# Transfer Learning in NLP

Natural Language Processing

Vladislav Lialin, Text Machine Lab

March 15, 2022

# Administrative

- Machine translation homework is due on Thursday before the class
  - If you haven't started working already, you are probably in trouble.
    DM me on Discord if you need help
  - Make sure your GPU setup works ASAP
- Midterm next week, **on March 27**
  - Interview-like, but you will have the chance for open-book preparation before you answer
  - **Only hand-written materials** are allowed for open-book preparation
  - You need to know everything from the midterm program
  - Use your lecture notes and slides to prepare
  - Additionally:
    - **For neural networks: the Deep Learning Book — free version on the website**
    - **NLP: Stanford CS224N lecture notes — free**
    - NLP: Natural Language Processing with Transformers — $15 for rental, $40 for printed
      this book is optional, stanford notes + deep learning book are all you need

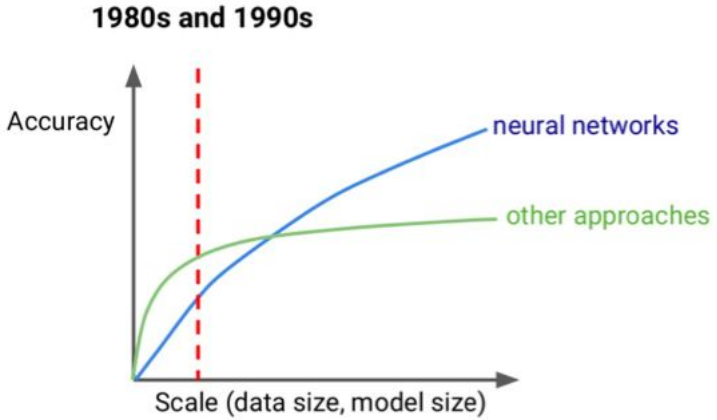
# What you should remember after this lecture

- Transfer learning
- Structuring input text for transformers
- Masked language modeling

# Transfer learning

# Why has deep learning been so successful?

- Better “tricks”
  - dropout, improved optimizers (e.g., Adam), batch norm, attention
- Better hardware (thanks video games!) -> larger models
- Larger datasets




Slide credit: Stanford cs224n

# Big Deep Learning Success

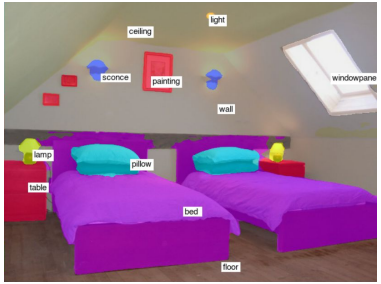
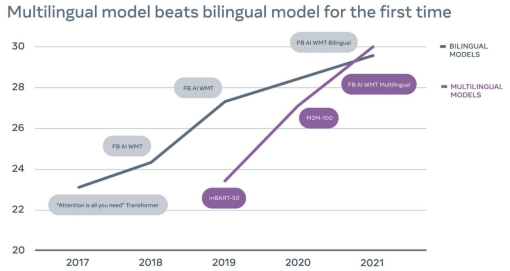



Image Recognition




Machine Translation

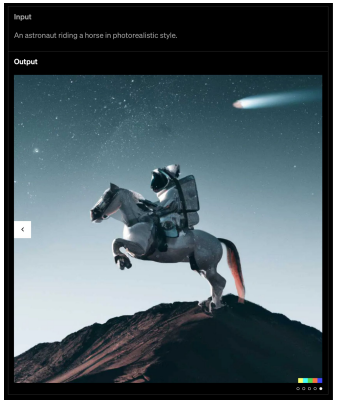



Image Generation


Game Playing

```python
import torch
import torch.nn as nn


class MultiHeadAttention(nn.Module):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.d_model = d_model
        self.num_heads = num_heads
        self.depth = d_model // num_heads

        assert (self.depth * num_heads == d_model), "d_model must be divisible by num_heads"

        self.query = nn.Linear(d_model, d_model)
        self.key = nn.Linear(d_model, d_model)
        self.value = nn.Linear(d_model, d_model)

        self.fc = nn.Linear(d_model, d_model)
```

Code Generation

Dialogue

# NLP dataset sizes

| Dataset (English) | Size (# sentences) |
|---|---|
| NER | 15K (CoNLL 2003) |
| Coreference Resolution | 75K (OntoNotes) |
| Parsing | 40K (Penn Treebank) |
| Question Answering | 100k questions (SQuAD) |
| Textual Entailment | 570k (SNLI) |
| Sentiment Analysis | 10k (SST) |

Slide credit: Stanford cs224n

# … and that's for core tasks in **English**!

| Dataset (English) | Size (# sentences) |
|---|---|
| NER | 15K (CoNLL 2003) |
| Coreference Resolution | 75K (OntoNotes) |
| Parsing | 40K (Penn Treebank) |
| Question Answering | 100k questions (SQuAD) |
| Textual Entailment | 570k (SNLI) |
| Sentiment Analysis | 10k (SST) |

Slide credit: Stanford cs224n

# … and that's for core tasks in **English**!

- Most tasks have less data
- There are thousands of languages
  - Hundreds with > 1 million native speakers
  - Less than 10% of people speak English as their first language
  - Little-to-no annotated data for many other languages

Slide credit: Stanford cs224n

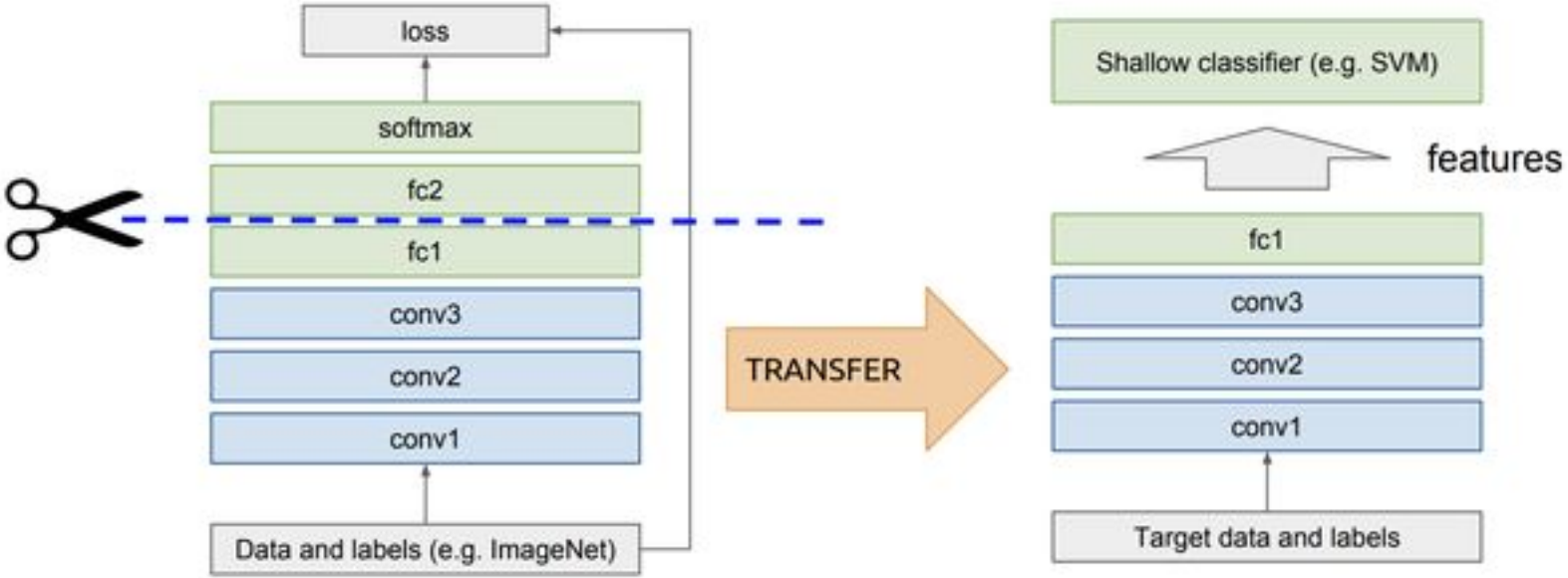
# What to do?

- Just collect more data?
  - Expensive!
    - Crowdsourcing alleviates this a bit
  - Requires linguistic knowledge for some tasks

- Semi-Supervised Learning
  - Use **unlabeled** examples during training
  - Easy to find for NLP!

Slide credit: Stanford cs224n

# Transfer Learning in Computer Vision



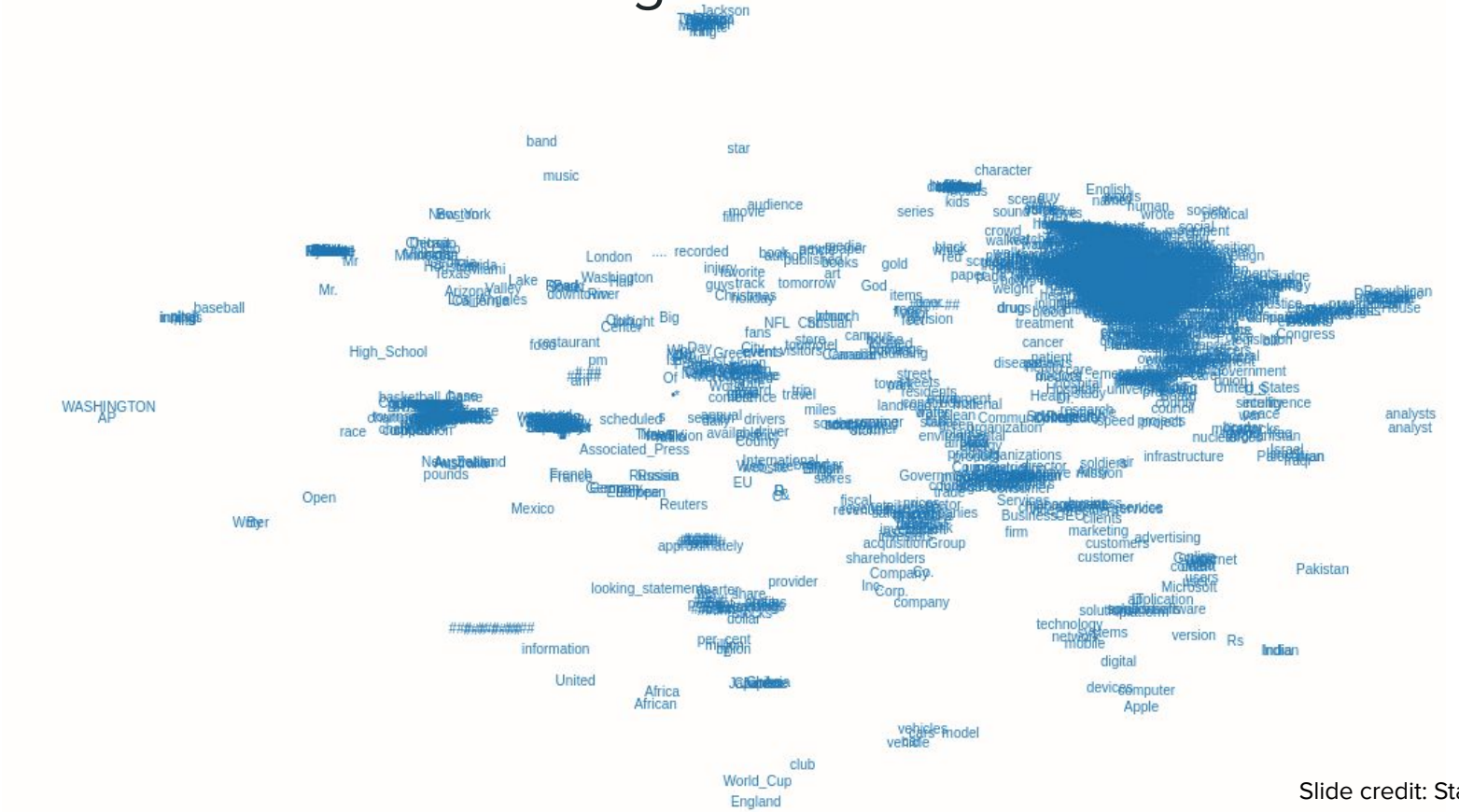

Slide credit: Stanford cs224n

# Transfer Learning in NLP

- Pre-2018: Shallow representations
- 2018: Deep fixed representations
- 2018+: Deep model fine-tuning
- 2022+: In-context learning with language models (topic of the next lecture)

Slide credit: Stanford cs224n

# Pre-2018: Word Embeddings




Slide credit: Stanford cs224n

# Pre-2018: Word Embeddings

- Most popular type of semi-supervised learning in NLP at the time
- **Easy and fast to train**
- Computationally cheap
- Easily accessible
  (libraries, pre-trained models)

- Strongly domain-dependent
- Shallow
  (Mostly captures semantic features)
- Only allow a single context-independent representation for each word
- Limited capacity
  (If you have enough data, it is almost always better to train your own embeddings from scratch)

2018

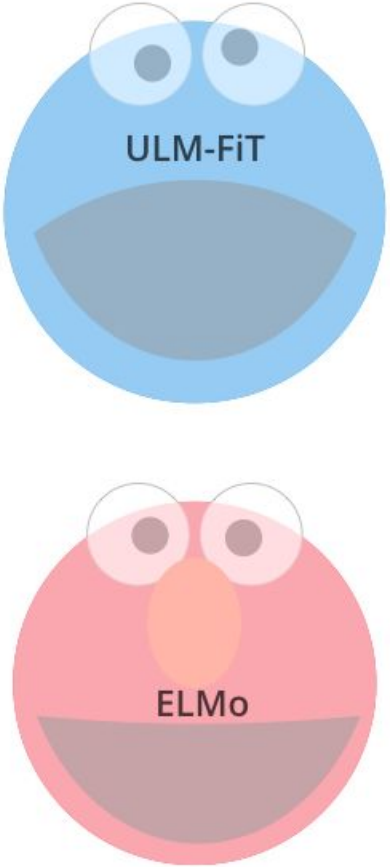




Image credit: Jay Alammar
jalammar.github.io/illustrated-bert

# ELMo: Embeddings from Language Models


Image credit: Jay Alammar
jalammar.github.io/illustrated-bert

# ELMo

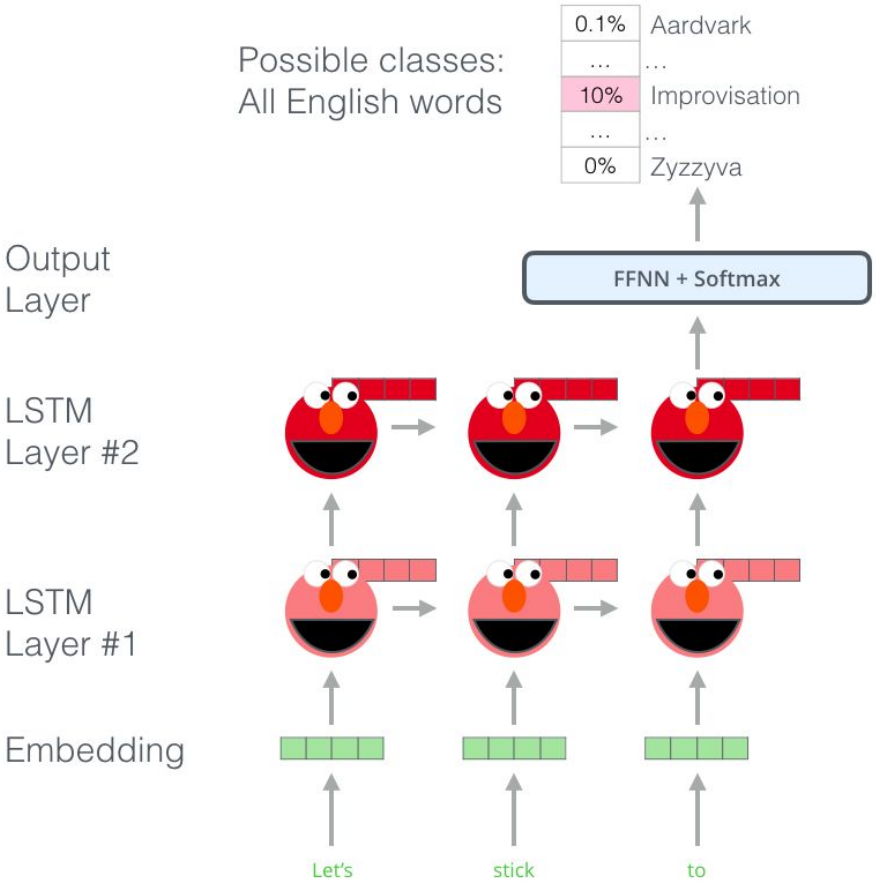



Image credit: Jay Alammar
jalammar.github.io/illustrated-bert

# ELMo = contextual word embedding + architecture on top

1. Extract deep contextualized embeddings for every token in the sentence
2. Use these embeddings instead of randomly initialized embeddings in a neural network that fits your task
3. Only tune 3 scalar weights (s0, s1, and s2) in ELMo, not the model parameters

# ELMo: Comments from the Original Paper

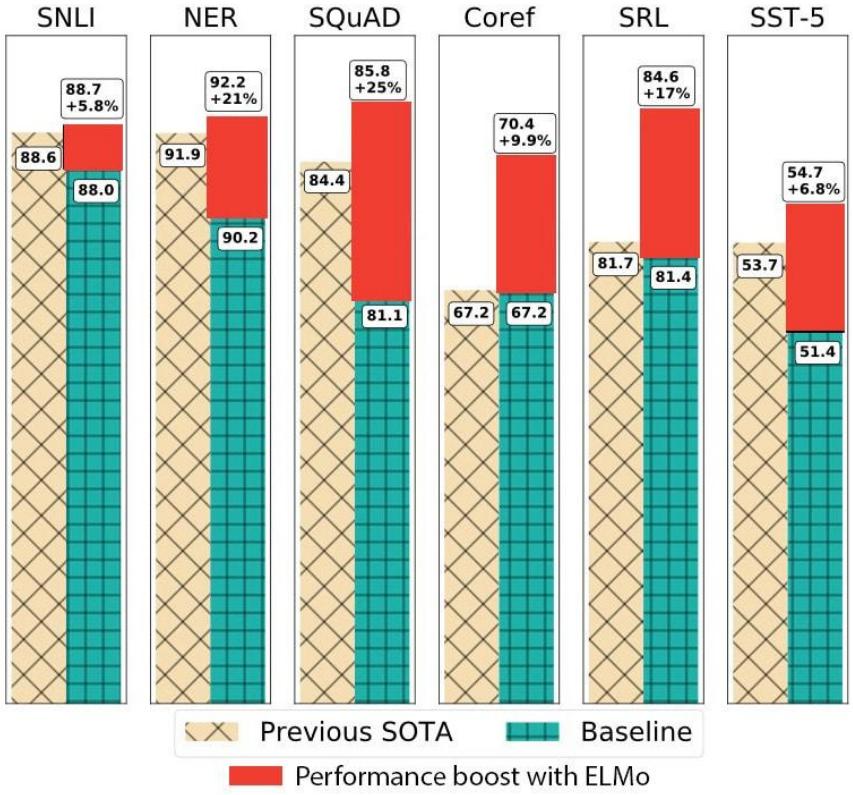
"Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a **function of the entire input sentence**"

"Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a **function of all of the internal layers** of the biLM"

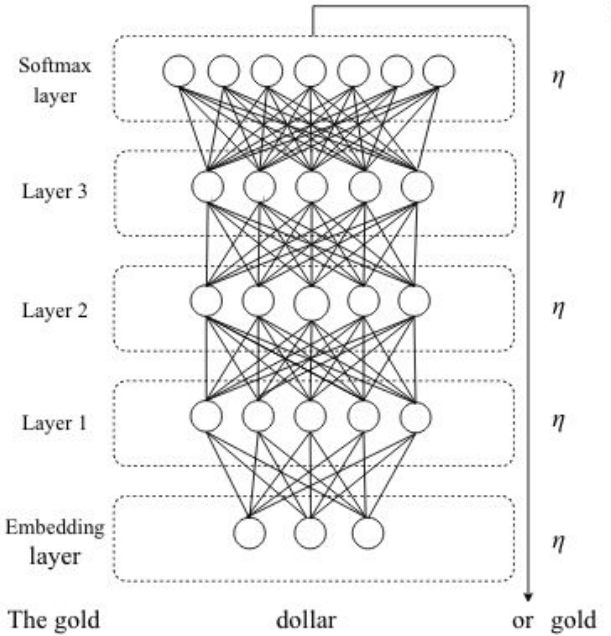
"Can be easily **added to existing models** for six diverse and challenging language understanding problems"

# ELMo: results

SOTA on **6** NLP tasks

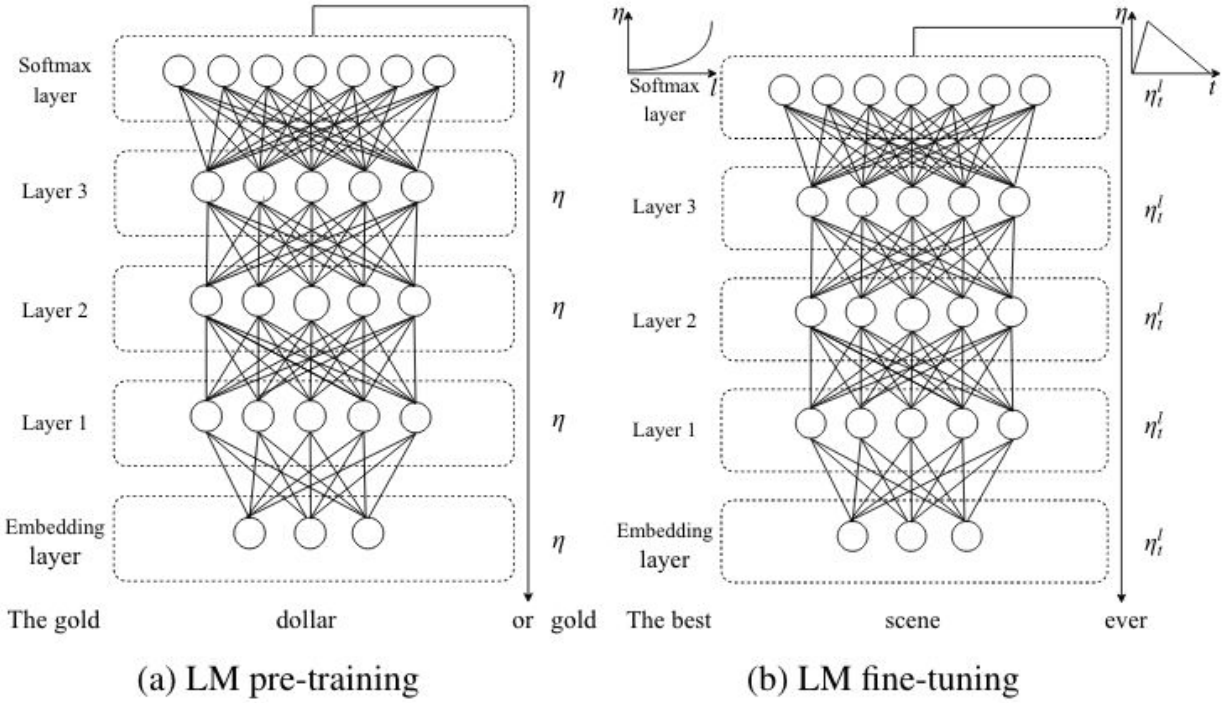


# ULMfit: Universal Language Model Fine-Tuning for Text Classification

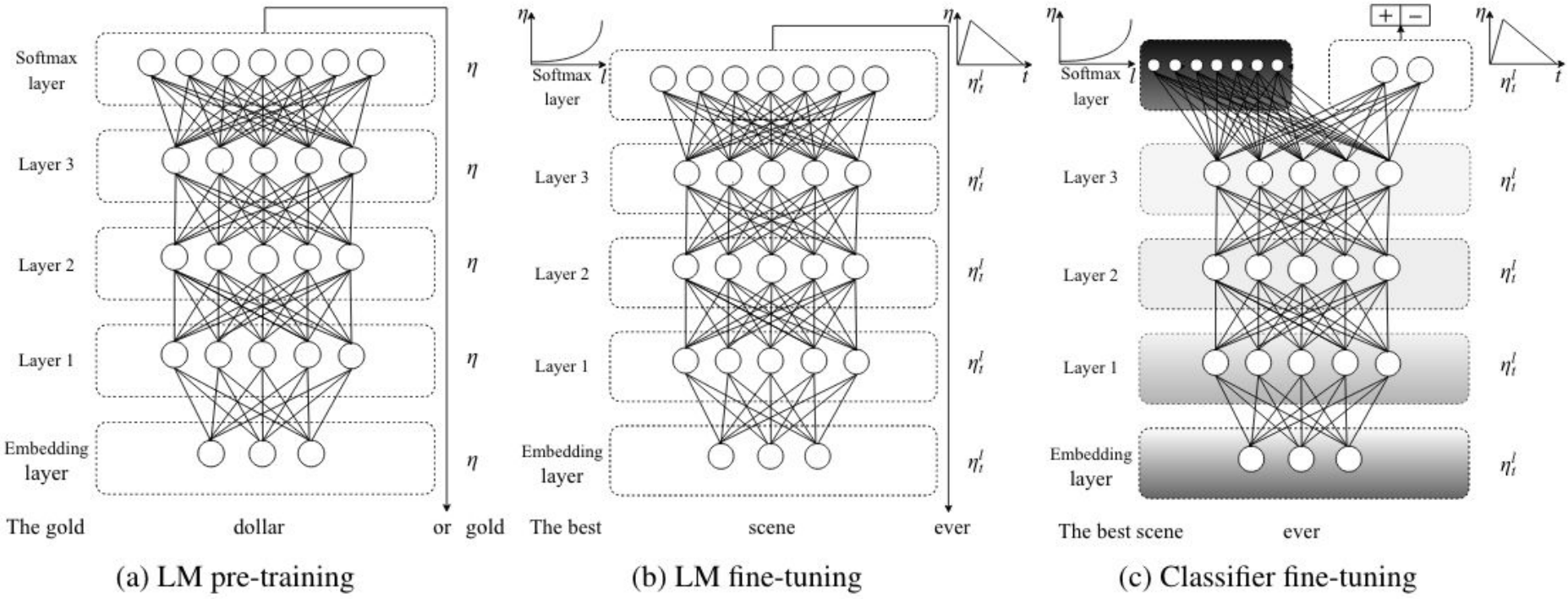



(a) LM pre-training

# ULMfit: Universal Language Model Fine-Tuning for Text Classification



# ULMfit: Universal Language Model Fine-Tuning for Text Classification



# ULMfit: Results

SOTA on **6 classification** tasks

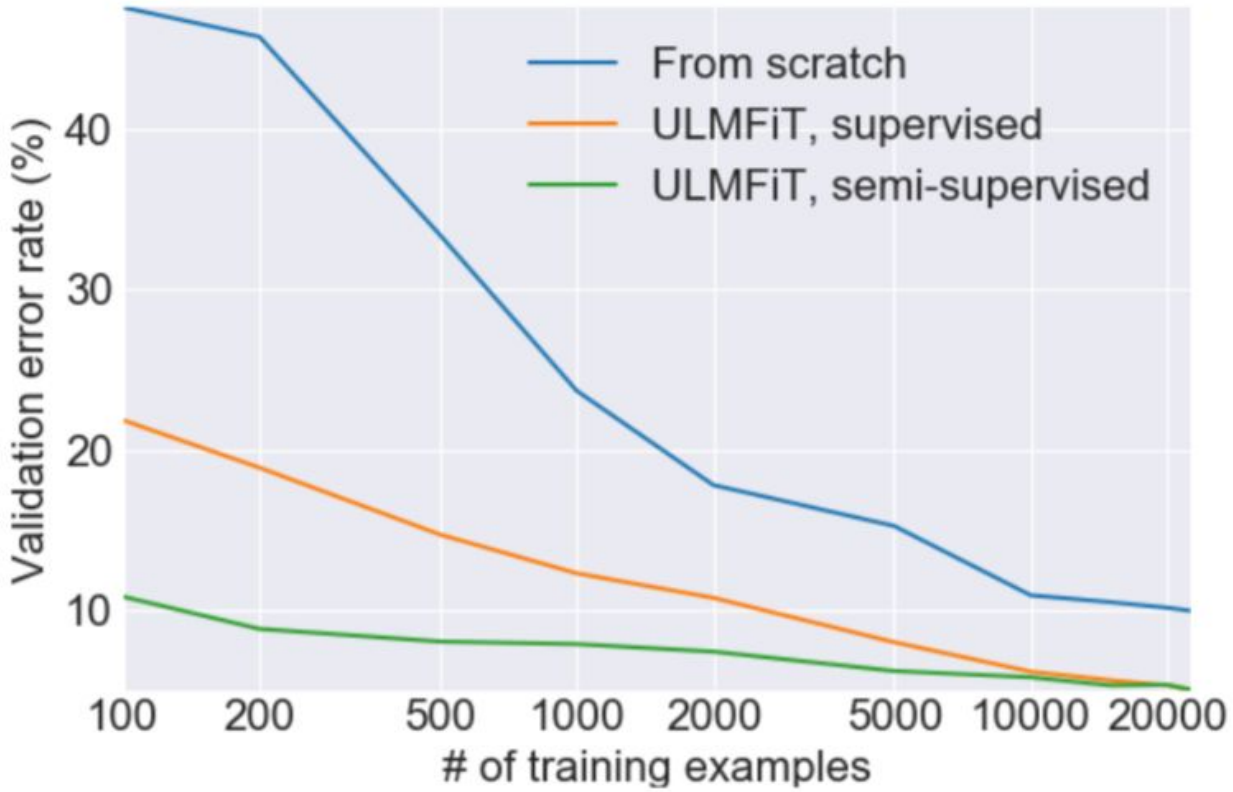
| | Model | Test | | Model | Test |
|---|---|---|---|---|---|
| IMDb | CoVe (McCann et al., 2017) | 8.2 | TREC-6 | CoVe (McCann et al., 2017) | 4.2 |
| | oh-LSTM (Johnson and Zhang, 2016) | 5.9 | | TBCNN (Mou et al., 2015) | 4.0 |
| | Virtual (Miyato et al., 2016) | 5.9 | | LSTM-CNN (Zhou et al., 2016) | 3.9 |
| | ULMFiT (ours) | **4.6** | | ULMFiT (ours) | **3.6** |

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

| | AG | DBpedia | Yelp-bi | Yelp-full |
|---|---|---|---|---|
| Char-level CNN (Zhang et al., 2015) | 9.51 | 1.55 | 4.88 | 37.95 |
| CNN (Johnson and Zhang, 2016) | 6.57 | 0.84 | 2.90 | 32.39 |
| DPCNN (Johnson and Zhang, 2017) | 6.87 | 0.88 | 2.64 | 30.58 |
| ULMFiT (ours) | **5.01** | **0.80** | **2.16** | **29.98** |

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

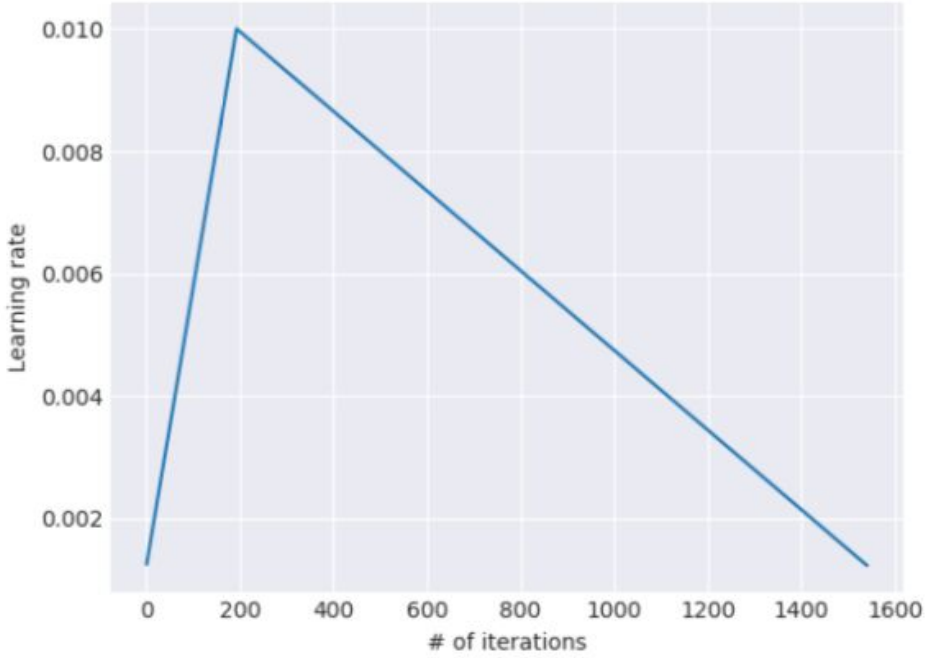
# ULMfit: Results



# ULMfit: Tricks

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})]$$



# ULMfit: Tricks

| LM | IMDb | TREC-6 | AG |
|---|---|---|---|
| Vanilla LM | 5.98 | 7.41 | 5.76 |
| AWD-LSTM LM | **5.00** | **5.69** | **5.38** |

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

| LM fine-tuning | IMDb | TREC-6 | AG |
|---|---|---|---|
| No LM fine-tuning | 6.99 | 6.38 | 6.09 |
| Full | 5.86 | 6.54 | 5.61 |
| Full + discr | 5.55 | 6.36 | 5.47 |
| Full + discr + stlr | **5.00** | **5.69** | **5.38** |

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

| Classifier fine-tuning | IMDb | TREC-6 | AG |
|---|---|---|---|
| From scratch | 9.93 | 13.36 | 6.81 |
| Full | 6.87 | 6.86 | 5.81 |
| Full + discr | 5.57 | 6.21 | 5.62 |
| Last | 6.49 | 16.09 | 8.38 |
| Chain-thaw | 5.39 | 6.71 | 5.90 |
| Freez | 6.37 | 6.86 | 5.81 |
| Freez + discr | 5.39 | 5.86 | 6.04 |
| Freez + stlr | 5.04 | 6.02 | 5.35 |
| Freez + cos | 5.70 | 6.38 | **5.29** |
| Freez + discr + stlr | **5.00** | **5.69** | 5.38 |

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

## ULMfit

- RNN architecture
- Language modeling
  - Language model adaptation step
- CV-style transfer learning
  - Fine-tune the whole model
  - With a lot of extra tricks
  - Just replace the last layer of the model
  - Not obvious how to apply to tasks that are not text classification

## ELMo

- RNN architecture
- Language modeling*
- Word embeddings 2.0
  - Complex word embedding construction procedure
  - Insert embeddings into other model
  - Easy to apply to any NLP task

Comparing original approaches from the papers

# GPT: General Purpose Transformer

Main idea is very close to ULMfit,
here's the differences

- Replace RNN with a Transformer
- BPE tokens instead of words
- Format model input according to each task (we'll discuss it a bit later)

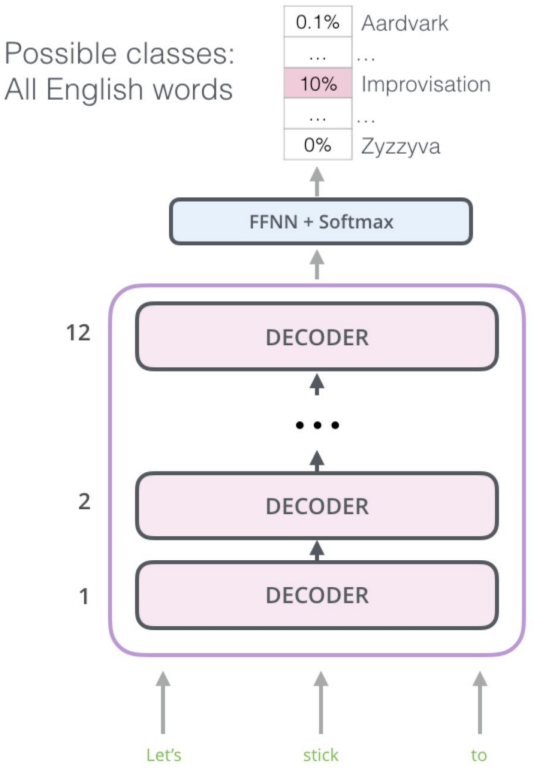



Image credit: Jay Alammar
jalammar.github.io/illustrated-bert

# GPT: Results

SOTA on **9 NLP** tasks

| Method | Classification | | Semantic Similarity | | | GLUE |
|---|---|---|---|---|---|---|
| | CoLA (mc) | SST2 (acc) | MRPC (F1) | STSB (pc) | QQP (F1) | |
| Sparse byte mLSTM [16] | - | **93.2** | - | - | - | - |
| TF-KLD [23] | - | - | **86.0** | - | - | - |
| ECNU (mixed ensemble) [60] | - | - | - | 81.0 | - | - |
| Single-task BiLSTM + ELMo + Attn [64] | 35.0 | 90.2 | 80.2 | 55.5 | 66.1 | 64.8 |
| Multi-task BiLSTM + ELMo + Attn [64] | 18.9 | 91.6 | 83.5 | 72.8 | 63.3 | 68.9 |
| Finetuned Transformer LM (ours) | **45.4** | 91.3 | 82.3 | **82.0** | **70.3** | **72.8** |

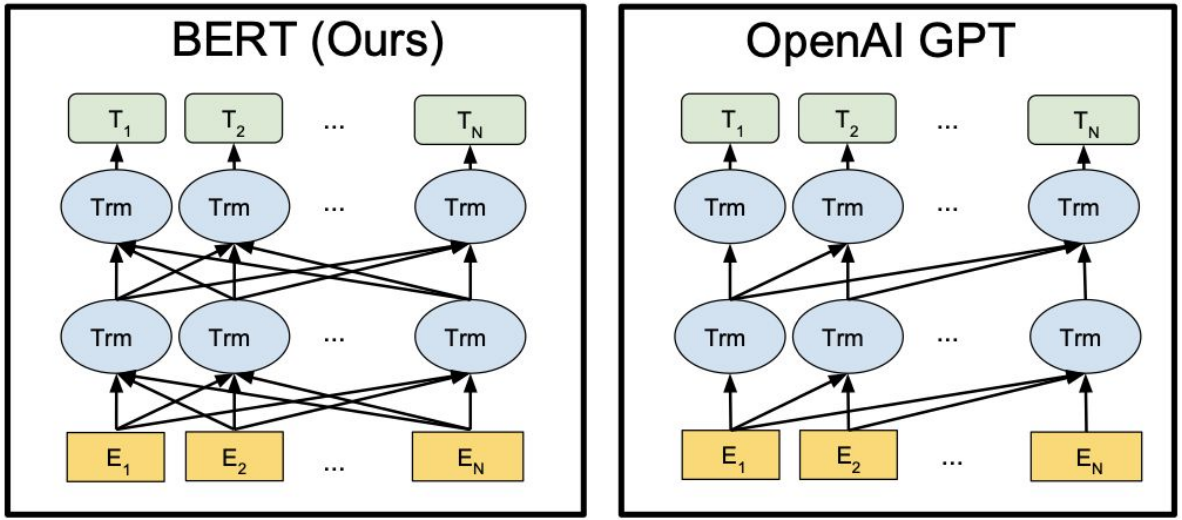
# Computational costs: pre-training

- Word2Vec: **a couple of hours** on a single GPU
- ULMfit: a **couple of days** on a single GPU
- ELMo: up to **6 weeks** on a single GPU
- GPT: **one month** on **8** GPUs

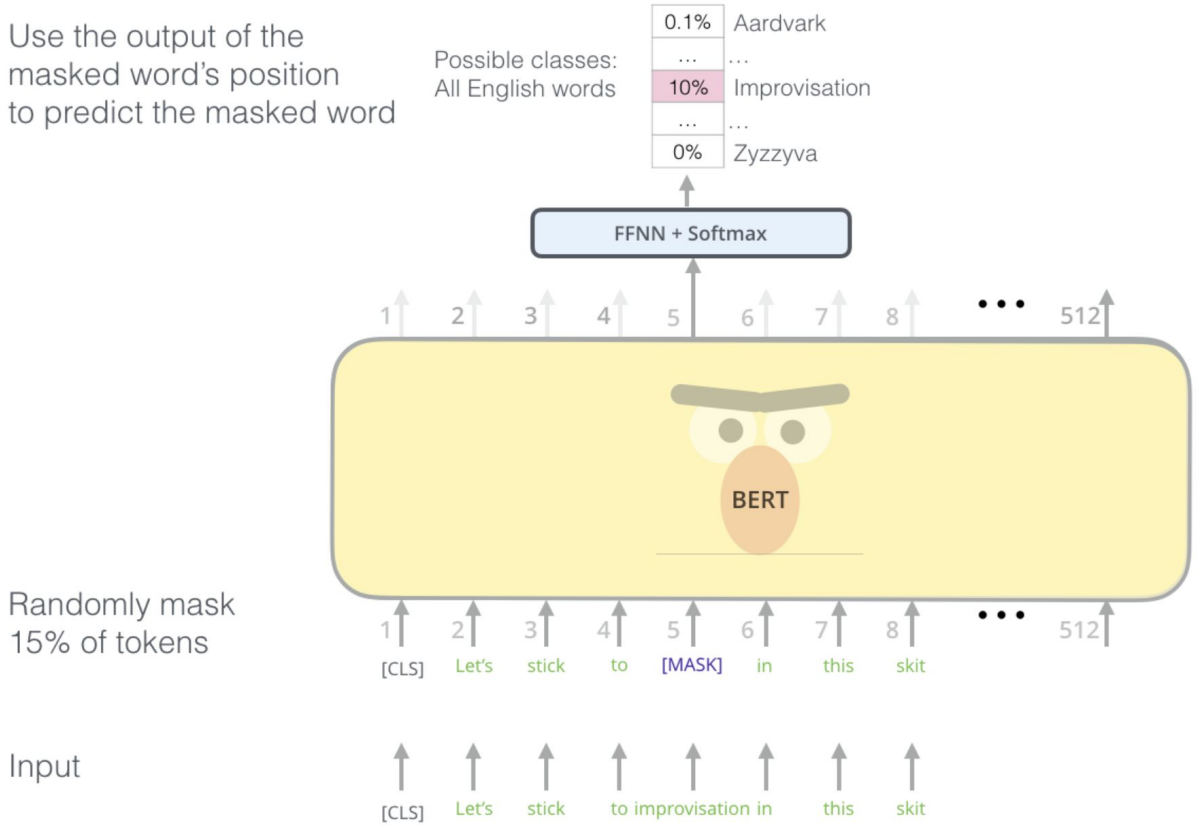
*Very old P100 GPUs from 2018

# BERT: BidirEctional TransfoRmers for Language Understanding

- Language models cannot see future tokens
- Can we figure out a pre-training task that does not restrict the model in such a way?



# Masked Language Modeling




BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

Image credit: Jay Alammar
jalammar.github.io/illustrated-bert

# BERT: Input Representation

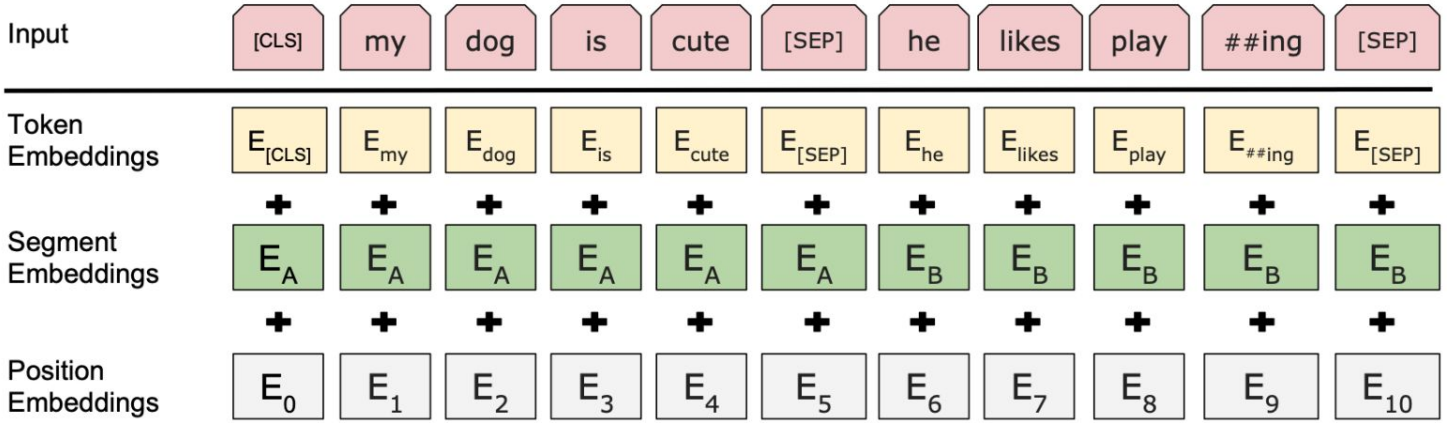



Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

# BERT: Results

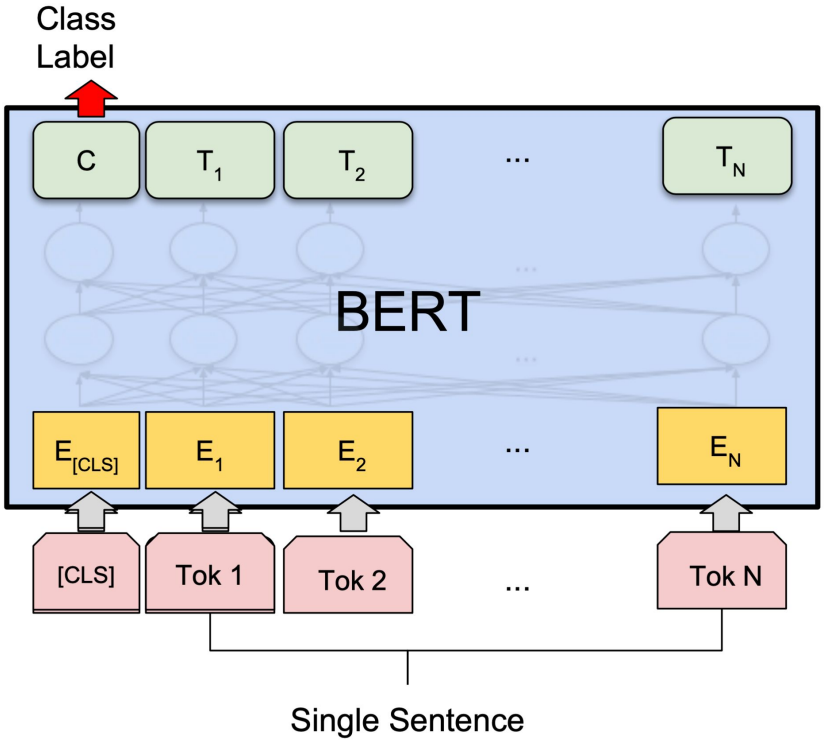
SOTA on **11** NLP tasks

| System | MNLI-(m/mm)<br>392k | QQP<br>363k | QNLI<br>108k | SST-2<br>67k | CoLA<br>8.5k | STS-B<br>5.7k | MRPC<br>3.5k | RTE<br>2.5k | **Average**<br>- |
|---|---|---|---|---|---|---|---|---|---|
| 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** |

# BERT: Input Formatting for classification

Examples:

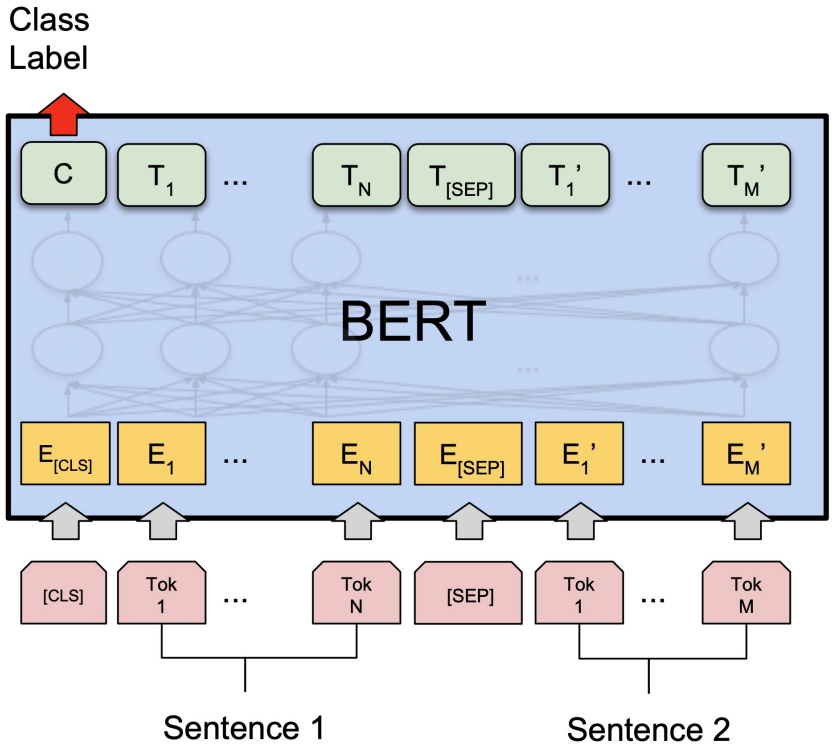
- Spam detection
  - Spam or ham
- Sentiment analysis
  - Positive or negative review
- Intent classification
  - Get weather
  - Set timer
  - Add calendar event
- Toxicity classification
  - Toxic or non-toxic



# BERT: Input Formatting for two-sentence classification

Examples:

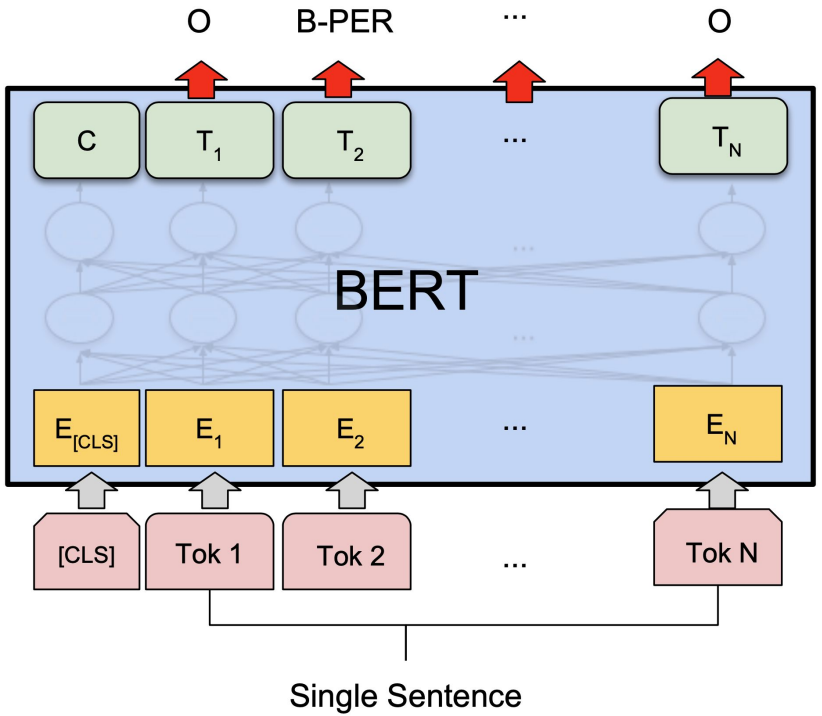
- Natural language inference
  - Does the second sentence logically follow from the first: “Penguins are birds”, “Penguins have wings”
- Sentence similarity
  - Do these two questions from Quora ask the same thing?



# BERT: Input Formatting for Sequence Tagging

Examples:

- Part of speech tagging
  - Noun
  - Verb
  - Preposition
- Named entity recognition
  - Name
  - Address
  - Not an entity



# BERT: Input Formatting for Question Answering

Special kind of question answering:

- Given the question and the context
- Detect if the question is answerable given the context
- If yes, find the span from the context that answers the question

“SQuAD question answering approach”

**Article:** Endangered Species Act
**Paragraph:** “ . . . *Other legislation followed, including the Migratory Bird Conservation Act of 1929, a 1937 treaty prohibiting the hunting of right and gray whales, and the Bald Eagle Protection Act of 1940. These later laws had a low cost to society—the species were relatively rare—and little opposition was raised.*”

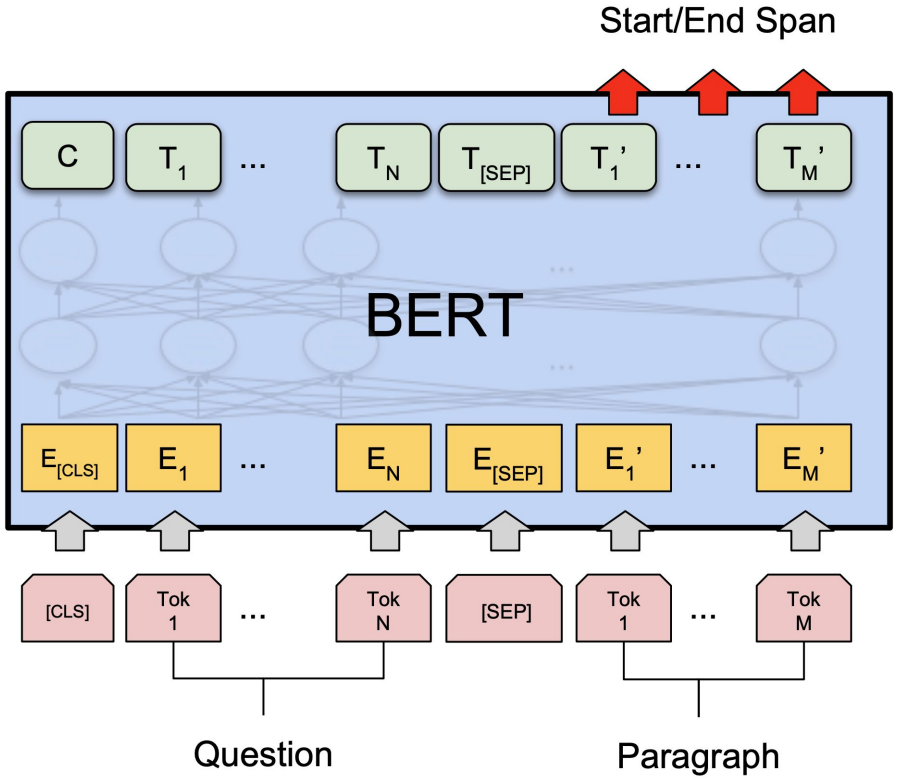
**Question 1:** “*Which laws faced significant opposition?*”
**Plausible Answer:** *later laws*

**Question 2:** “*What was the name of the 1937 treaty?*”
**Plausible Answer:** *Bald Eagle Protection Act*

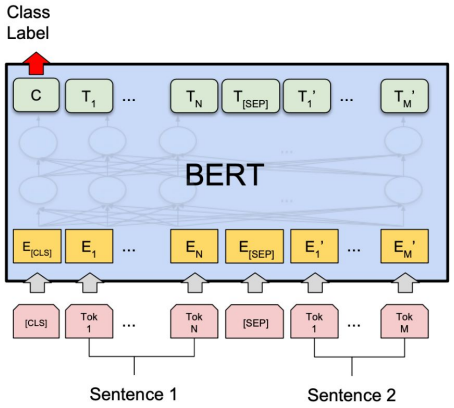
# BERT: Input Formatting for Question Answering

Special kind of question answering:

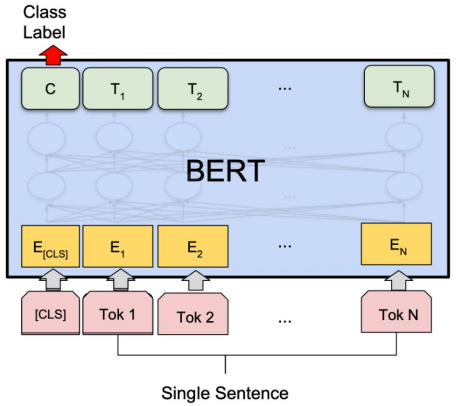
- Given the question and the context
- Detect if the question is answerable given the context
- If yes, find the span from the context that answers the question



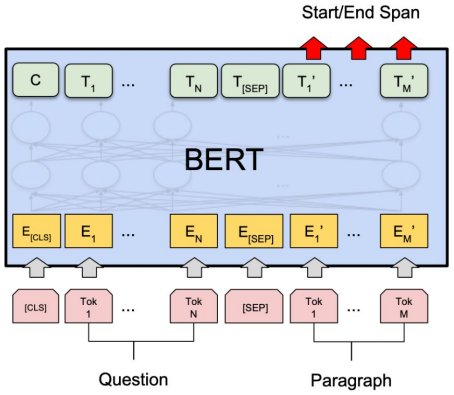
# BERT: Input Formatting



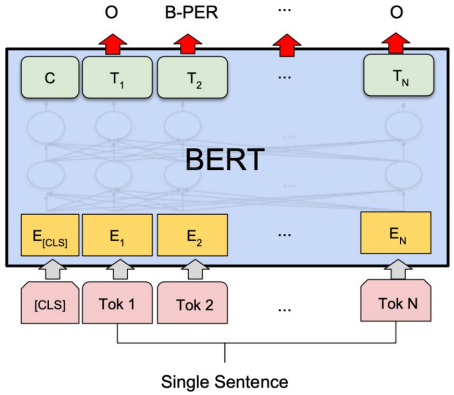

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG




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




(c) Question Answering Tasks: SQuAD v1.1




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

# GPT: General Purpose Transformer

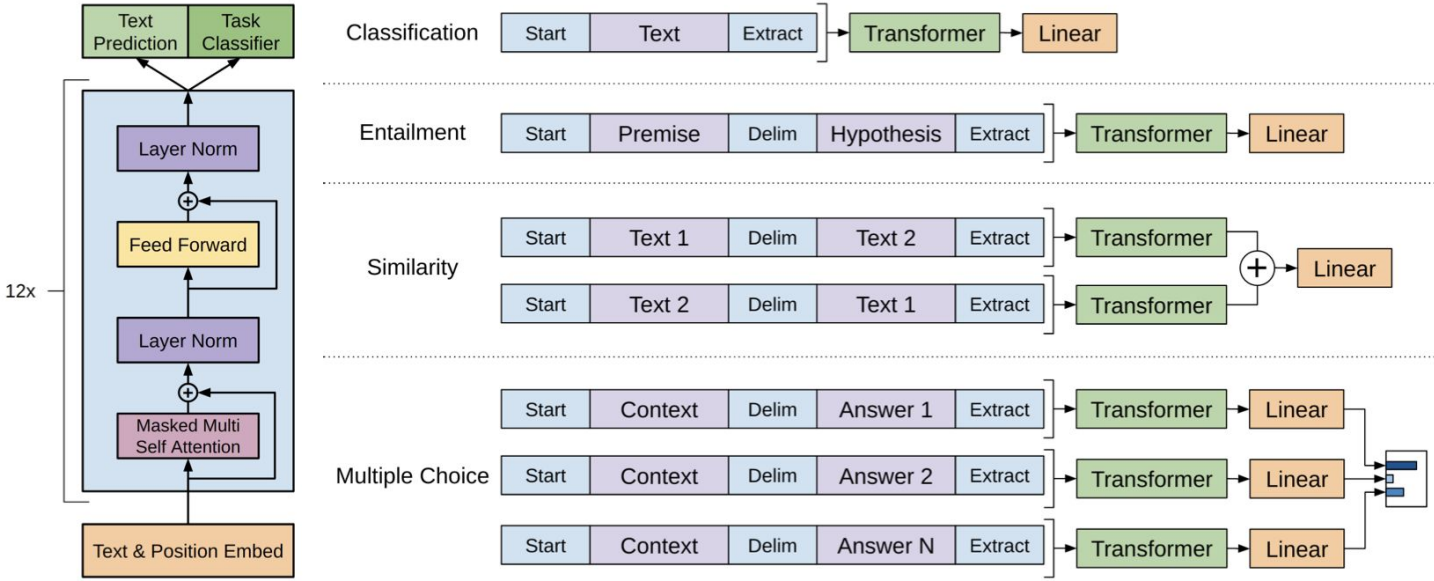
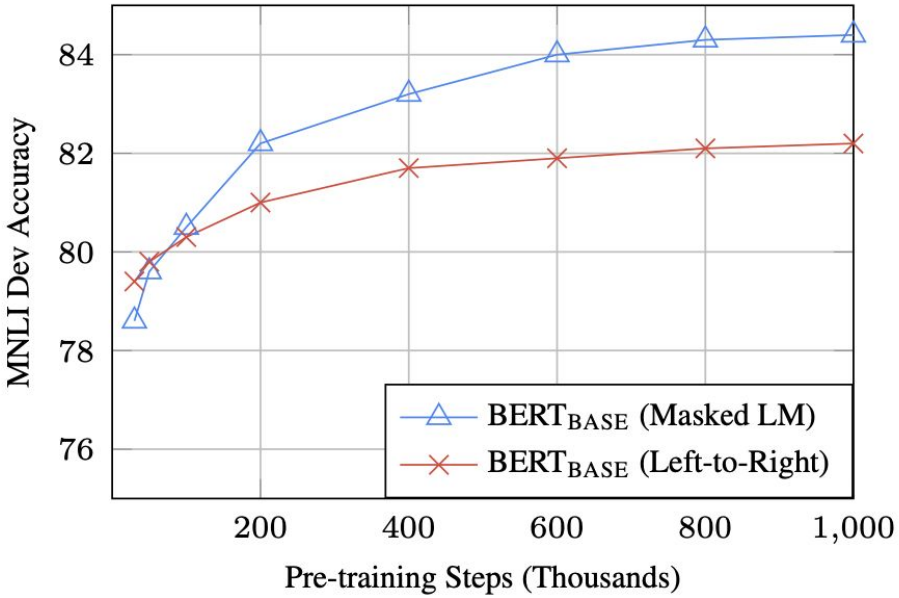



Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

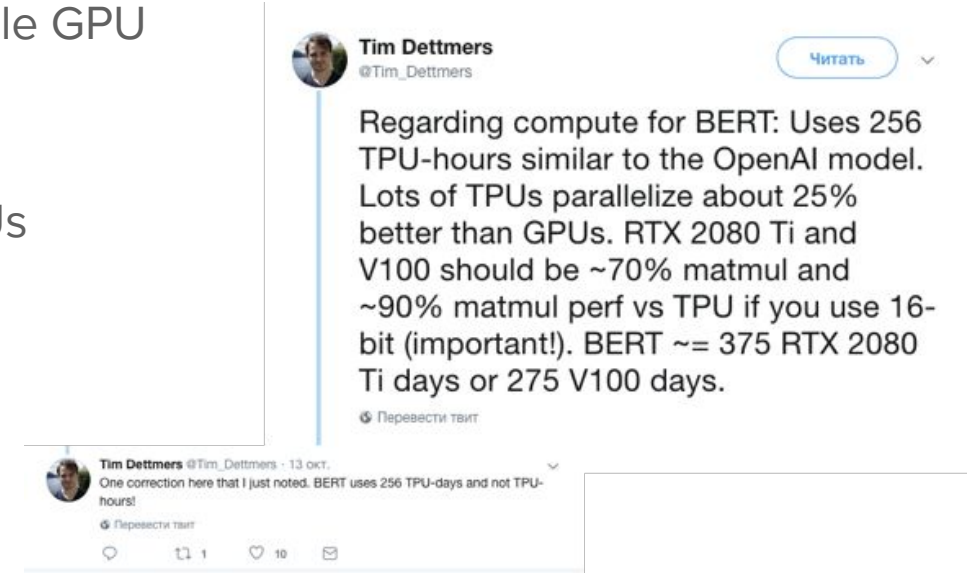
# Other differences from GPT

- Larger model
  - 24 layers instead of 12
  - Larger hidden sizes
- More pre-training data
  - BookCorpus + Wikipedia
- Extra pre-training objective
  - “Next Sentence Prediction” — classify if the pair of sentences follow each other or not
  - Turned out to be not important



# Computational costs: pre-training

- Word2Vec: **a couple of hours** on a single GPU
- ULMfit: a **couple of days** on a single GPU
- ELMo: up to **6 weeks** on a single GPU
- GPT: **one month** on **8** GPUs
- BERT: 4 days on 64 TPUs
  About **1 - 1.5 months** on 8 GPUs



# BERT and benchmarks

## CoQA

| | | | | |
|---|---|---|---|---|
| 1<br>Sep 05, 2019 | RoBERTa + AT + KD (ensemble)<br>*Zhuiyi Technology*<br>https://arxiv.org/abs/1909.10772 | **91.4** | **89.2** | **90.7** |

## SQuAD

| | | | |
|---|---|---|---|
| 1<br>Apr 06, 2020 | SA-Net on Albert (ensemble)<br>*QIANXIN* | **90.724** | **93.011** |

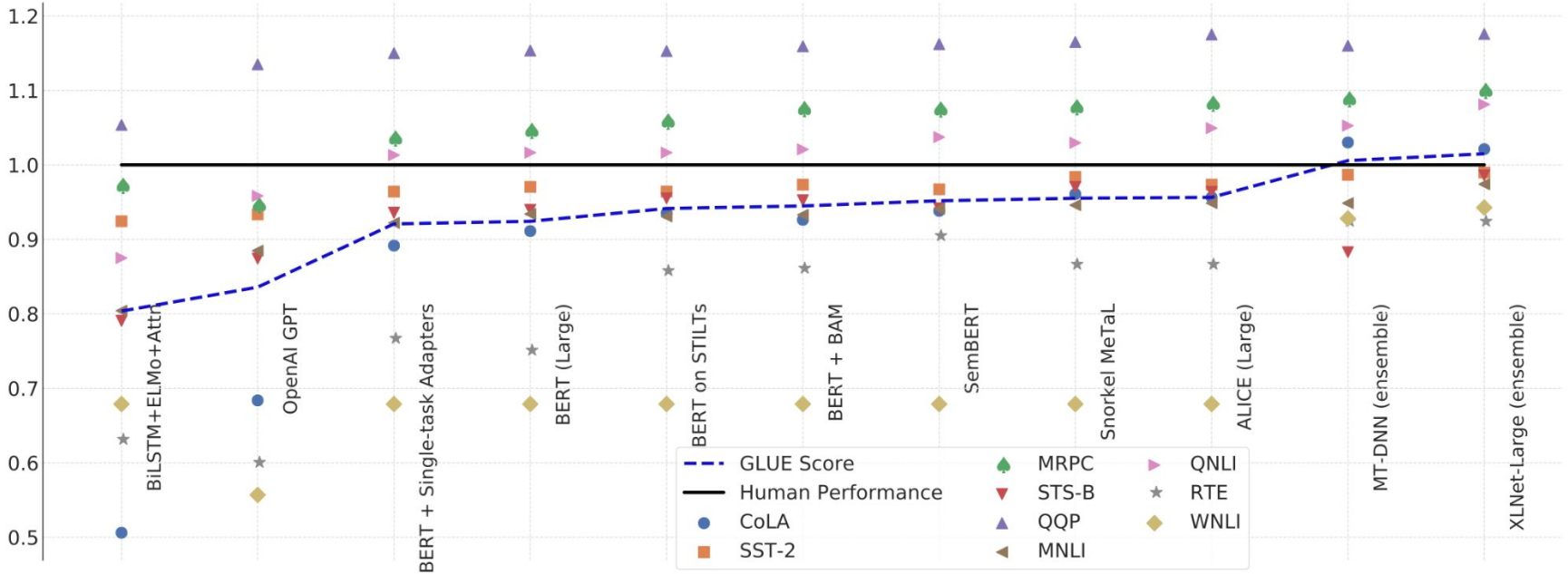
## Yelp reviews

| | | | |
|---|---|---|---|
| 1 | **BERT large** | 29.32 | Unsupervised Data Augmentation |

## Ontonotes-5

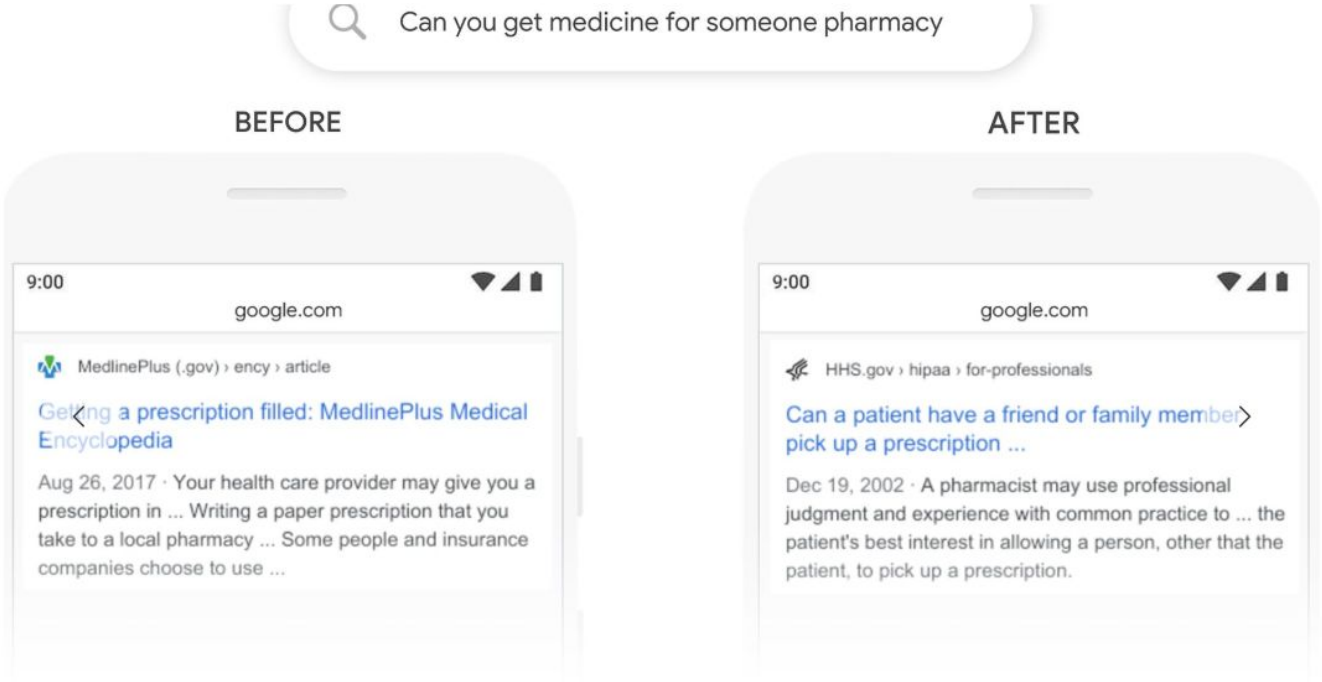
| | | | | |
|---|---|---|---|---|
| 1 | **BERT-MRC+DSC** | 92.07 | ✓ | Dice Loss for Data-imbalanced NLP Tasks |

# BERT and benchmarks: superhuman performance




https://gluebenchmark.com/leaderboard

# BERT improves ~10% of Google search queries




https://techcrunch.com/2019/10/25/google-brings-in-bert-to-improve-its-search-results/

Slide credit: Olga Kovaleva

# Better benchmarks

## SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems


**Alex Wang*** New York University

**Yada Pruksachatkun*** New York University

**Nikita Nangia*** New York University

**Amanpreet Singh*** Facebook AI Research

**Julian Michael** University of Washington

**Felix Hill** DeepMind

**Omer Levy** Facebook AI Research

**Samuel R. Bowman** New York University

# Better benchmarks

| Rank | Name | Model | URL | Score | BoolQ | CB | COPA | MultiRC | ReCoRD | RTE | WiC | WSC | AX-b | AX-g |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | JDExplore d-team | Vega v2 | | 91.3 | 90.5 | 98.6/99.2 | 99.4 | 88.2/62.4 | 94.4/93.9 | 96.0 | 77.4 | 98.6 | -0.4 | 100.0/50.0 |
| 2 | Liam Fedus | ST-MoE-32B | | 91.2 | 92.4 | 96.9/98.0 | 99.2 | 89.6/65.8 | 95.1/94.4 | 93.5 | 77.7 | 96.6 | 72.3 | 96.1/94.1 |
| 3 | Microsoft Alexander v-team | Turing NLR v5 | | 90.9 | 92.0 | 95.9/97.6 | 98.2 | 88.4/63.0 | 96.4/95.9 | 94.1 | 77.1 | 97.3 | 67.8 | 93.3/95.5 |
| 4 | ERNIE Team - Baidu | ERNIE 3.0 | | 90.6 | 91.0 | 98.6/99.2 | 97.4 | 88.6/63.2 | 94.7/94.2 | 92.6 | 77.4 | 97.3 | 68.6 | 92.7/94.7 |
| 5 | Yi Tay | PaLM 540B | | 90.4 | 91.9 | 94.4/96.0 | 99.0 | 88.7/63.6 | 94.2/93.3 | 94.1 | 77.4 | 95.9 | 72.9 | 95.5/90.4 |
| 6 | Zirui Wang | T5 + UDG, Single Model (Google Brain) | | 90.4 | 91.4 | 95.8/97.6 | 98.0 | 88.3/63.0 | 94.2/93.5 | 93.0 | 77.9 | 96.6 | 69.1 | 92.7/91.9 |
| 7 | DeBERTa Team - Microsoft | DeBERTa / TuringNLRv4 | | 90.3 | 90.4 | 95.7/97.6 | 98.4 | 88.2/63.7 | 94.5/94.1 | 93.2 | 77.5 | 95.9 | 66.7 | 93.3/93.8 |
| 8 | SuperGLUE Human Baselines | SuperGLUE Human Baselines | | 89.8 | 89.0 | 95.8/98.9 | 100.0 | 81.8/51.9 | 91.7/91.3 | 93.6 | 80.0 | 100.0 | 76.6 | 99.3/99.7 |

# Modern NLP tooling: 🤗

DEMO?

# Conclusion

- Pre-training is cool, MLM is a standard pre-training method
- Pre-trained transformers are de-facto standard in modern NLP
- Numerous pre-training objectives have been explored, we'll talk more about some of them in future lectures
- Tooling is important