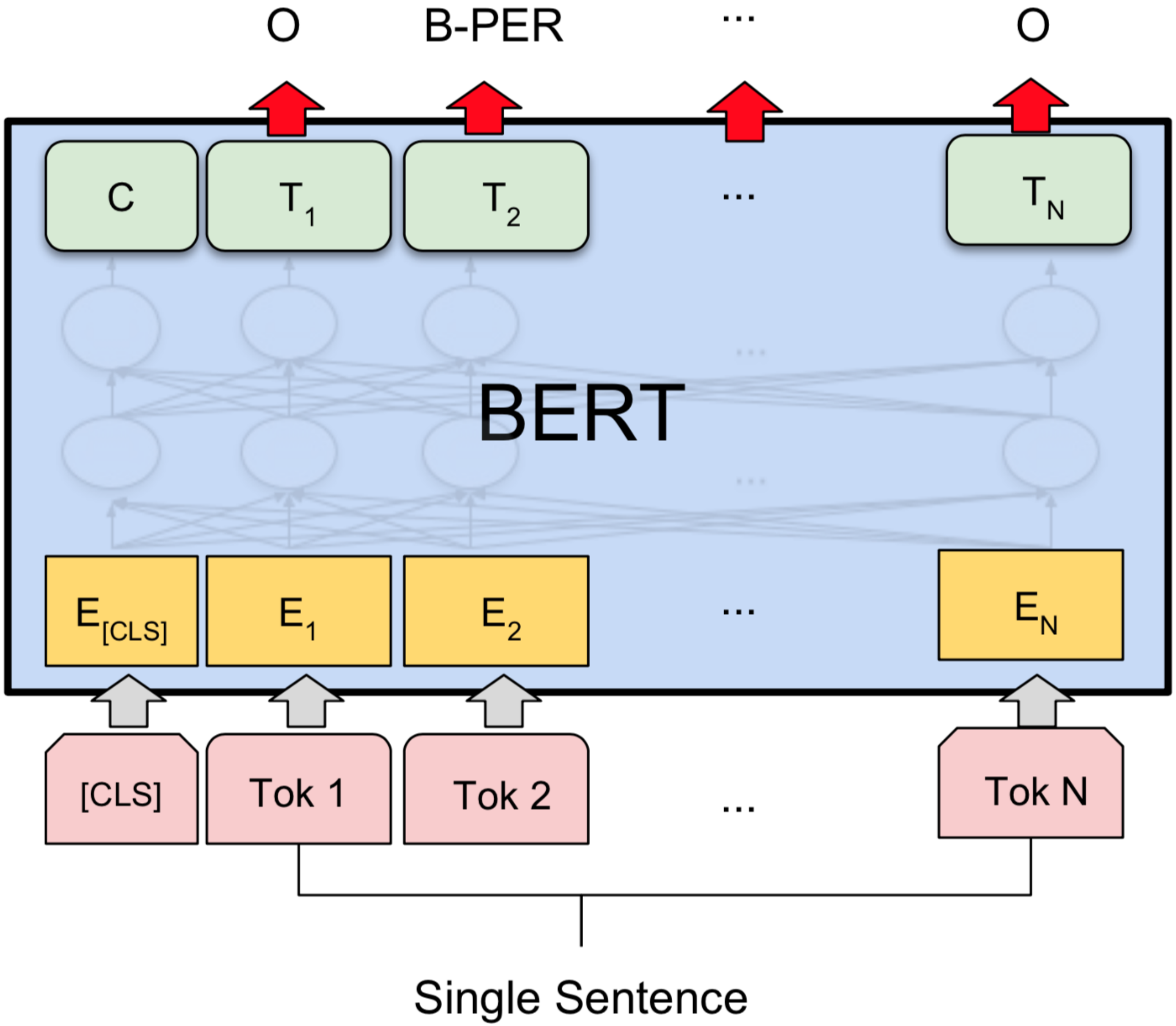


This length 768 vector is the **weights** for the start token classifier.
The **same weights** are applied to **every position**.
$\frac{e^x}{\sum e^x}$
start
Transformer Layer 12
Transformer Layer 2
Transformer Layer 1
BERT
large
has
340
M
params
total
!

# Fine-Tune BERT for Named Entity Recognition



1
2
3
4
512
BERT
1
2
3
4
512
[CLS]
Help
Prince
Mayuko


O
B-PER
...
O
C
$T_1$
$T_2$
...
$T_N$
BERT
$E_{[CLS]}$
$E_1$
$E_2$
...
$E_N$
[CLS]
Tok 1
Tok 2
...
Tok N
Single Sentence

# Results

**QNLI** Question Natural Language Inference is a version of the Stanford Question Answering Dataset (Rajpurkar et al., 2016) which has been converted to a binary classification task (Wang et al., 2018a). The positive examples are (question, sentence) pairs which do contain the correct answer, and the negative examples are (question, sentence) from the same paragraph which do not contain the answer.

n2

a least natural number?

ny calories does a Dominos ve?

ı one start a bakery business?

o choose between learning
d Python, what should I choose first?

**SST-2** The Stanford Sentiment Treebank is a binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment (Socher et al., 2013).

| | | P | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | | k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| [illegible] | [illegible] | [illegible] | 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** |

**STS-B** The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines and other sources (Cer et al., 2017). They were annotated with a score from 1 to 5 denoting how similar the two sentences are in terms of semantic meaning.

**CoLa**

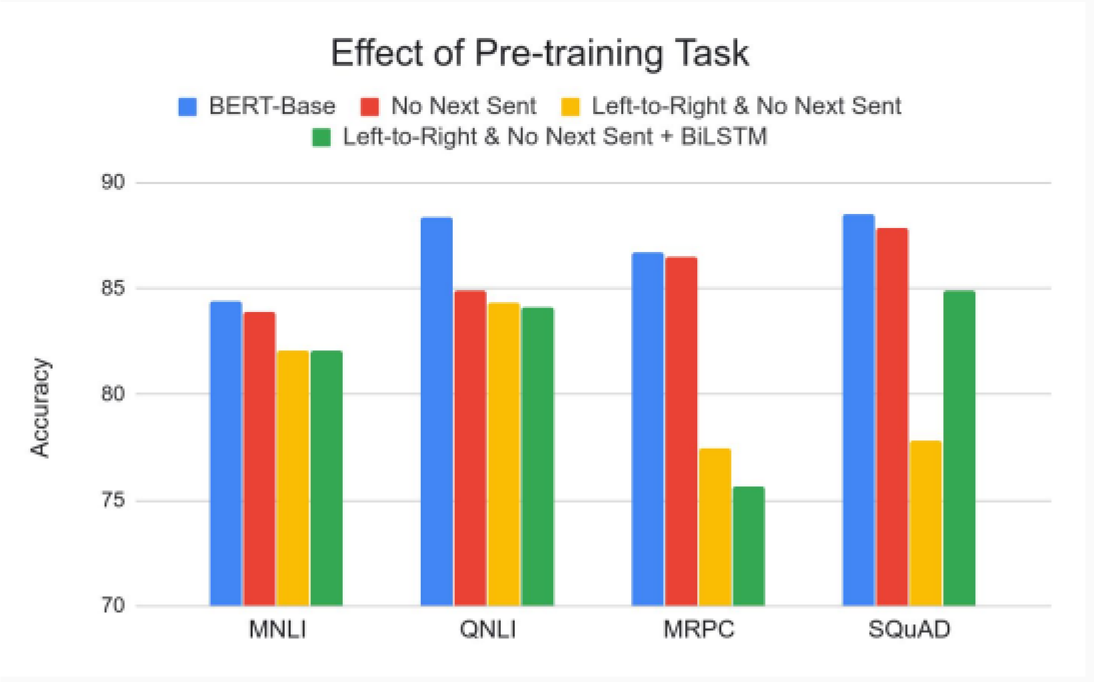
tence: The wagon rumbled down the road.
el: Acceptable

tence: The car honked down the road.
el: Unacceptable

Source - https://nlp.stanford.edu/seminar/details/jdevlin.pdf

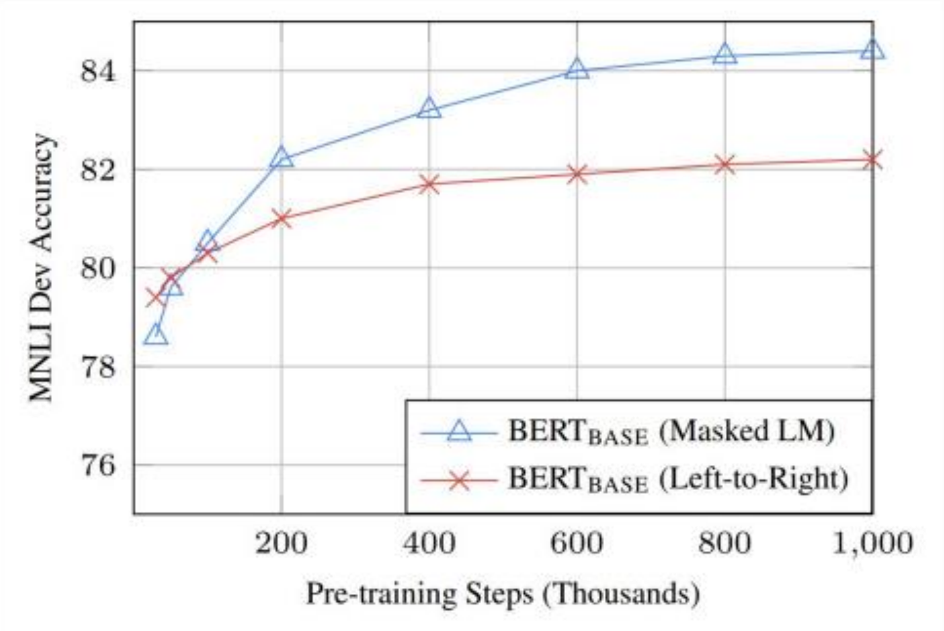
# Effect of Pre-training Task

- Masked LM (compared to left-to-right LM) is very important on some tasks, Next Sentence Prediction is important on other tasks.
- Left-to-right model does very poorly on word-level task (SQuAD), although this is mitigated by BiLSTM



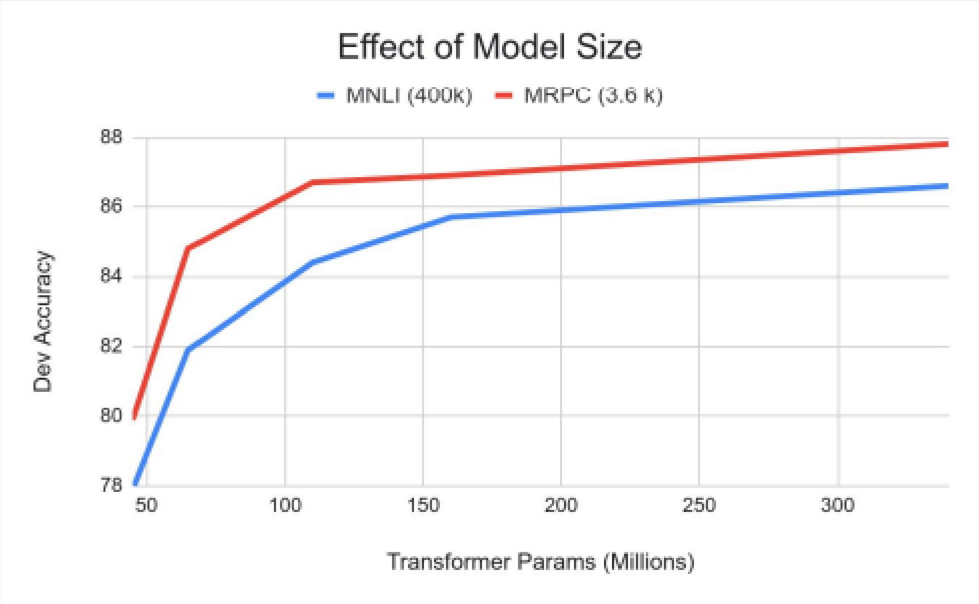
# Effect of Directionality and Training Time

» Masked LM takes slightly longer to converge because we only predict 15% instead of 100%. But absolute results are much better almost immediately



# Effect of Model Size

- Big models help a lot. Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples.



# Effect of Masking Strategy

- Masking 100% of the time hurts on feature-based approach
- Using random word 100% of time hurts slightly

| Masking Rates | | | Dev Set Results | | |
|---|---|---|---|---|---|
| MASK | SAME | RND | MNLI Fine-tune | NER Fine-tune | NER Feature-based |
| 80% | 10% | 10% | 84.2 | 95.4 | 94.9 |
| 100% | 0% | 0% | 84.3 | 94.9 | 94.0 |
| 80% | 0% | 20% | 84.1 | 95.2 | 94.6 |
| 80% | 20% | 0% | 84.4 | 95.2 | 94.7 |
| 0% | 20% | 80% | 83.7 | 94.8 | 94.6 |
| 0% | 0% | 100% | 83.6 | 94.9 | 94.6 |

## References


1. http://jalammar.github.io/illustrated-bert/
2. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
3. https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/
4. https://nlp.stanford.edu/seminar/details/jdevlin.pdf

## THANKS!



# Any questions?