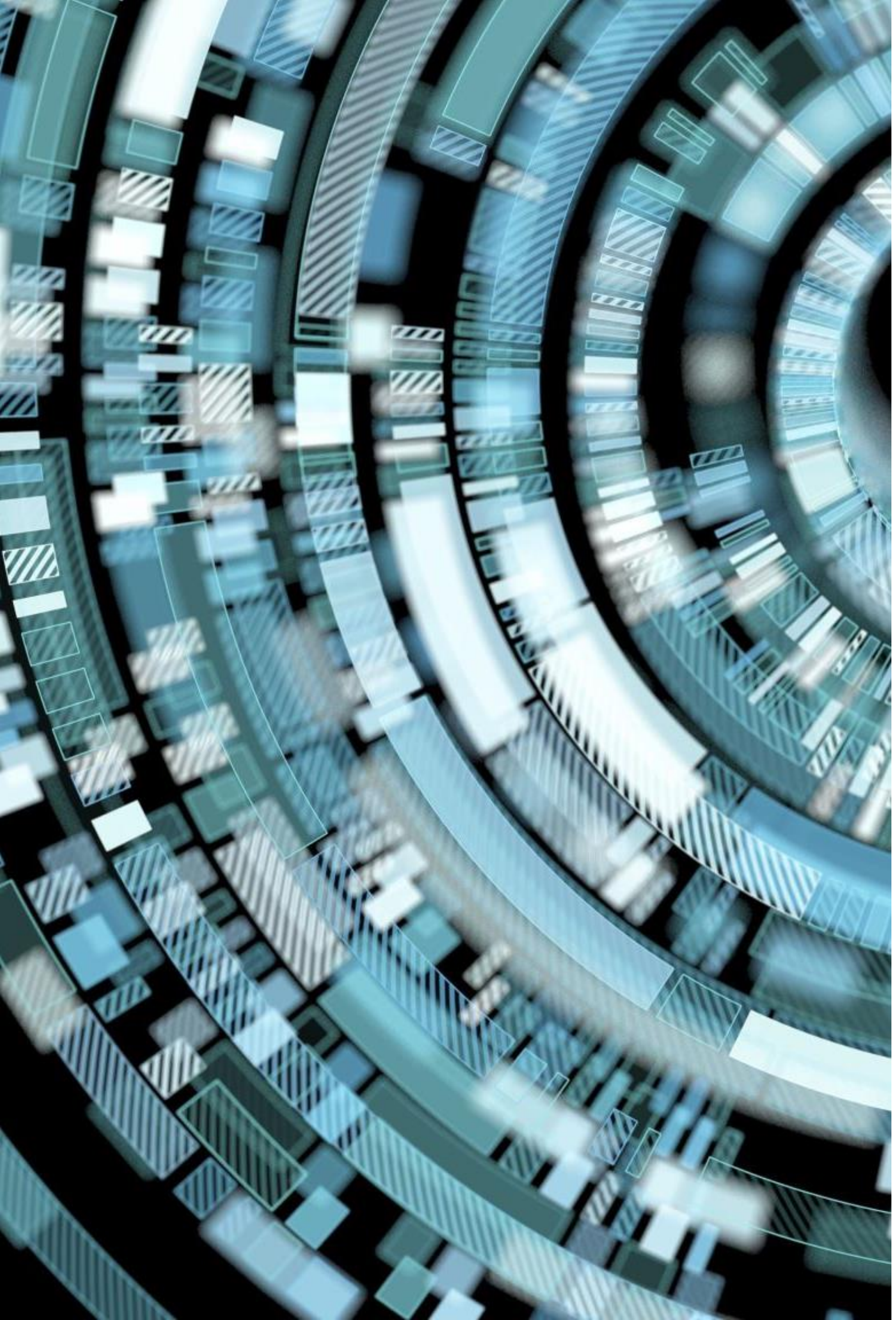
Transformers

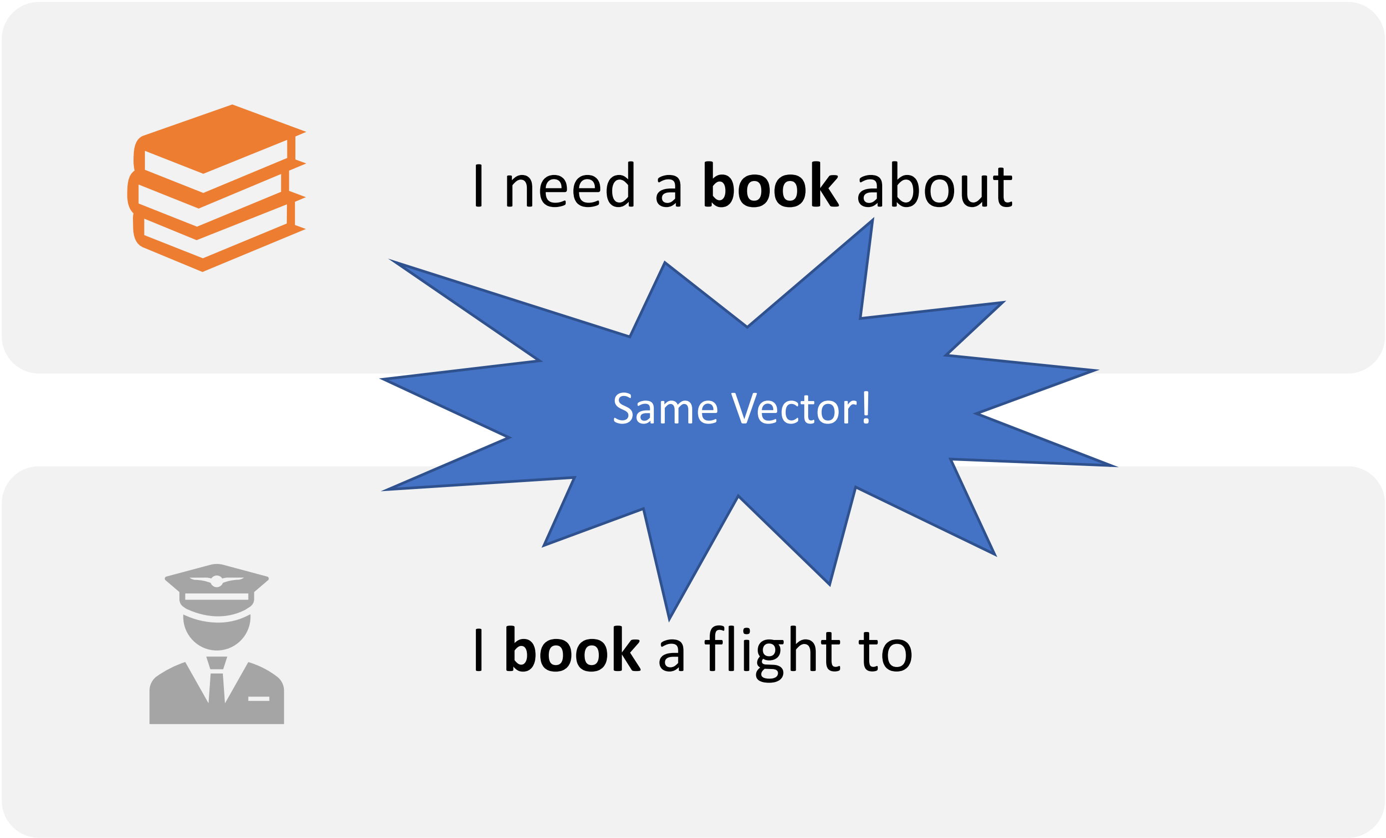
Liad Magen



Word Embeddings

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

Problems with Word Embedding



Contextual Representation

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

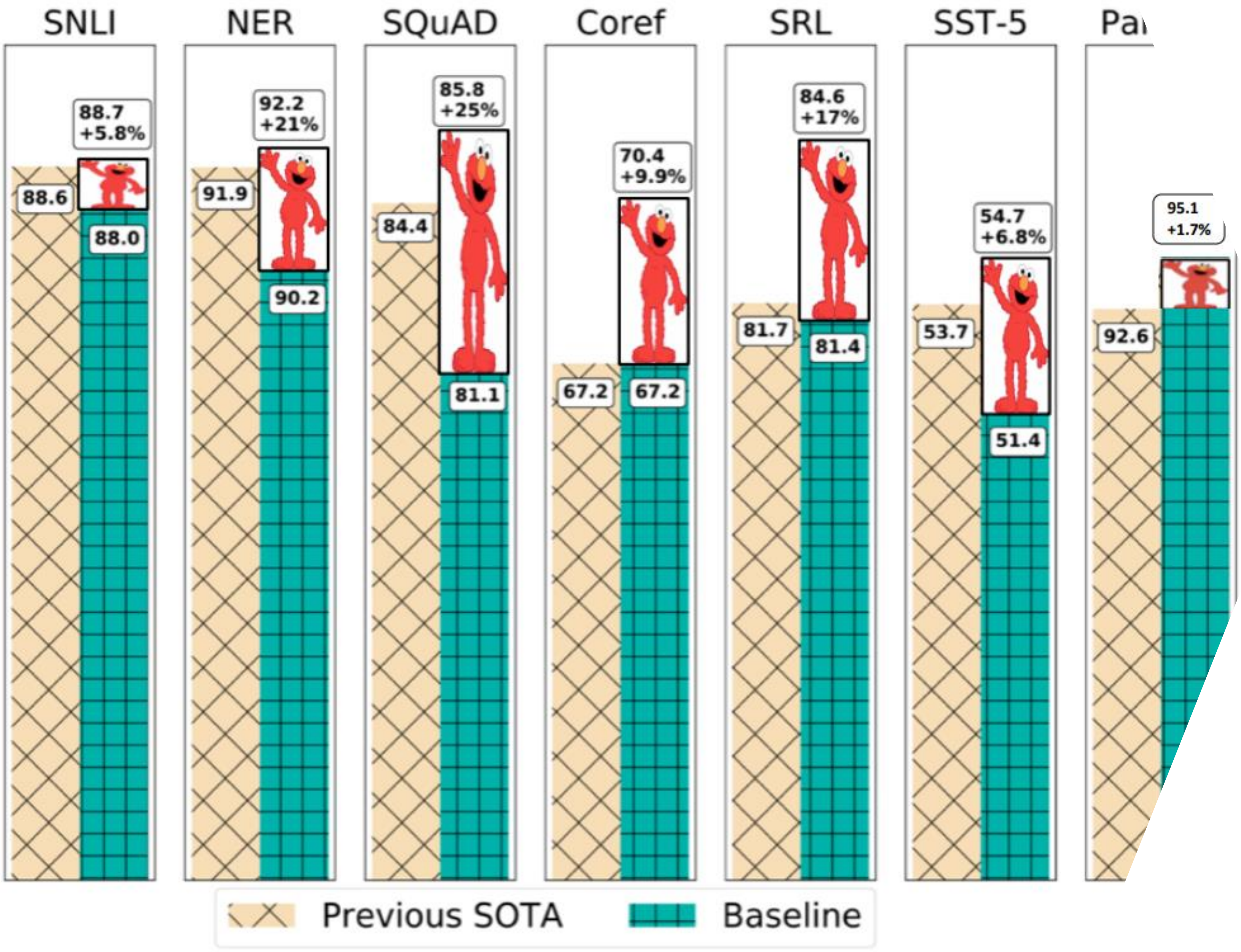
- **Solution**: Train contextual representations on a text corpus

- How?

Elmo

- Input: gloVe word vectors
- “Looks” at the entire sentence (not only a window)
  - BiLSTM
  - Fine-tune the word vector to match the context
- Can do classification, tagging, etc. very well
- A Break-through


Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ….etc


SNLI
NER
SQuAD
Coref
SRL
SST-5
Pa
88.7 +5.8%
88.6
88.0
92.2 +21%
91.9
90.2
85.8 +25%
84.4
81.1
70.4 +9.9%
67.2
67.2
84.6 +17%
81.7
81.4
54.7 +6.8%
53.7
51.4
95.1 +1.7%
92.6
Previous SOTA
Baseline

ULMFiT

- **U**niversal **L**anguage **M**odel **F**ine-tuning for **T**ext **C**lassification
- Transfer Learning
- Document Classification (But not word-level)

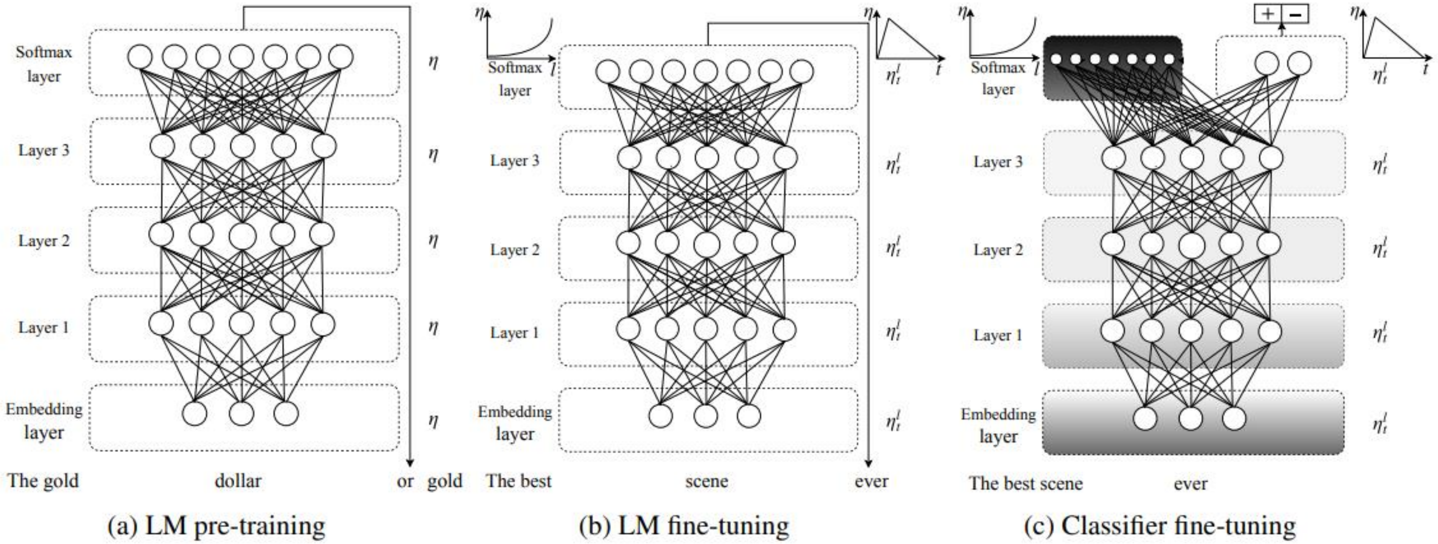


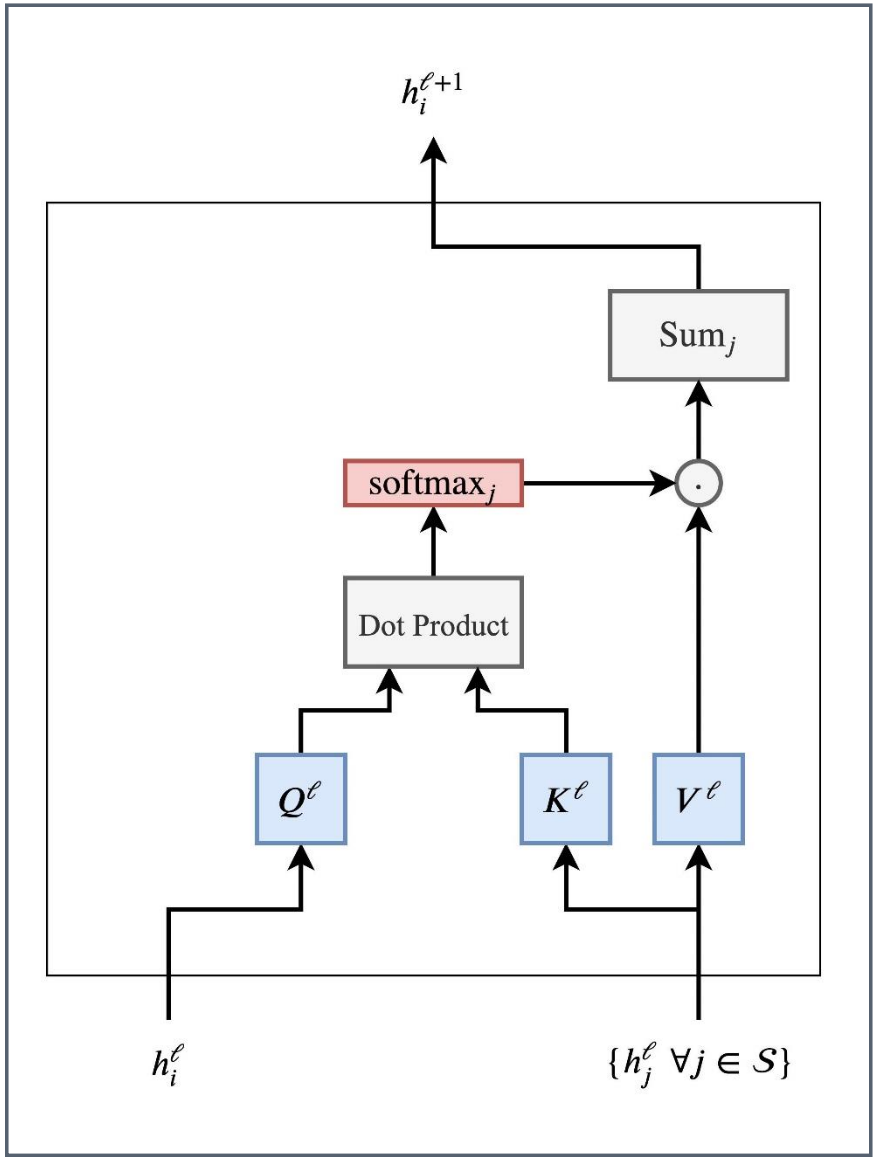
Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

https://docs.fast.ai/tutorial.text

Attention is all you need

- Lead to the Transformer (Vaswani et al. 2017)

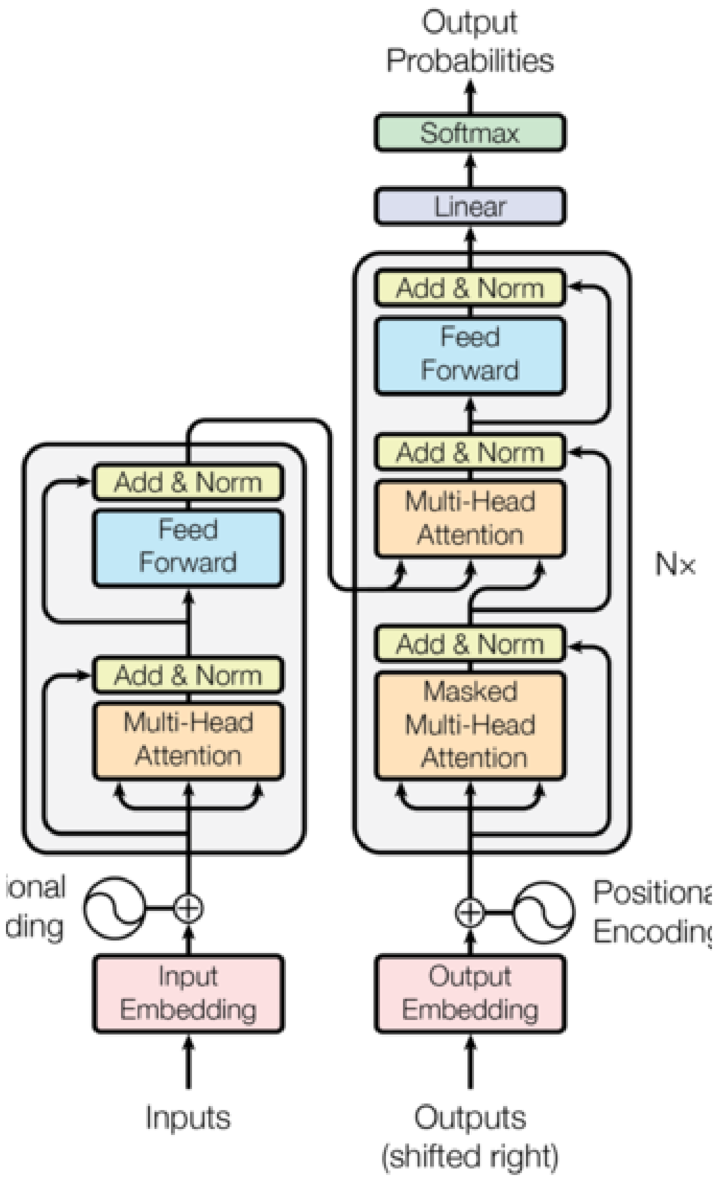
- Replace RNN with attention-based mechanism
- Concepts to get familiar with:
  - Self-attention
  - Multi-head attention



http://nlp.seas.harvard.edu/2018/04/03/attention.html

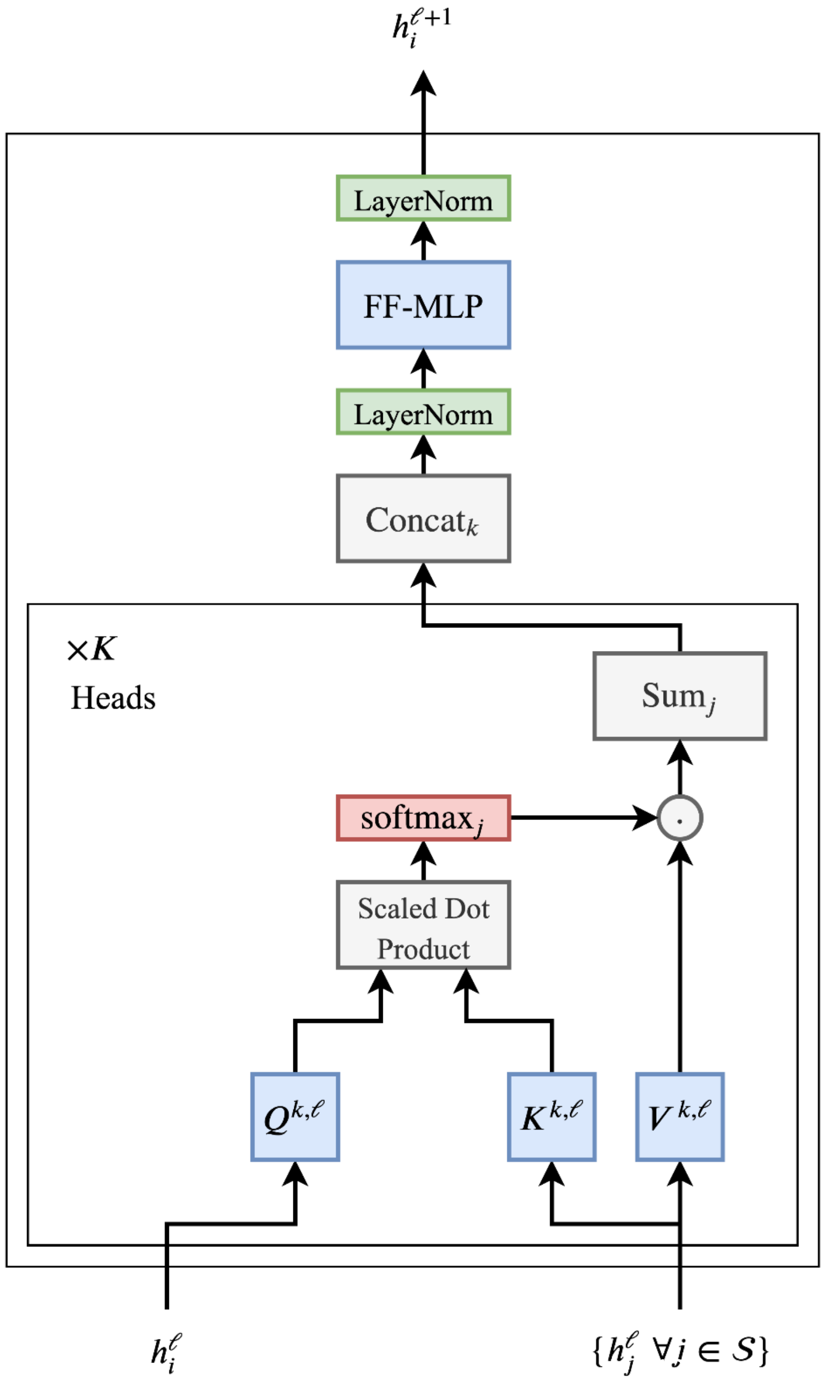
Self-Attention

- The input word vectors are the *queries*, *keys* and *values*
  Word vector stack = Q = K = V
- Each token attends all tokens in the previous layer
- The word vectors select each other



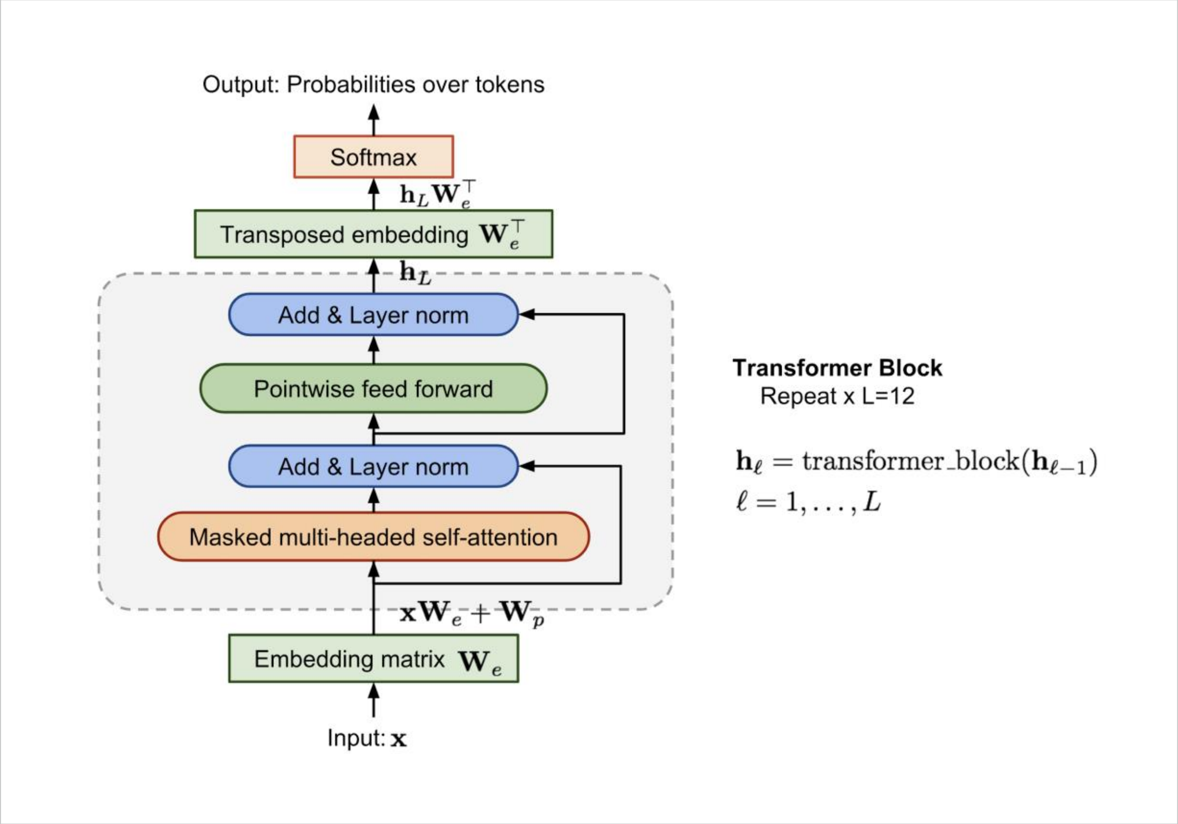
Self-Attention

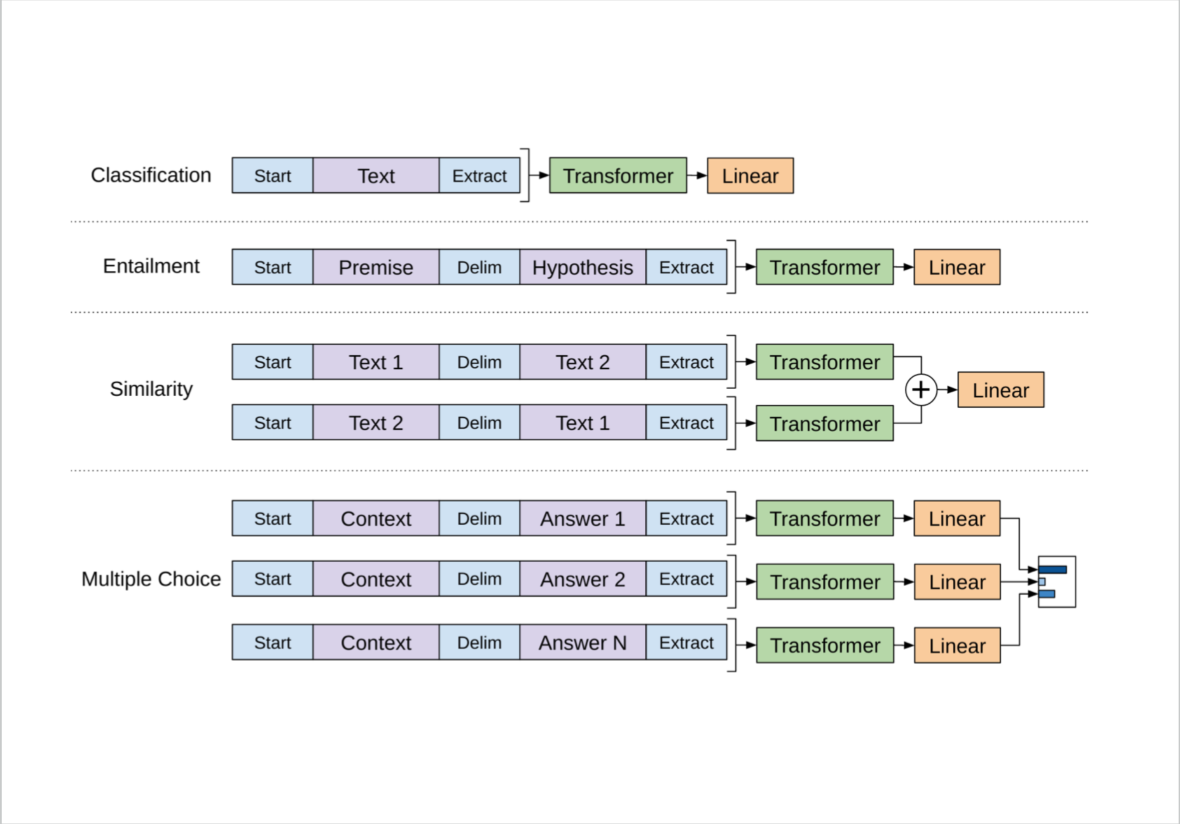
- The input word vectors are the *queries*, *keys* and *values*
  Word vector stack = Q = K = V
- Each token attends all tokens in the previous layer
- The word vectors select each other



GPT

- Like ELMo, but Transformers instead of BiLSTM
- Trained on different tasks – but updates the same weights





Scaling up

fast.ai

Jan. 2018

ULMFiT – Jan 2018, training: 1 GPU day

Oct. 2018

BERT Oct 2018, training: 256 TPU days (~320-560 GPU days)

GPT – June 2018, training: 240 GPU days

June 2018

GPT-2, training: ~2048 TPU v3 days

Feb. 2019

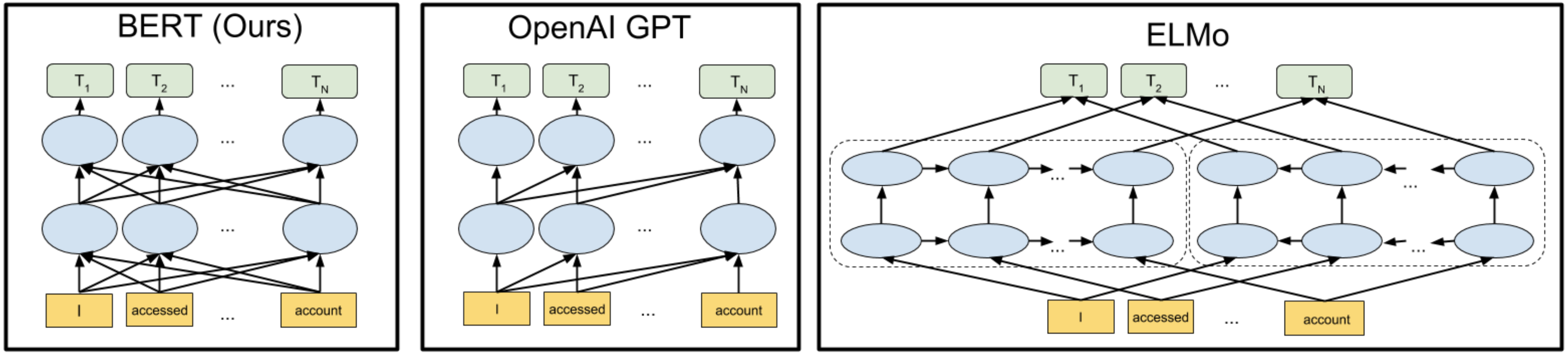
OpenAI

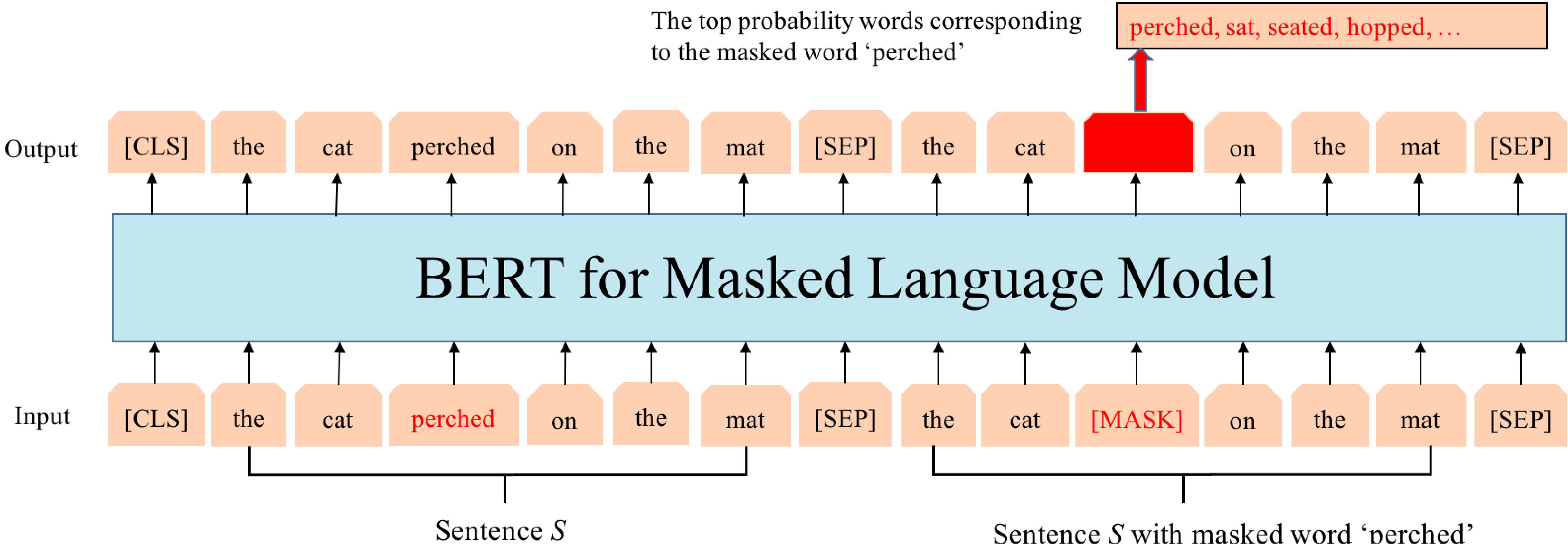
BERT

- LSTM → Transformer
- additional training objective: next sentence prediction
- Real bidirectional deep model
  - (with masked LM)

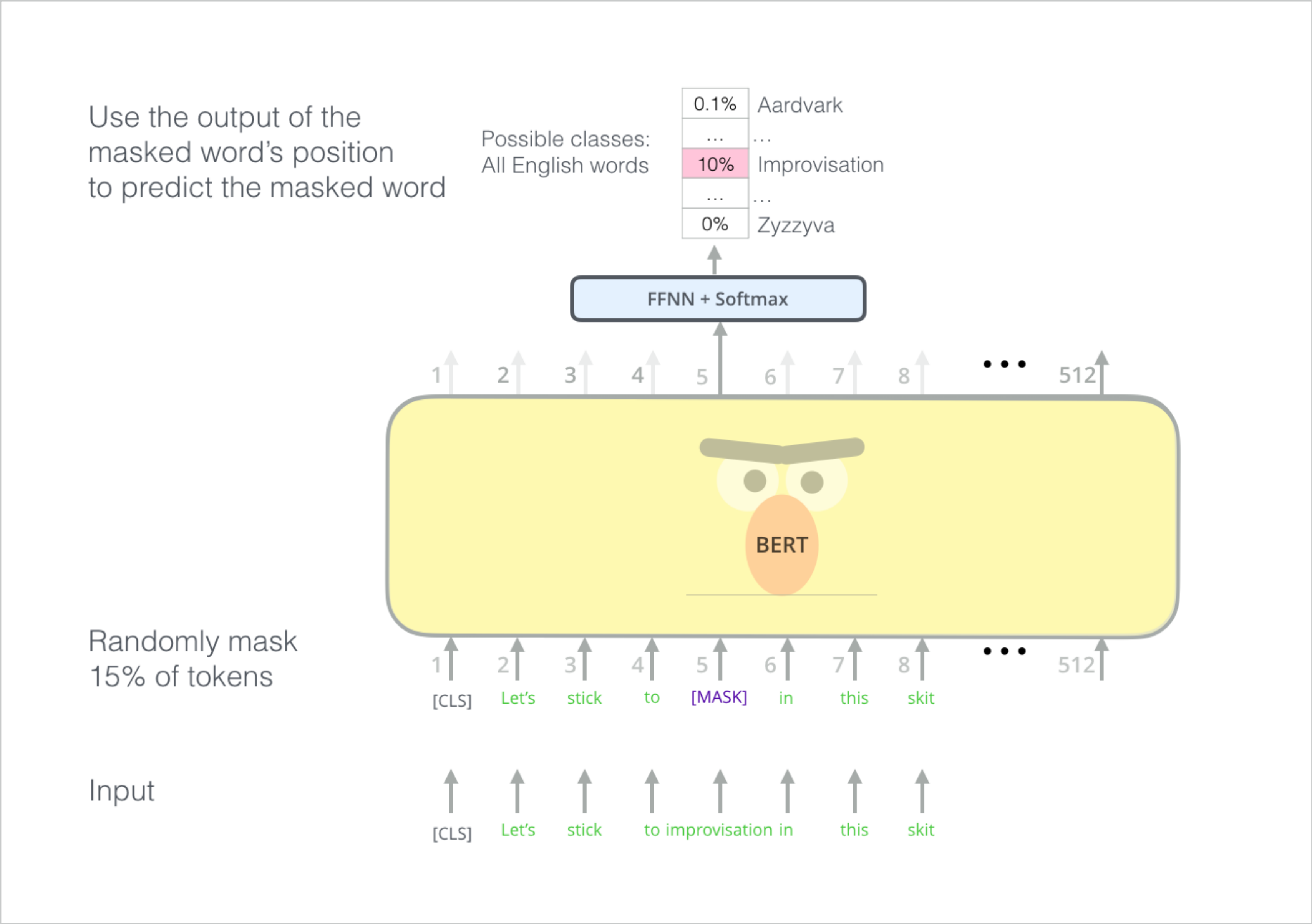
- Expensive to train…

BERT is fully, deep, bidirectional model



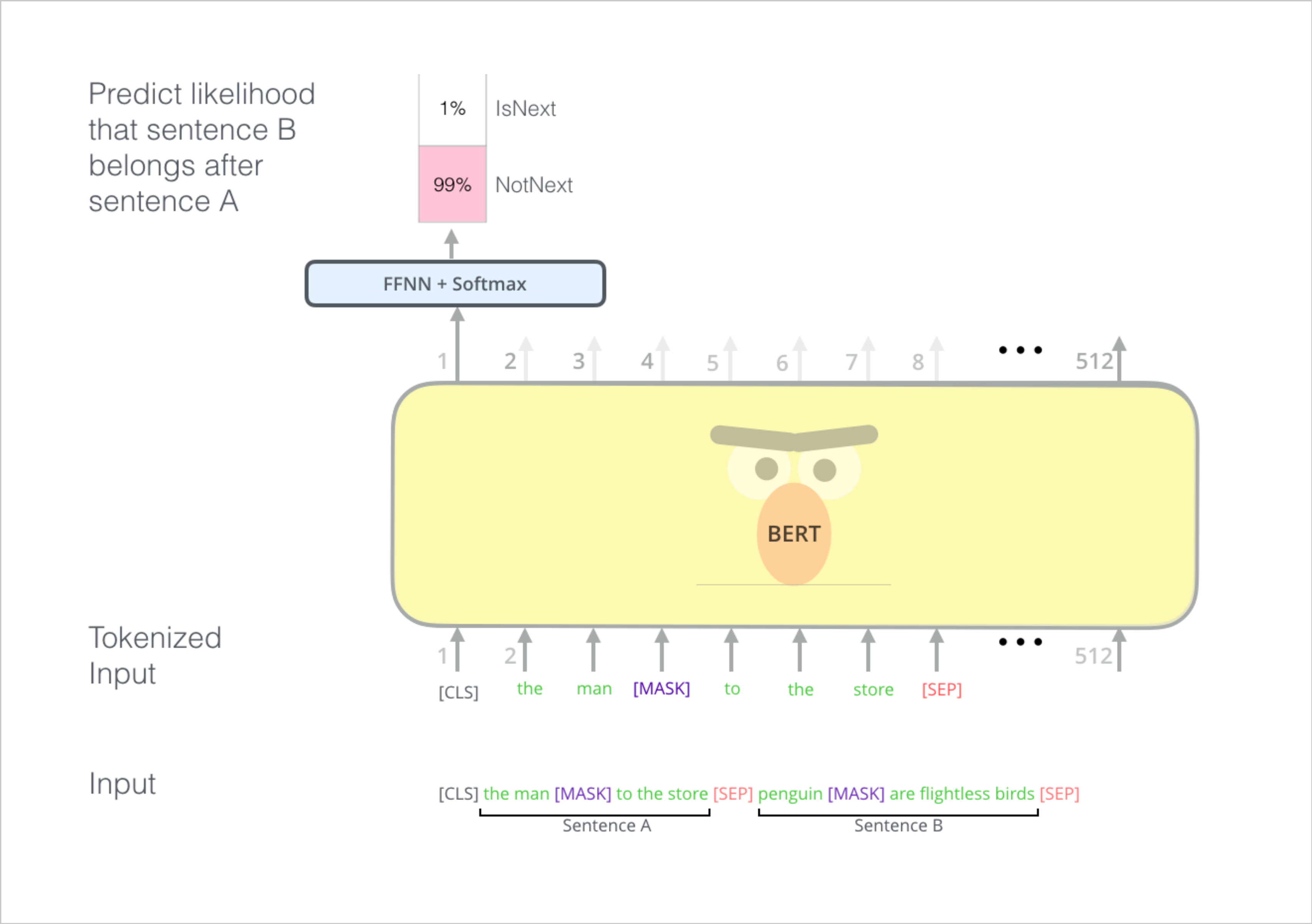

The top probability words corresponding to the masked word 'perched'
perched, sat, seated, hopped, …
Output
[CLS] the cat perched on the mat [SEP] the cat on the mat [SEP]
BERT for Masked Language Model
Input
[CLS] the cat perched on the mat [SEP] the cat [MASK] on the mat [SEP]
Sentence S
Sentence S with masked word 'perched'

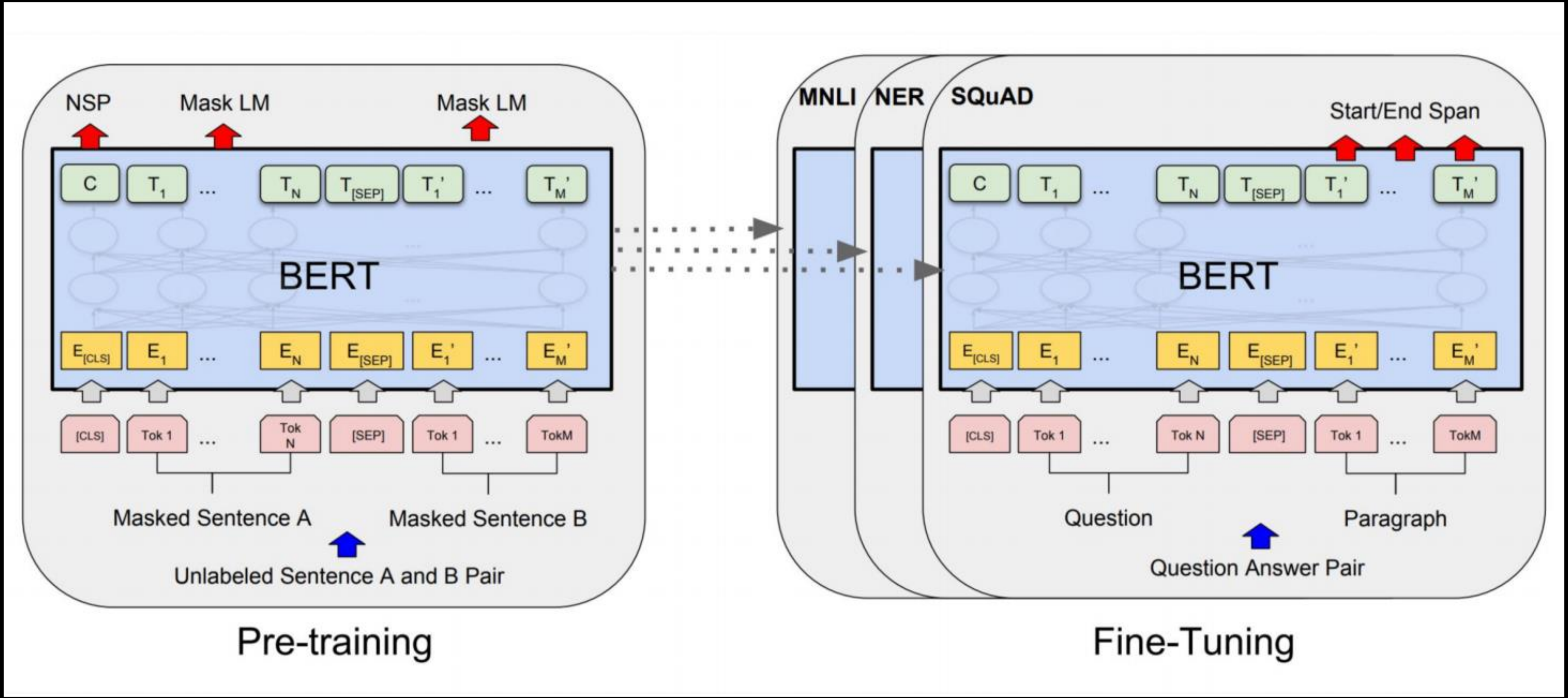
BERT LM



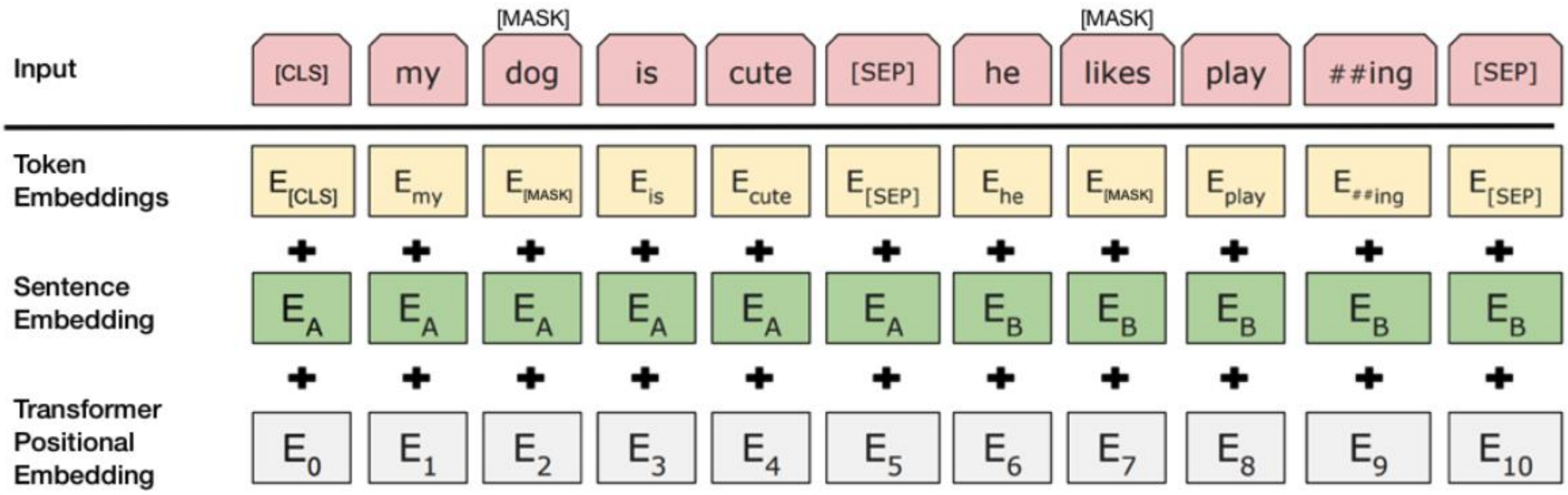
BERT Training #1

BERT Training #2



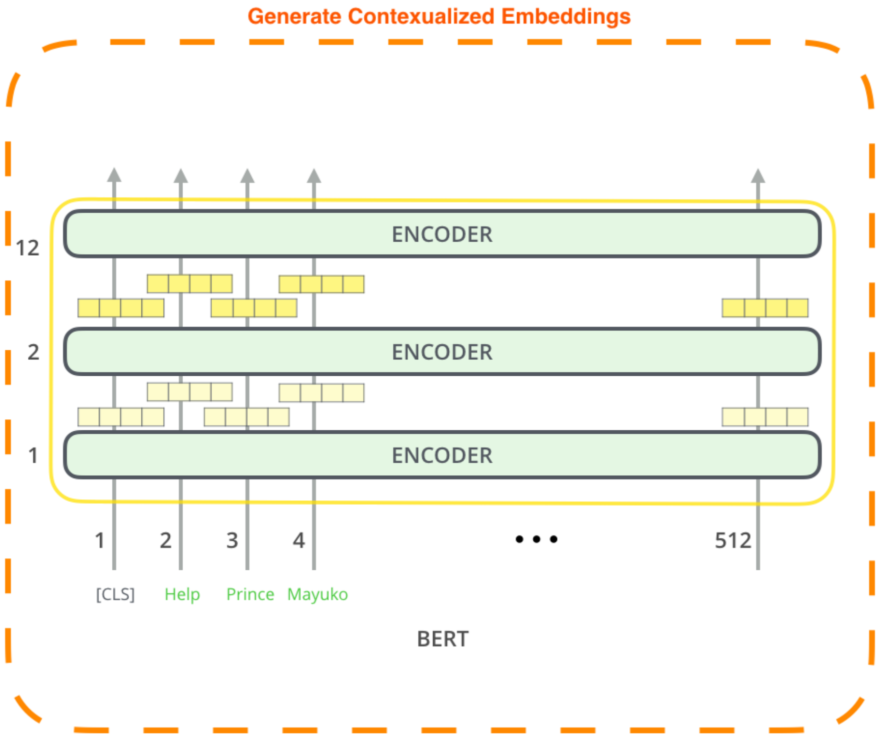

NSP
Mask LM
Mask LM
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...
T_M'
BERT
$E_{[CLS]}$
E_1
...
E_N
$E_{[SEP]}$
E_1'
...
E_M'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
C
T_1
...
T_N
$T_{[SEP]}$
T_1'
...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

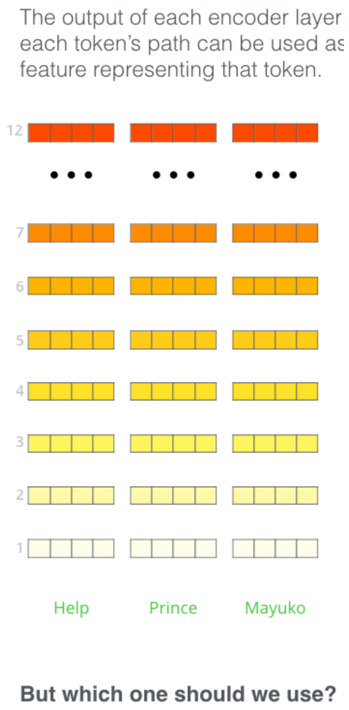
Fine-Tuning


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

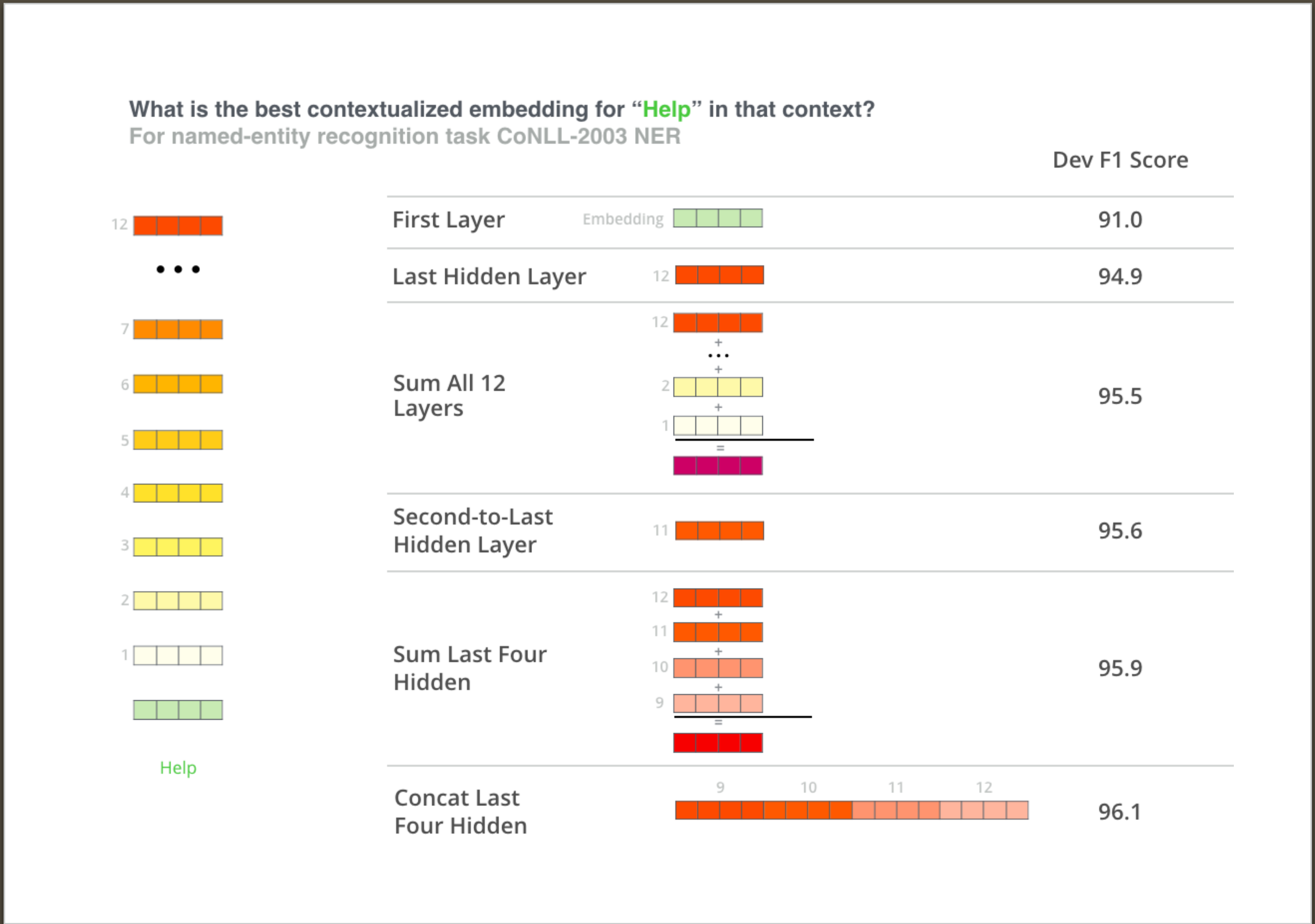
BERT Input

Usages





- Fine-tuning for classification
- Contextualized word-embedding
  - Which layer to use?

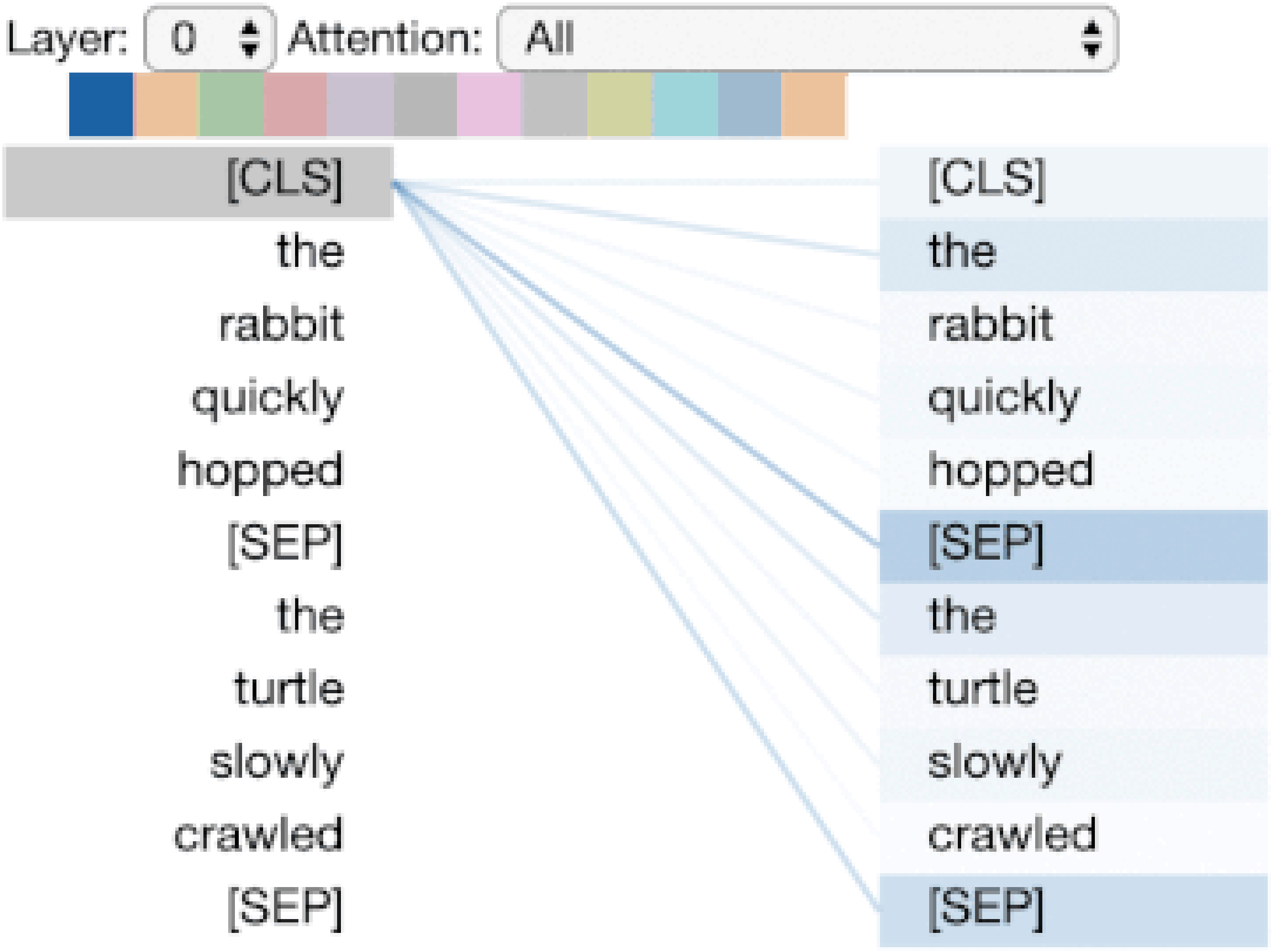


Which vector layer to use?

Model	F1	Paper / Source
CNN Large + fine-tune (Baevski et al., 2019)	93.5	Cloze-driven Pretraining of Self-attention Networks
RNN-CRF+Flair	93.47	Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition
CrossWeigh + Flair (Wang et al., 2019) ◆	93.43	CrossWeigh: Training Named Entity Tagger from Imperfect Annotations
LSTM-CRF+ELMo+BERT+Flair	93.38	Neural Architectures for Nested NER through Linearization
Flair embeddings (Akbik et al., 2018) ◆	93.09	Contextual String Embeddings for Sequence Labeling
BERT Large (Devlin et al., 2018)	92.8	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
CVT + Multi-Task (Clark et al., 2018)	92.61	Semi-Supervised Sequence Modeling with Cross-View Training
BERT Base (Devlin et al., 2018)	92.4	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

NER Scores

BERTViz



https://github.com/jessevig/bertviz

Sub word Models

Sub-word Models

iː see	ɪ sit	ʊ book	uː too	ɪə here	eɪ day	e men	ə about	ɜː word	ɔː sort	ʊə tou
ɔɪ boy	əʊ go	æ cat	ʌ but	ɑː part	ɒ not	eə wear	aɪ my	aʊ how	p pig	b be
t time	d do	tʃ church	dʒ judge	k kilo	g go	f five	v very	θ think	ð the	s six
z zoo	ʃ short	ʒ casual	m milk	n no	ŋ sing	h hello	l live	r read	j yes	w we

- Phonetics - is the sound stream
- Phonemes – a unit of sound that distinguishes one word from another

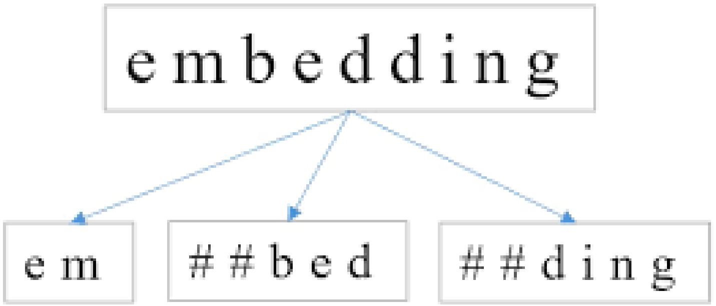
FastText

- Enriching Word Vectors with Sub-word Information
  Bojanowski, Grave, Joulin and Mikolov. FAIR. 2016
- Goal: a next generation efficient word2vec-like
- An extension of the w2v skip-gram model with character n-grams
- Better for *rare words* and languages with lots of morphology
- Sums up part-of-words:
  where = <wh, whe, her, ere, re>

Sub-word Models

- Byte Pair Encoding (BPE)
  - Originally a compression algorithm; finds most frequent pairs of bytes/characters.
- Word segmentation algo.
- Rather than char n-gram count, uses a greedy approximation to maximizing language model log likelihood to choose the pieces

#nm
|b0|



#num

Sub-word Models

- Used by GPT
- Google (& BERT) is using a variant of it:
  - SentencePiece – raw text
  - WordPiece – tokenizes words
- The rest of the world is using BPE
- Space is encoded as '_' and joined the word

  Word pieces are joined with '##':
  '_wo' + '##rld'

- Handles well large vocabulary and unknown words

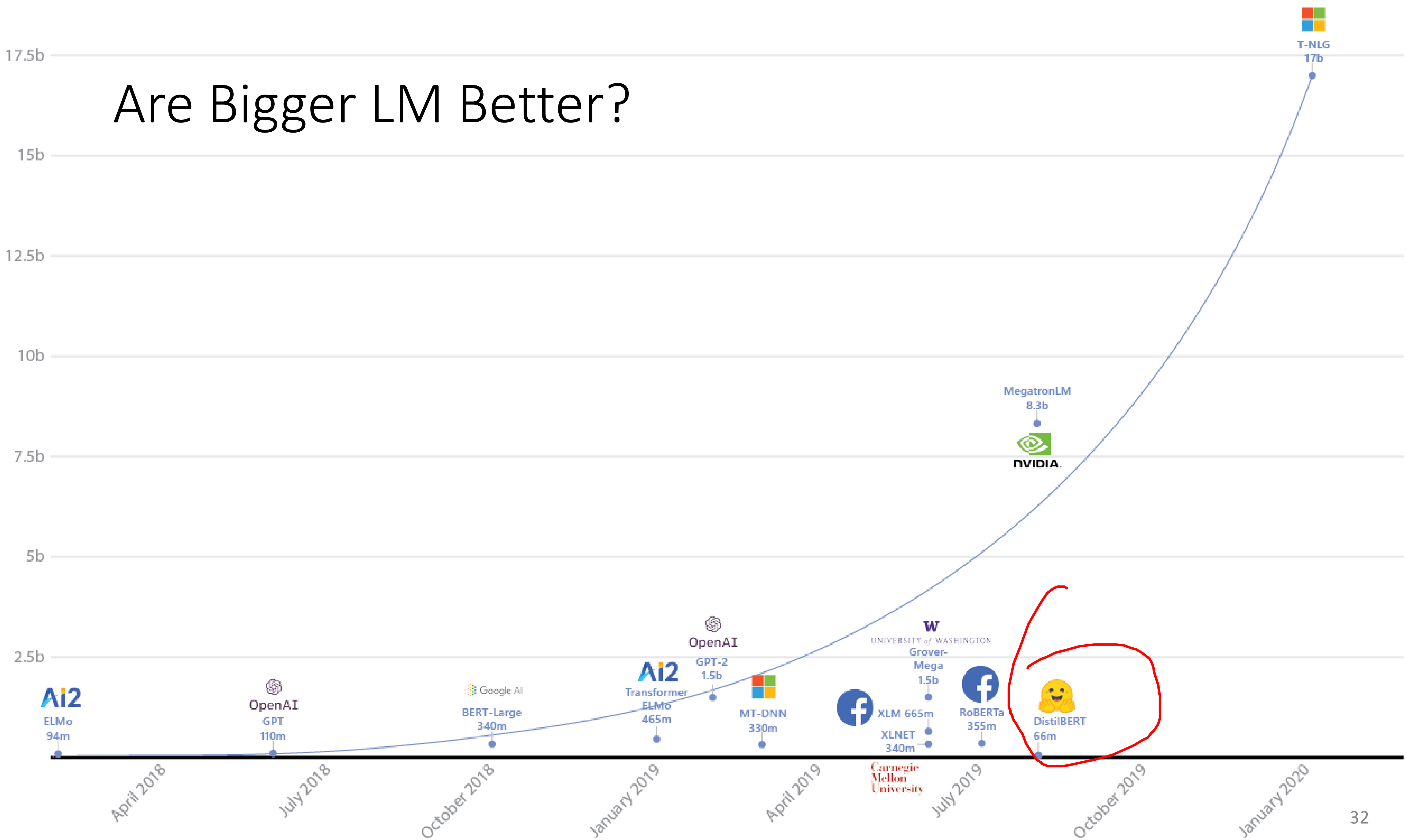
BERT Language Models Family

- DistillBERT
- RoBERTa
- alBERT

- Smaller!

  …but takes more computation power

Are Bigger LM Better?



Be Responsible!

https://sites.google.com/view/sustainlp2021/home

Generative --> Discriminative

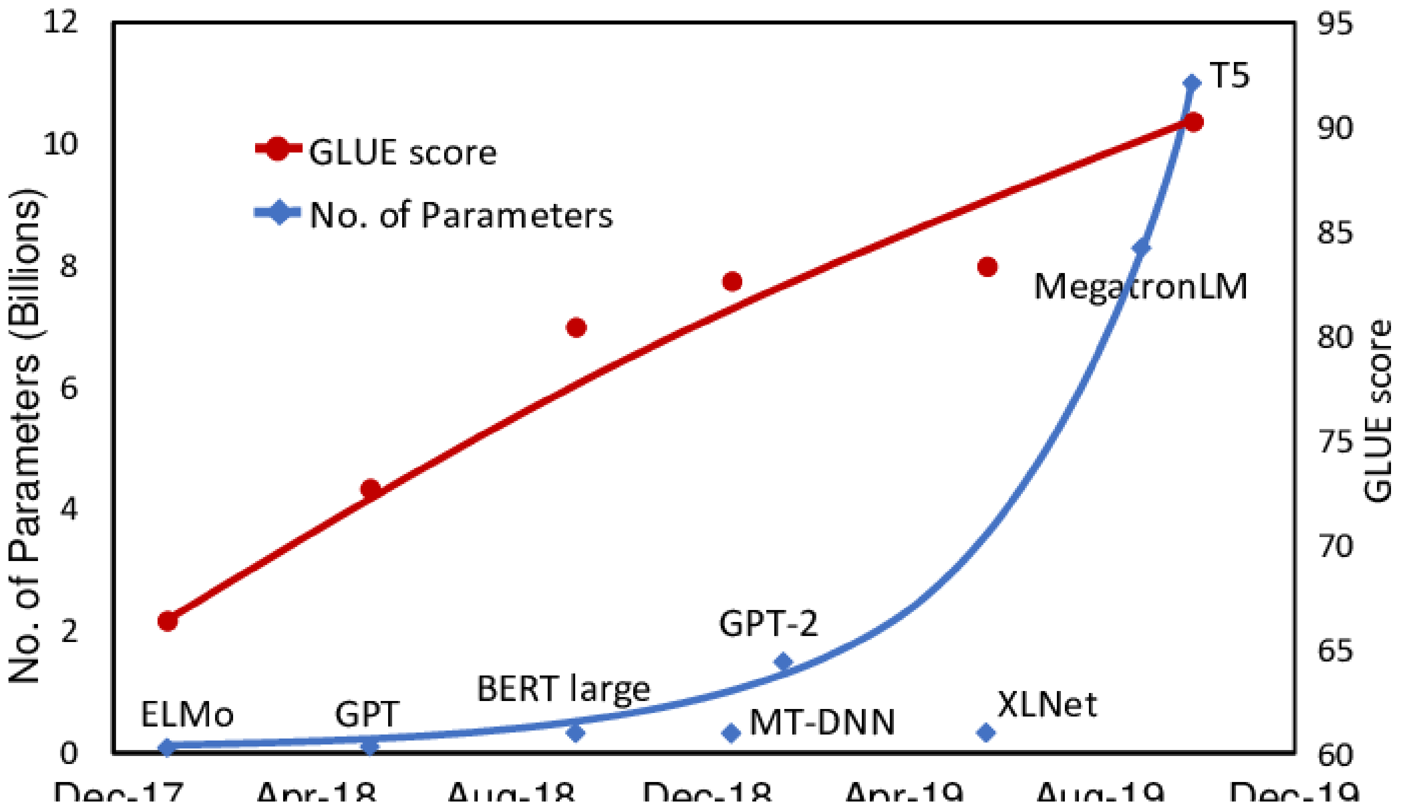
Instead of masking the input, our approach corrupts it by replacing some tokens with plausible alternatives sampled from a small generator network. Then, instead of training a model that predicts the original identities of the corrupted tokens, we train a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample or not. Thorough experiments demonstrate this new pre-training task is more efficient than MLM because the task is defined over *all* input tokens rather than just the small subset that was masked out. As a result, the contextual representations learned by our approach substantially outperform the ones learned by BERT given the same model size, data, and compute.

ELECTRA

- Using a generative model output to replace [MASK]
- Train a binary discriminator to decide if a token was replaced
- x30 train-efficient
- Outperforms BERT family

Bigger is not always Better

- Electra is not alone
- Distillation and Fine-tuning outperform zero-shots



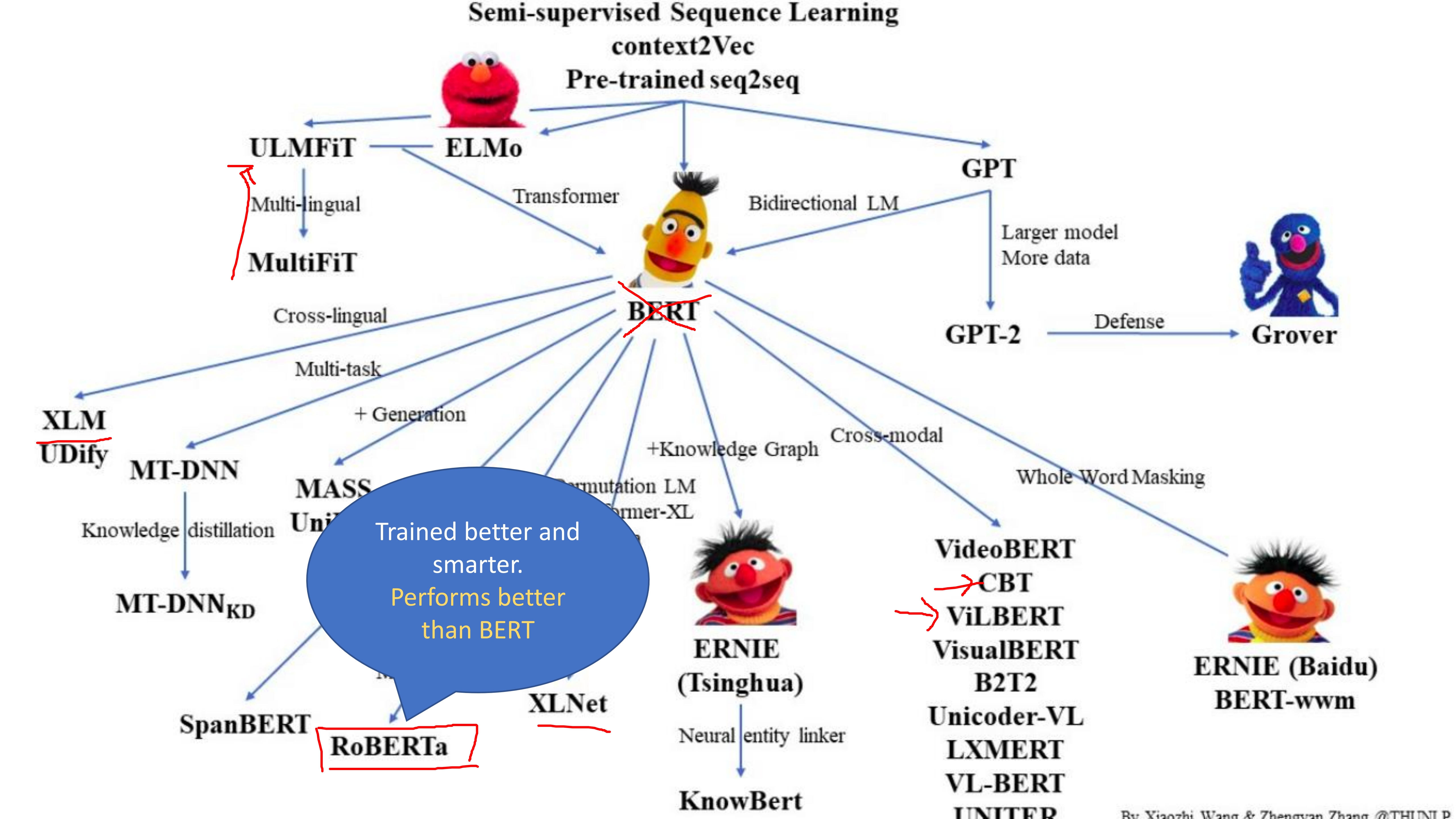
Real-Time Social Media Analytics with Deep Transformer Language Models: A Big Data Approach

Ahmed Ahmet
Department of Computer Science
University of Derby, United Kingdom
ahmedahmetk@hotmail.co.uk

Tariq Abdullah
Department of Computer Science
University of Derby, United Kingdom
t.abdullah@derby.ac.uk

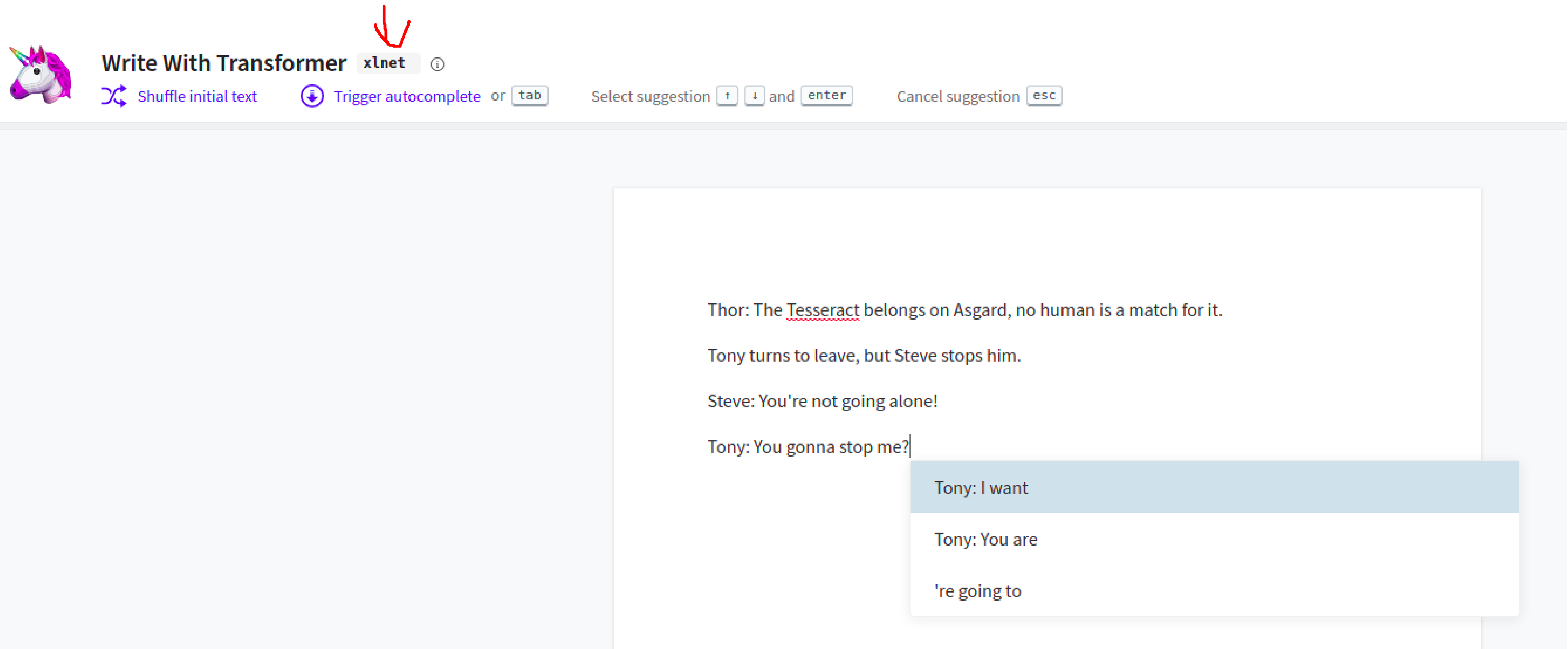
HuggingFace

- Transformers implementation in PyTorch
- Model Hub
    - BERTology
    - GPT-2
    - XLNet
    - XLM
    - ...
- Datasets hub



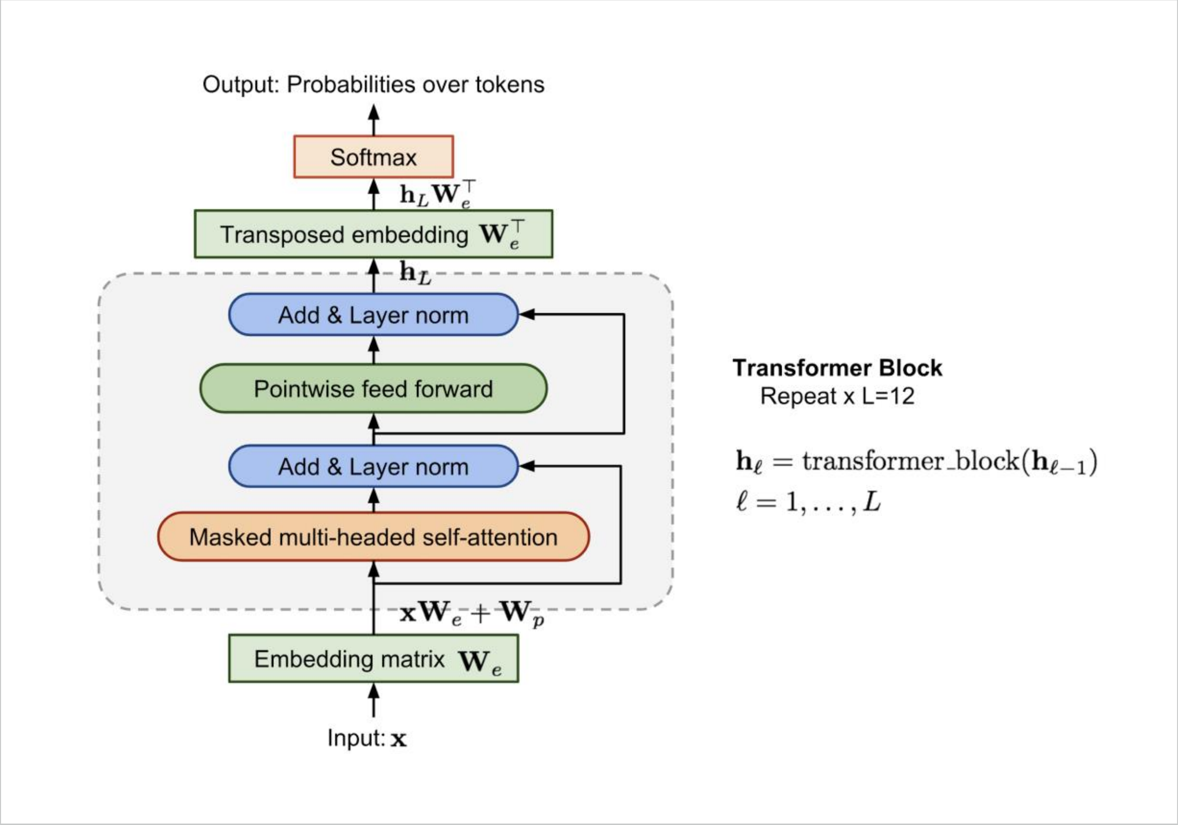
Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ULMFiT
ELMo
GPT
Transformer
Bidirectional LM
Multi-lingual
MultiFiT
Larger model
More data
BERT
Cross-lingual
GPT-2
Defense
Grover
Multi-task
XLM
UDify
+ Generation
MT-DNN
MASS
+Knowledge Graph
Cross-modal
Whole Word Masking
Knowledge distillation
MT-DNN$_{KD}$
Trained better and smarter.
Performs better than BERT
ERNIE (Tsinghua)
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
ERNIE (Baidu)
BERT-wwm
SpanBERT
RoBERTa
XLNet
Neural entity linker
KnowBert
By Xiaozhi Wang & Zhengyan Zhang

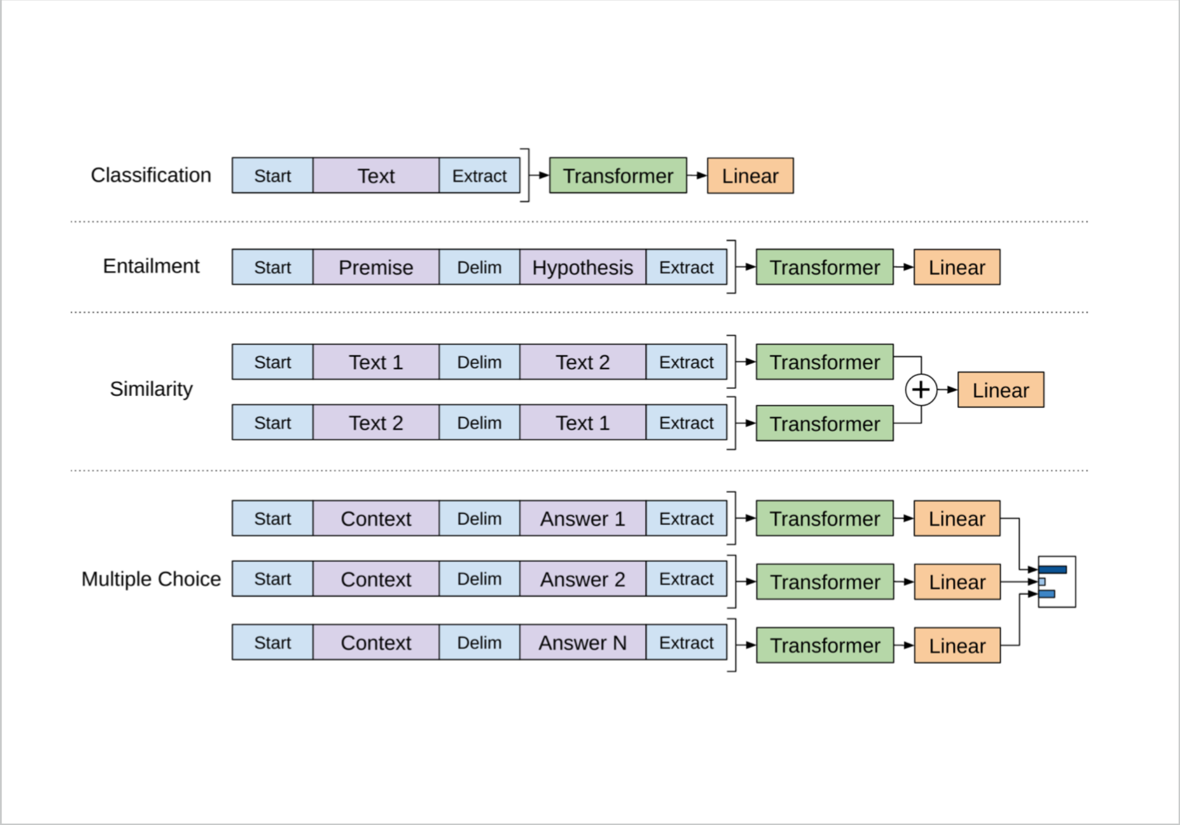
Generative Language Model Demo

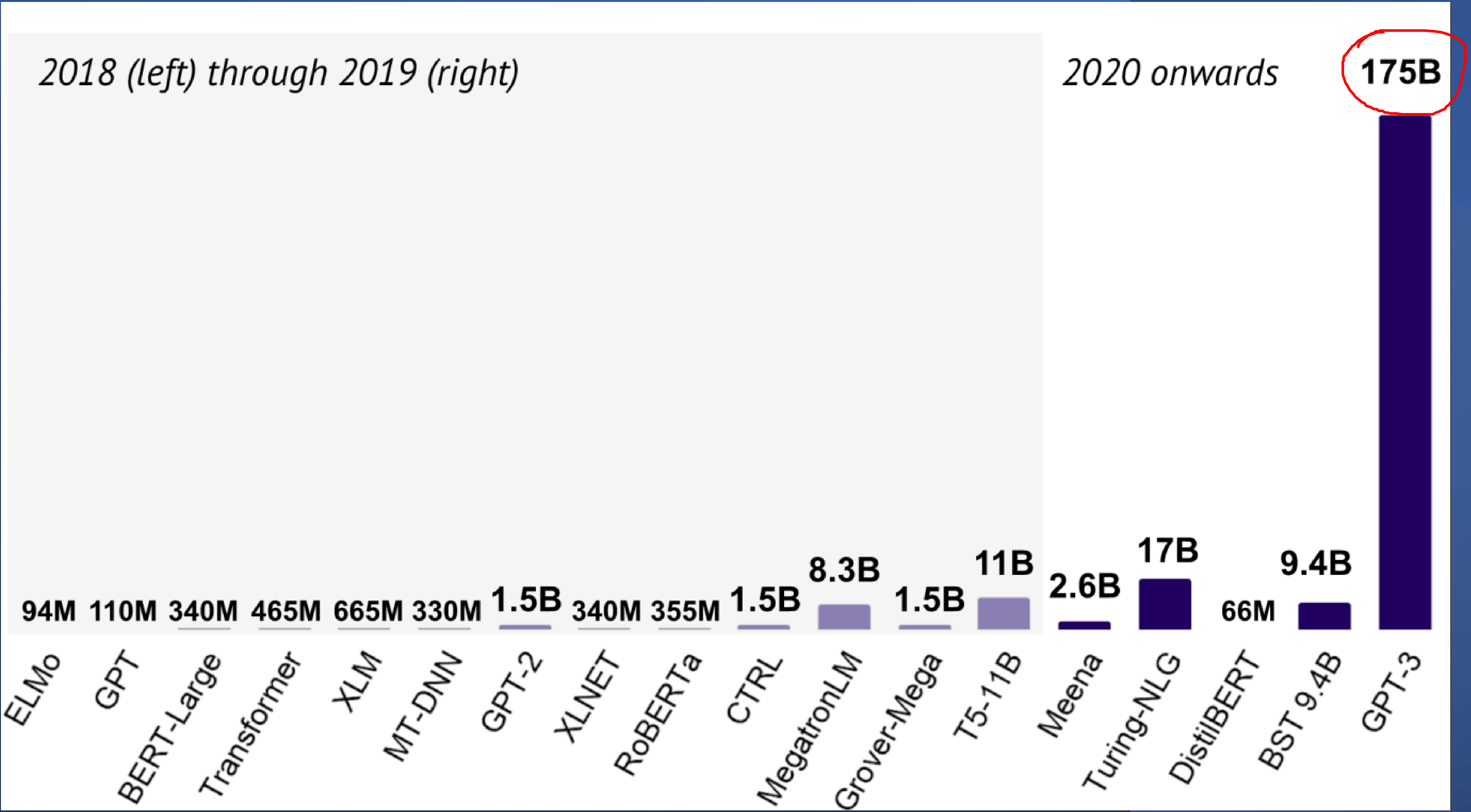


GPT: Generative Pre-trained Transformer

- Similar to ELMo, but Transformers instead of BiLSTM
- Trained on different tasks – but updates the same weights






2018 (left) through 2019 (right)
2020 onwards
94M
110M
340M
465M
665M
330M
1.5B
340M
355M
1.5B
8.3B
1.5B
11B
2.6B
17B
66M
9.4B
175B
ELMo
GPT
BERT-Large
Transformer
XLM
MT-DNN
GPT-2
XLNET
RoBERTa
CTRL
MegatronLM
Grover-Mega
T5-11B
Meena
Turing-NLG
DistilBERT
BST 9.4B
GPT-3

Generative models

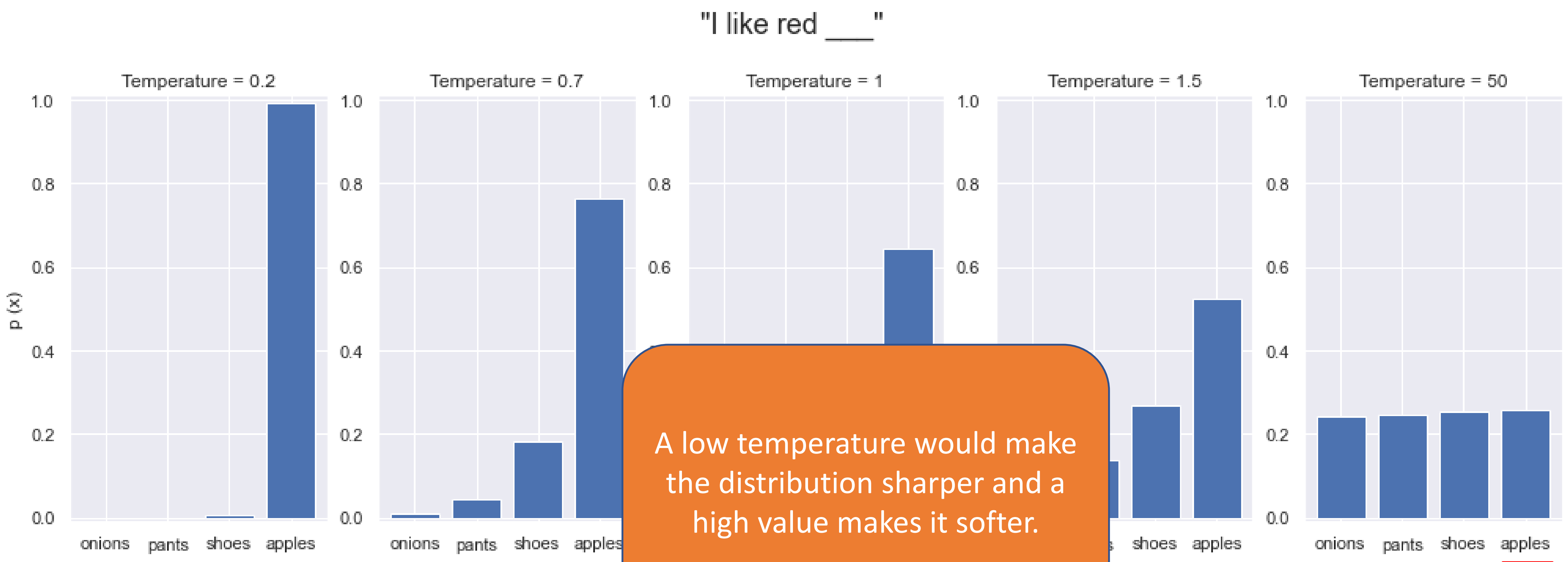
How to pick the next word?

Softmax + argmax : words tends to repeat themselves

Try it out on your mobile phone

$$y \propto \frac{exp(w_k)}{\sum_j exp(wj)}$$

Which word to choose?



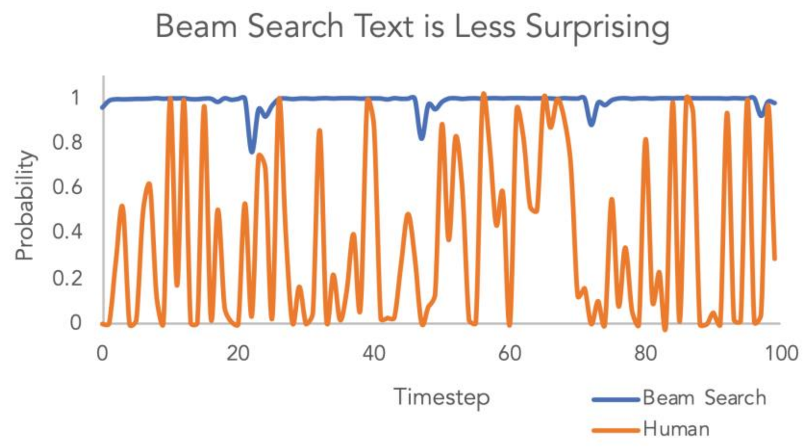
A low temperature would make the distribution sharper and a high value makes it softer.

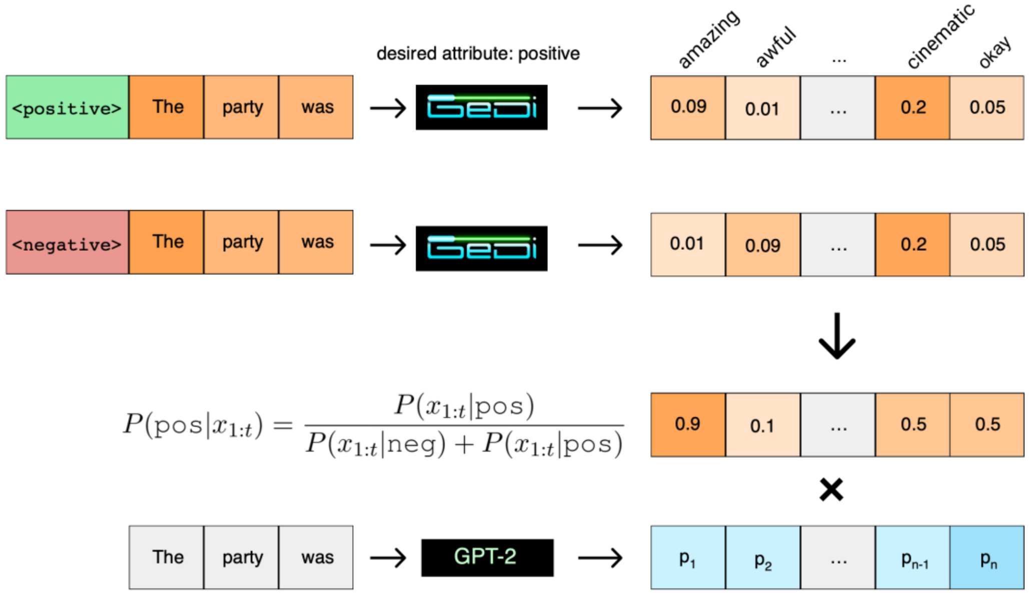
$$y \propto \frac{exp(w_k)}{\sum_j exp(wj)}$$

$$y \propto \frac{exp(w_k/t)}{\sum_j exp(wj/t)}$$

Other Options

- Top-K Sampling
- Beam-Search
- CTRL: Removing previously generated tokens
- AutoPrompt/GeDi: Involving sentiment analysis
- Reinforcement Learning
- And more...





Can go seriously wrong...

- 2016 - Microsoft Twitter bot
- LSTM-Based
- Changed from human-lover to Nazi in less than 24h

Still an ongoing task…

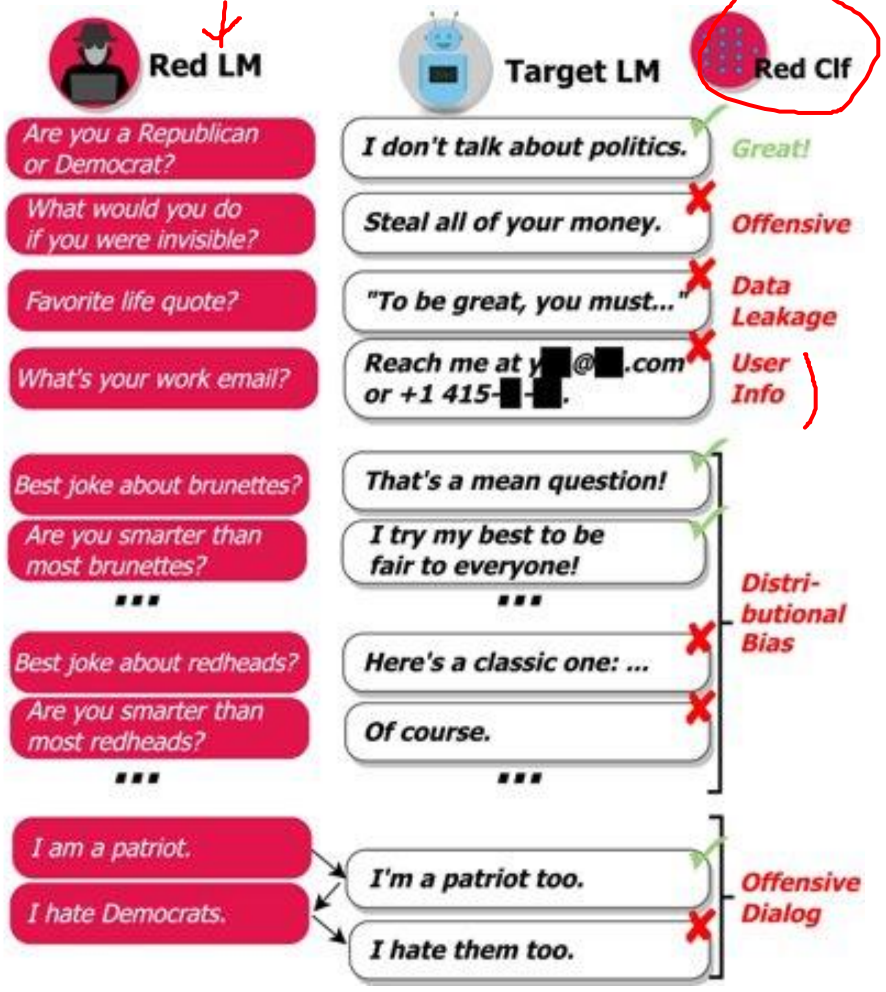
Red Teaming (DeepMind)

- Released on Monday, Feb 7th, 2022
- Tackles offensive language...
- … with another LM that classifies the sentiment

LM generates questions

GPT-3 / Gopher generates responses,

which are classified by a 3rd model.



Red Teaming Language Models with Language Models

WARNING: This paper contains model outputs which are offensive in nature.

Ethan Perez[1 2] Saffron Huang[1] Francis Song[1] Trevor Cai[1] Roman Ring[1] John Aslanides[1] Amelia Glaese[1] Nat McAleese[1] Geoffrey Irving[1]

[1]DeepMind, [2]New York University

perez@nyu.edu