# Pretraining Language Models

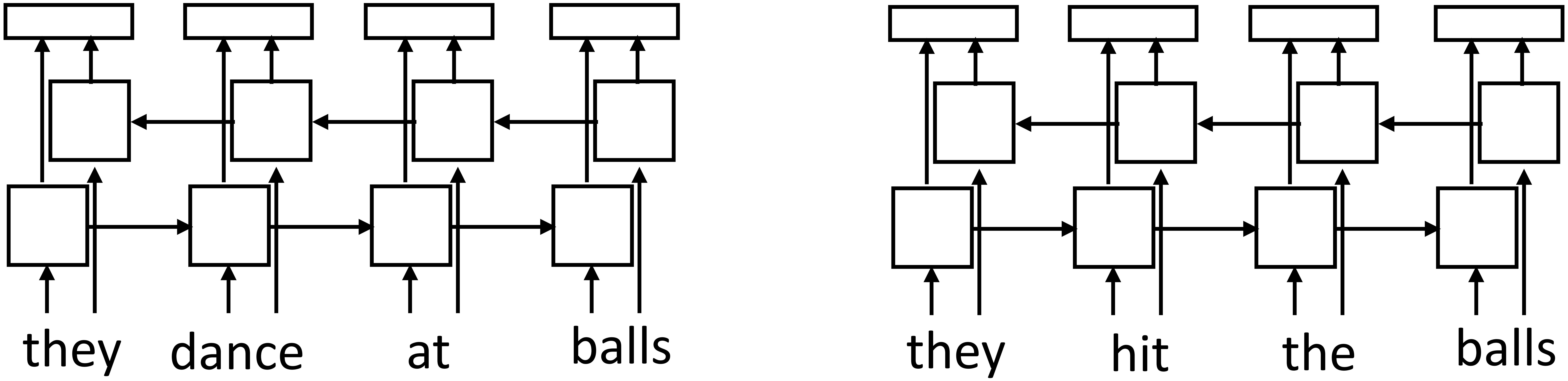
Wei Xu

(many slides from Greg Durrett)

# Pretraining / ELMo

# Recall: Context-dependent Embeddings

- How to handle different word senses? One vector for *balls*

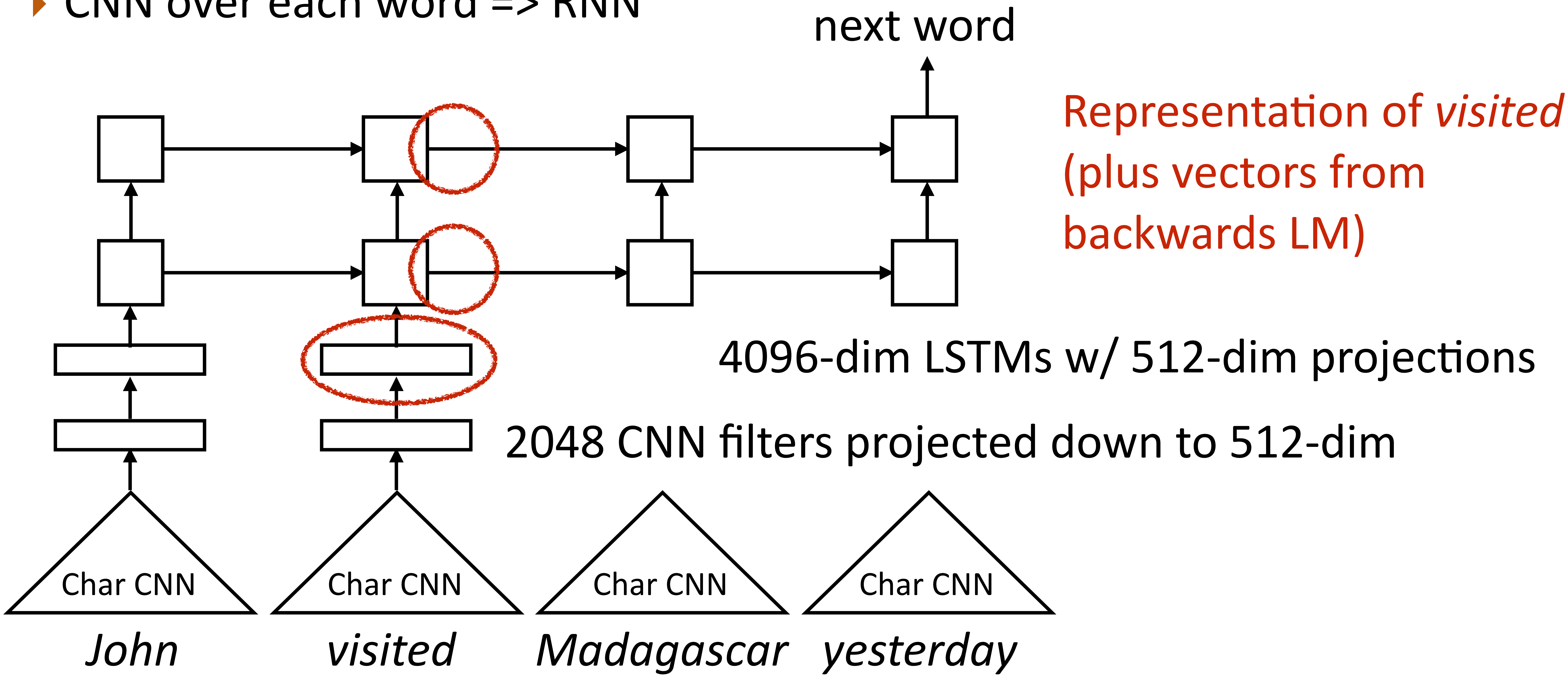



- Train a neural language model to predict the next word given previous words in the sentence, use its internal representations as word vectors

Peters et al. (2018)

# ELMo

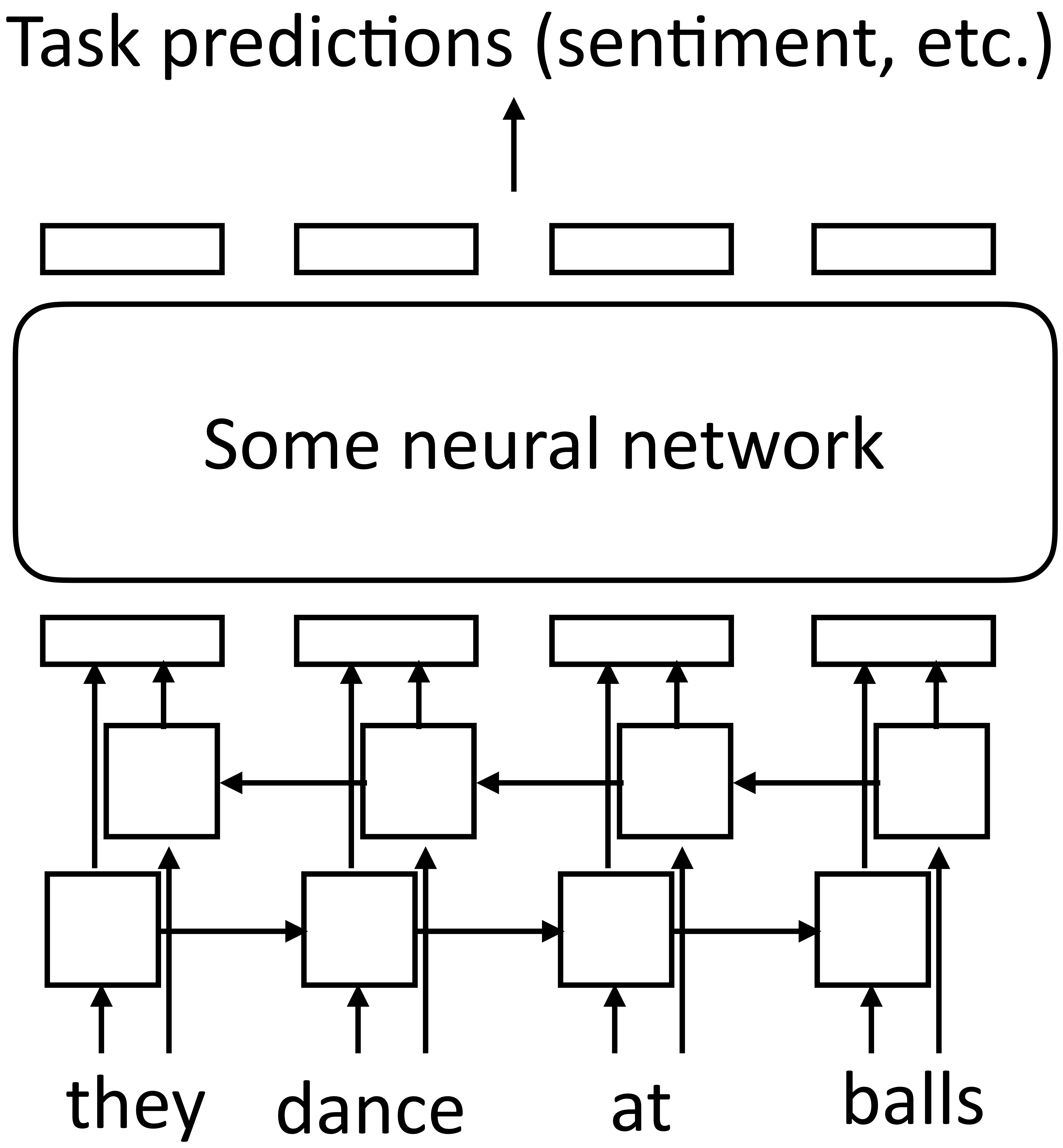
- CNN over each word => RNN




Peters et al. (2018)

# How to apply ELMo?

- Take those embeddings and feed them into whatever architecture you want to use for your task
- *Frozen* embeddings: update the weights of your network but keep ELMo's parameters frozen
- *Fine-tuning*: backpropagate all the way into ELMo when training your model




Peters, Ruder, Smith (2019)

# Results: Frozen ELMo

| 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 | $88.7 \pm 0.17$ | 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) | $91.93 \pm 0.19$ | 90.15 | $92.22 \pm 0.10$ | 2.06 / 21% |
| SST-5 | McCann et al. (2017) | 53.7 | 51.4 | $54.7 \pm 0.5$ | 3.3 / 6.8% |

- Massive improvements across 5 benchmark datasets: question answering, natural language inference, semantic role labeling (discussed later in the course), coreference resolution, named entity recognition, and sentiment analysis

# How to apply ELMo?

| Pretraining | Adaptation | NER CoNLL 2003 | SA SST-2 | Nat. lang. inference MNLI | SICK-E | Semantic textual similarity SICK-R | MRPC | STS-B |
|---|---|---|---|---|---|---|---|---|
| Skip-thoughts | ❄️ | - | 81.8 | 62.9 | - | 86.6 | 75.8 | 71.8 |
| ELMo | ❄️ | 91.7 | **91.8** | **79.6** | **86.3** | **86.1** | **76.0** | **75.9** |
| | 🔥 | **91.9** | 91.2 | 76.4 | 83.3 | 83.3 | 74.7 | 75.5 |
| | Δ=🔥-❄️ | 0.2 | -0.6 | -3.2 | -3.3 | -2.8 | -1.3 | -0.4 |

- How does frozen ( ❄️ ) vs. fine-tuned ( 🔥 ) compare?
- Recommendations:

| Conditions: Pretrain | Adapt. | Task | Guidelines |
|---|---|---|---|
| Any | ❄️ | Any | Add many task parameters |
| Any | 🔥 | Any | Add minimal task parameters<br>⚠️ Hyper-parameters |
| Any | Any | Seq. / clas. | ❄️ and 🔥 have similar performance |
| ELMo | Any | Sent. pair | use ❄️ |
| BERT | Any | Sent. pair | use 🔥 |

Peters, Ruder, Smith (2019)

# Why is language modeling a good objective?

- "Impossible" problem but bigger models seem to do better and better at distributional modeling (no upper limit yet)
- Successfully predicting next words requires modeling lots of different effects in text

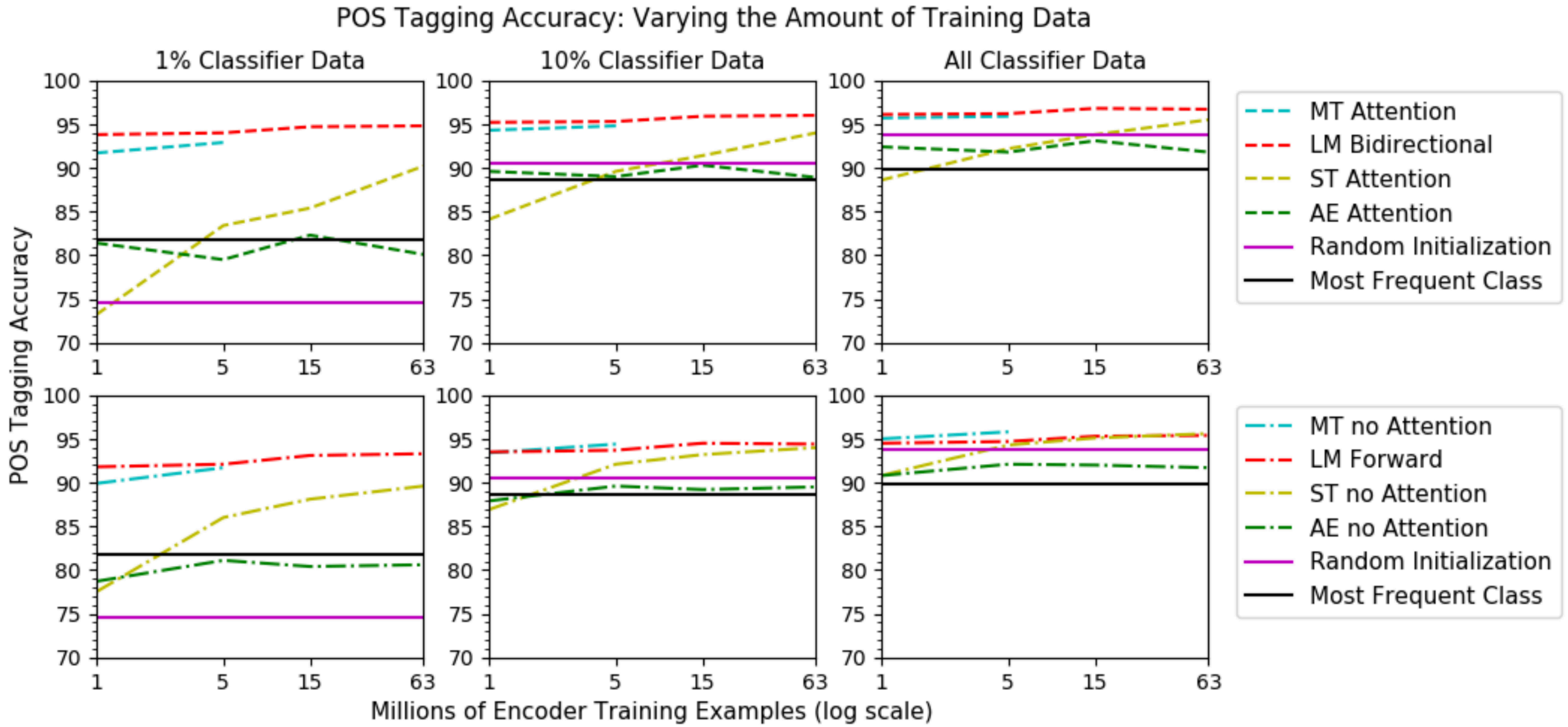
*Context:* My wife refused to allow me to come to Hong Kong when the plague was at its height and –" "Your wife, Johanne? You are married at last ?" Johanne grinned. "Well, when a man gets to my age, he starts to need a few home comforts.

*Target sentence:* After my dear mother passed away ten years ago now, I became _____.

*Target word:* lonely

- LAMBADA dataset (Papernot et al., 2016): explicitly targets world knowledge and very challenging LM examples
- Coreference, Winograd schema, and much more

# Why is language modeling a good objective?




Zhang and Bowman (2018)

# Why did this take time to catch on?

- Earlier version of ELMo by the same authors in 2017, but it was only evaluated on tagging tasks, gains were 1% or less

- Required: training on lots of data, having the right architecture, significant hyperparameter tuning

# Probing ELMo

- From each layer of the ELMo model, attempt to predict something: POS tags, word senses, etc.
- Higher accuracy => ELMo is capturing that thing more nicely

| Model | $F_1$ |
|---|---|
| WordNet 1st Sense Baseline | 65.9 |
| Raganato et al. (2017a) | 69.9 |
| Iacobacci et al. (2016) | **70.1** |
| CoVe, First Layer | 59.4 |
| CoVe, Second Layer | *64.7* |
| biLM, First layer | 67.4 |
| biLM, Second layer | *69.0* |

Table 5: All-words fine grained WSD $F_1$. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

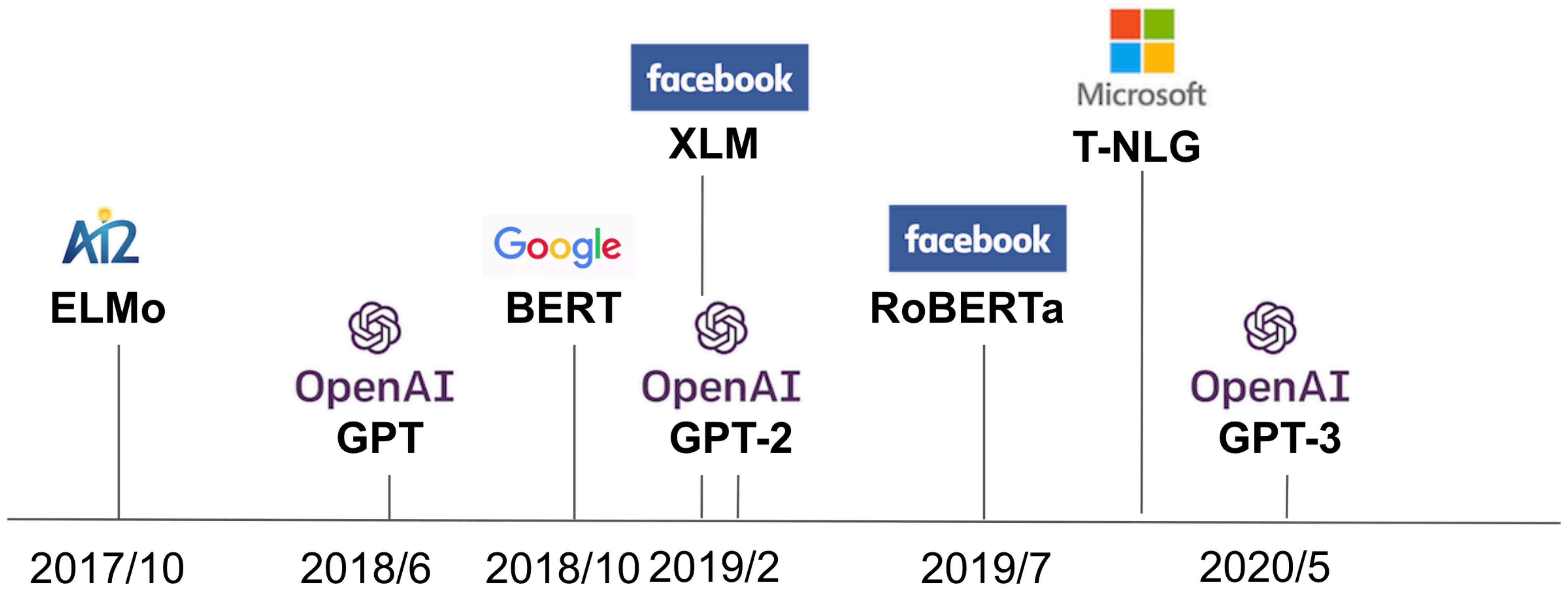
| Model | Acc. |
|---|---|
| Collobert et al. (2011) | 97.3 |
| Ma and Hovy (2016) | 97.6 |
| Ling et al. (2015) | **97.8** |
| CoVe, First Layer | *93.3* |
| CoVe, Second Layer | 92.8 |
| biLM, First Layer | *97.3* |
| biLM, Second Layer | 96.8 |

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

# BERT

# BERT

- AI2 made ELMo in spring 2018, GPT was released in summer 2018, BERT came out October 2018
- Three major changes compared to ELMo:
  - Transformers instead of LSTMs (transformers in GPT as well)
  - Bidirectional <=> Masked LM objective instead of standard LM
  - Fine-tune instead of freeze at test time



# BERT

## 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



### Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

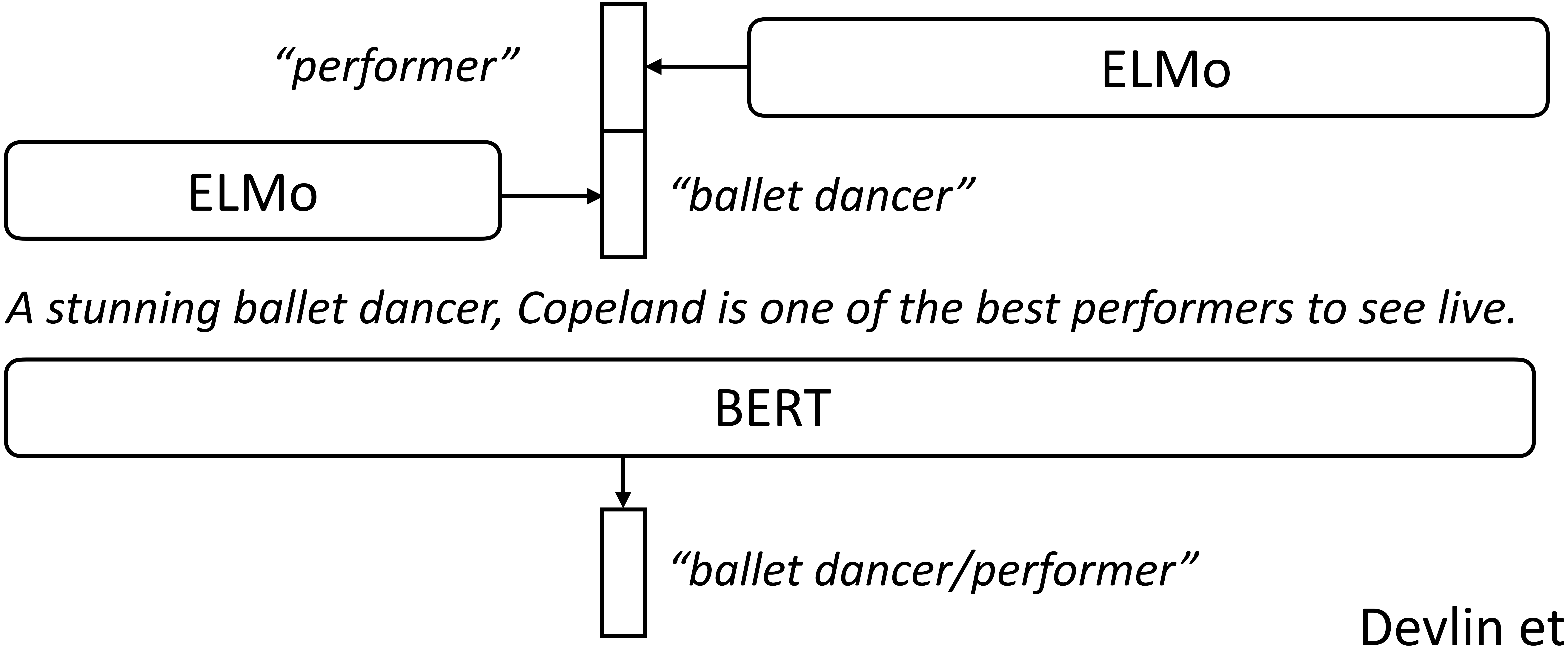
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers

# BERT

- ELMo is a unidirectional model (as is GPT): we can concatenate two unidirectional models, but is this the right thing to do?
- ELMo reprs look at each direction in isolation; BERT looks at them jointly

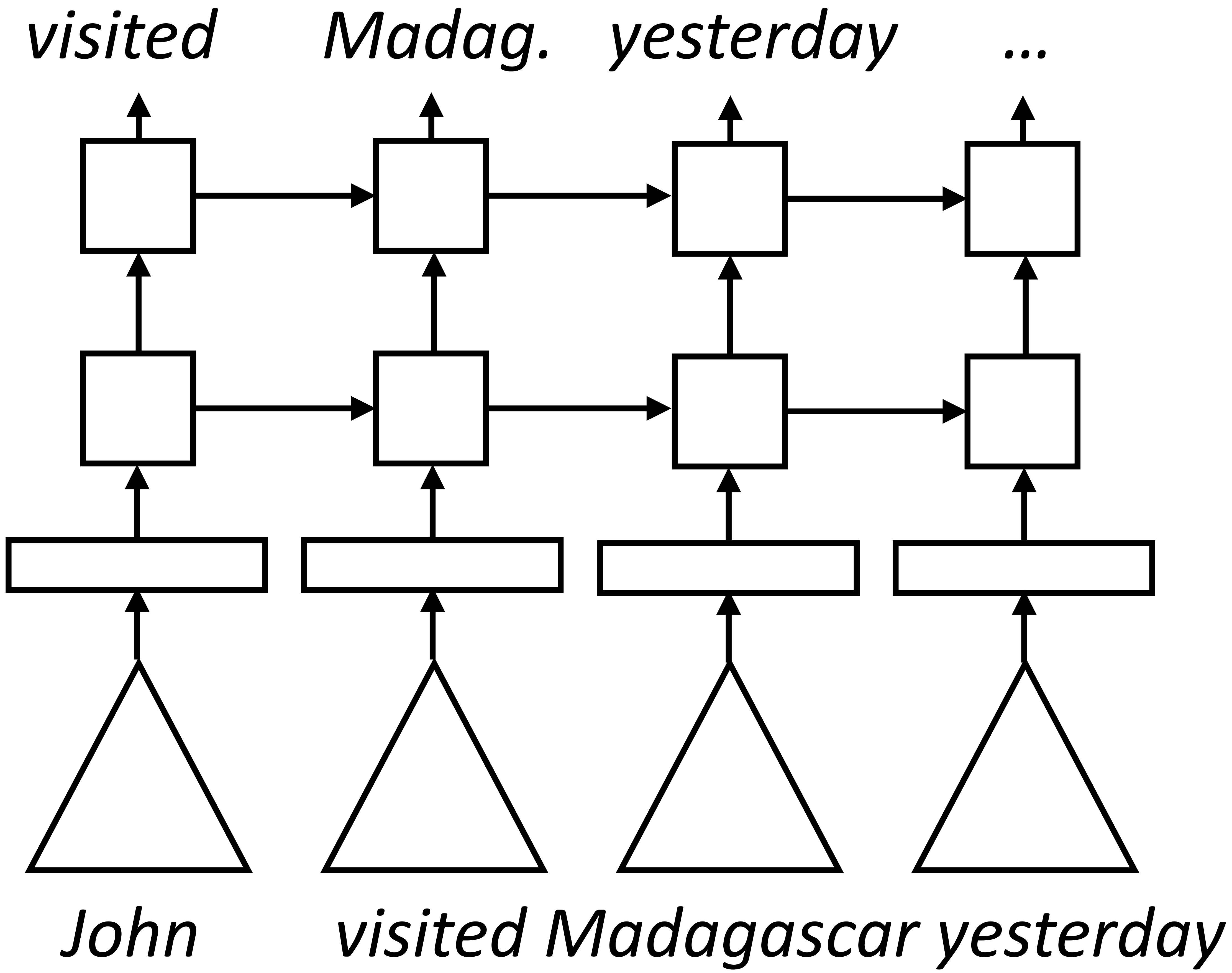



Devlin et al. (2019)

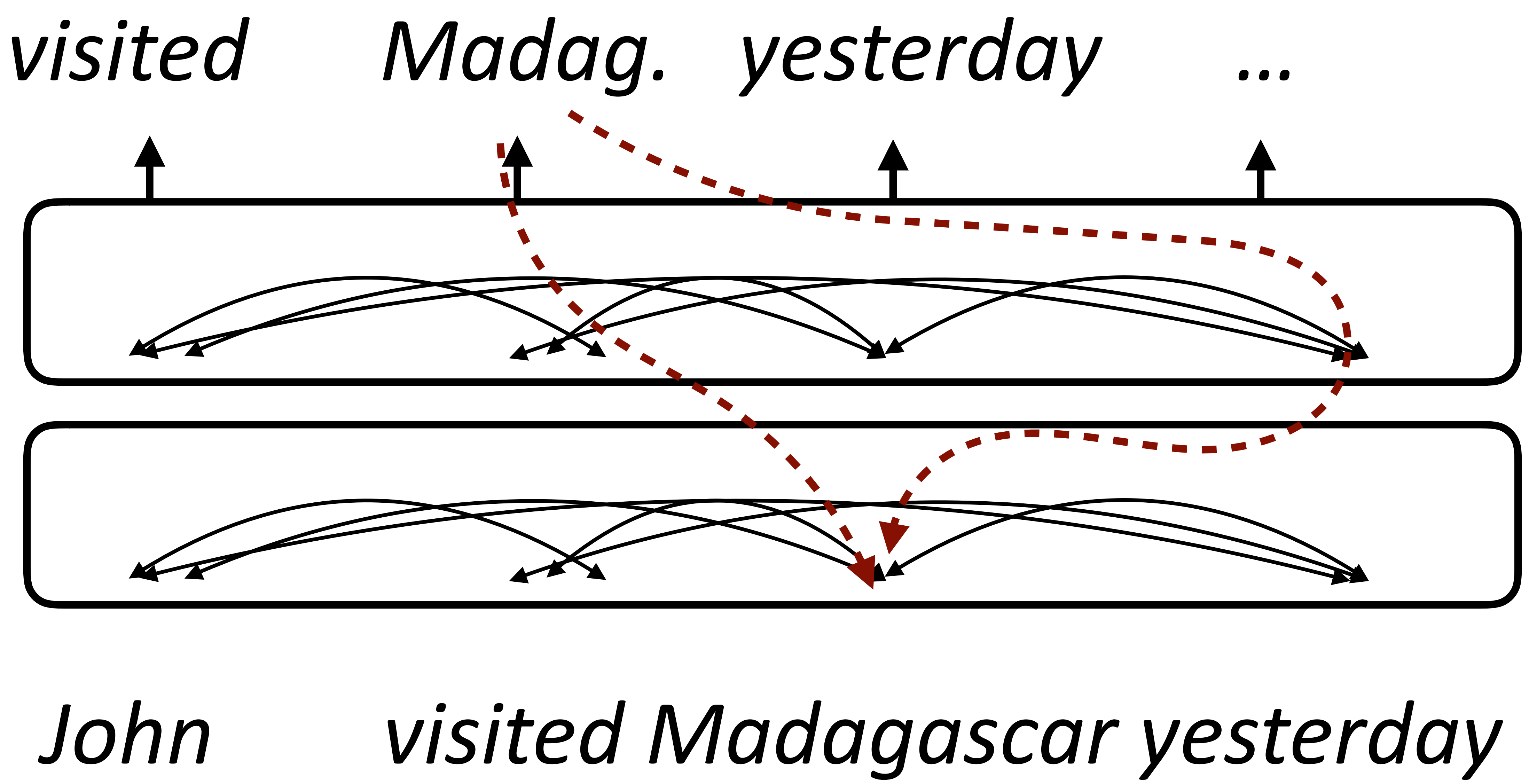
# BERT

- How to learn a "deeply bidirectional" model? What happens if we just replace an LSTM with a transformer?

ELMo (Language Modeling)




BERT

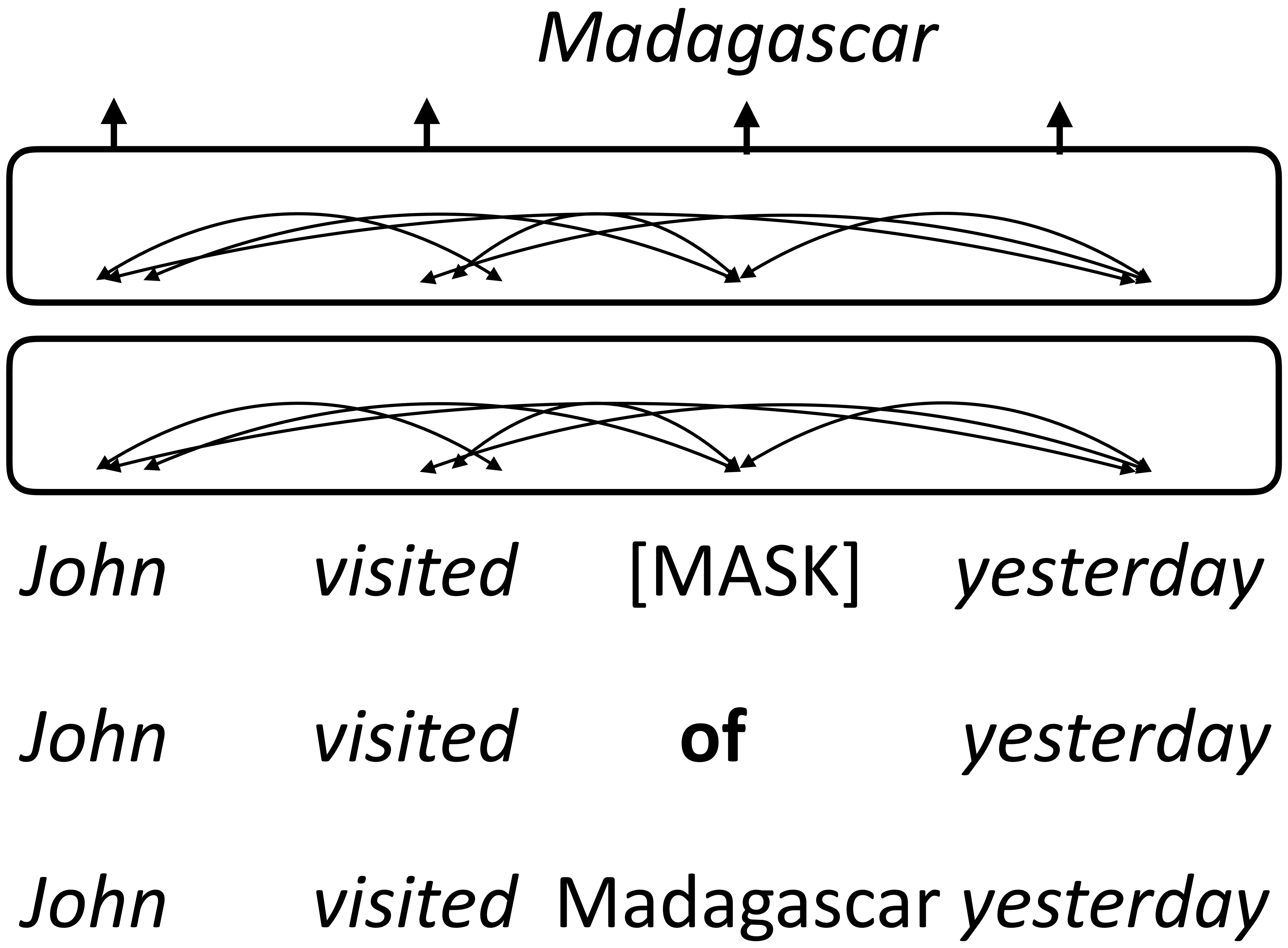



- Transformer LMs have to be "one-sided" (only attend to previous tokens), not what we want

# Masked Language Modeling

- How to prevent cheating? Next word prediction fundamentally doesn't work for bidirectional models, instead do *masked language modeling*

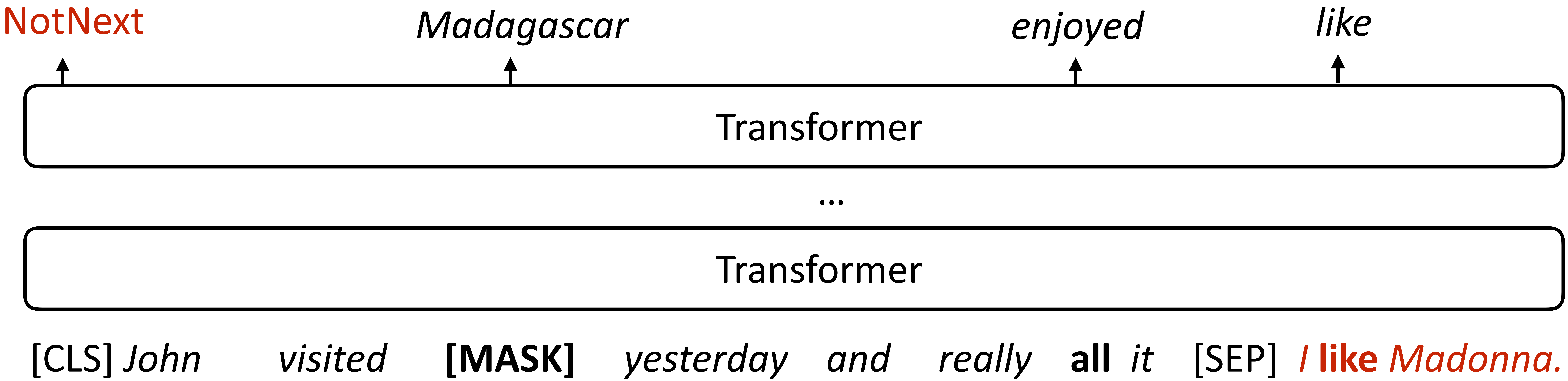
- BERT formula: take a chunk of text, predict 15% of the tokens
  - For 80% (of the 15%), replace the input token with [MASK]
  - For 10%, replace w/random
  - For 10%, keep same




Devlin et al. (2019)

# Next "Sentence" Prediction

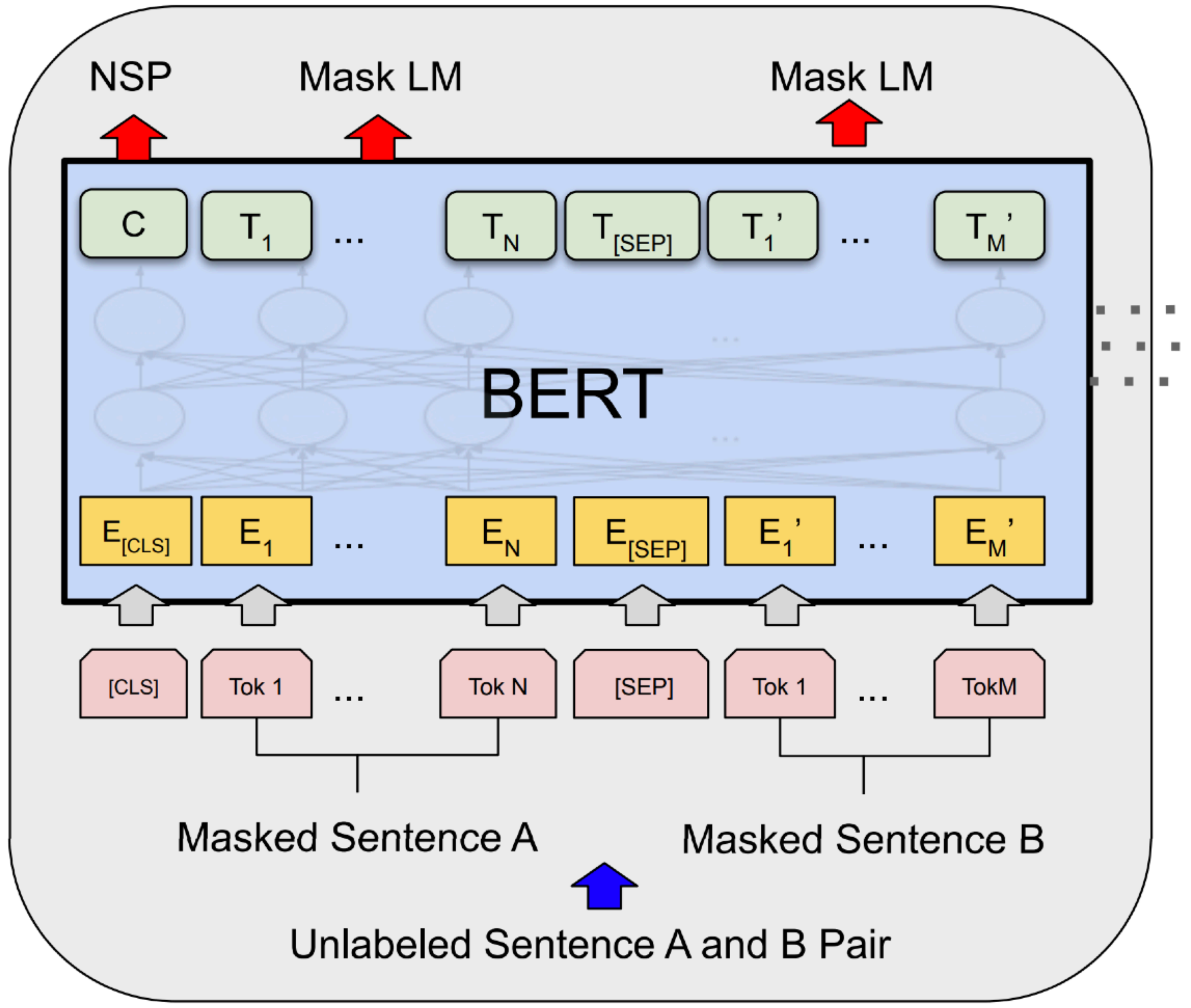
- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the "true" next
- BERT objective: masked LM + next sentence prediction

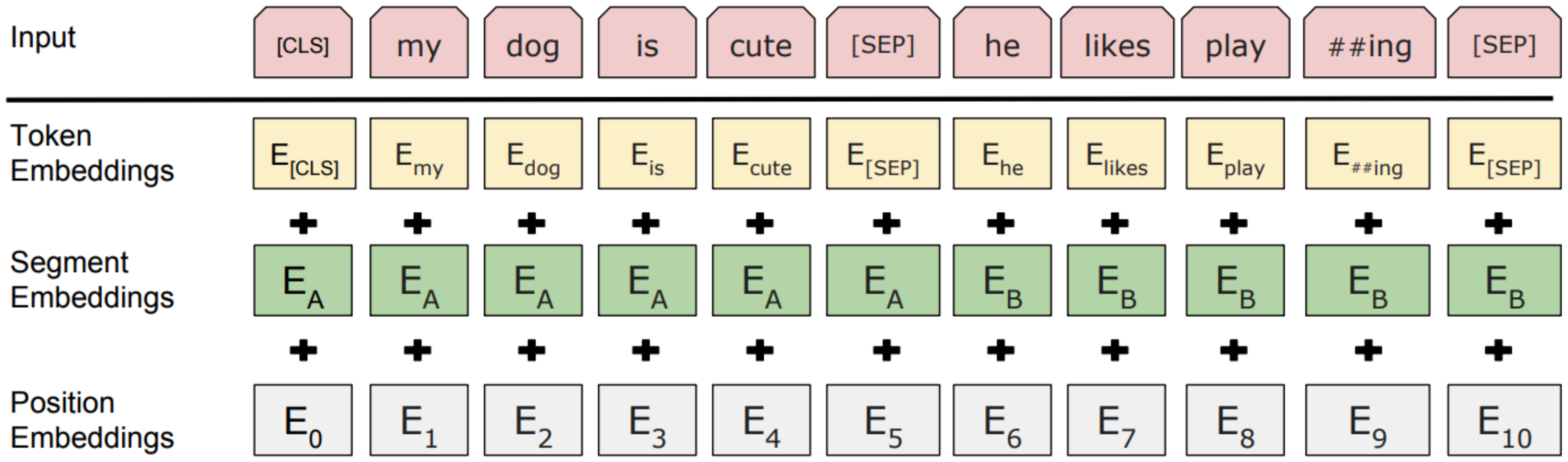



Devlin et al. (2019)

# BERT Architecture

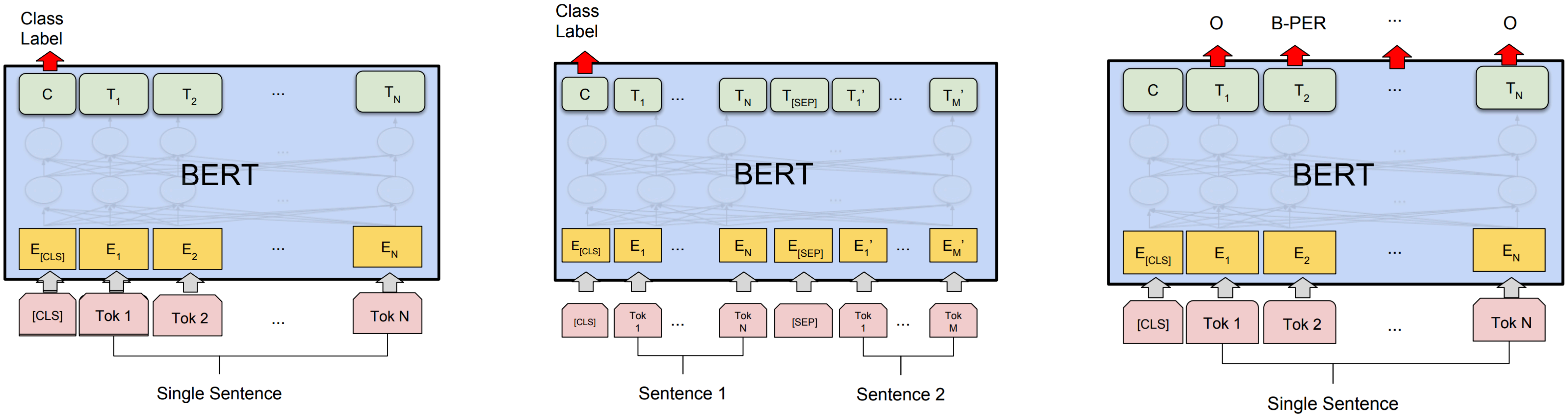
- BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M
- BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M
- Positional embeddings and segment embeddings, 30k word pieces
- This is the model that gets **pre-trained** on a large corpus




Devlin et al. (2019)

# What can BERT do?




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

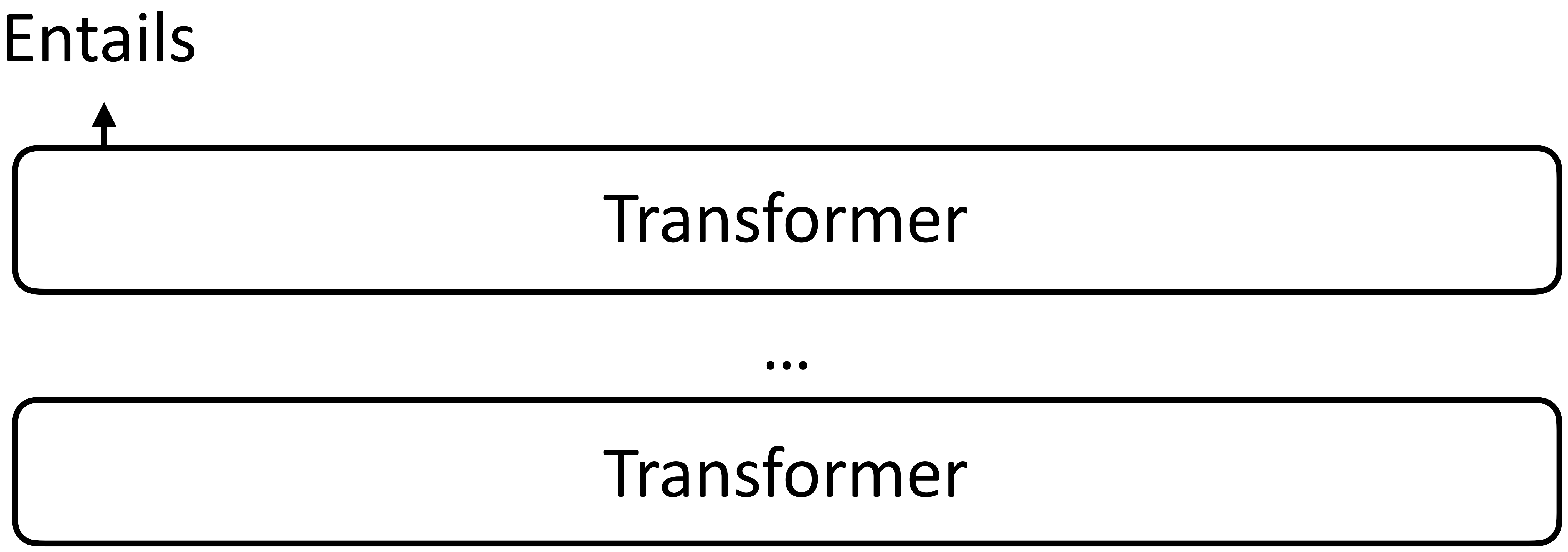
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

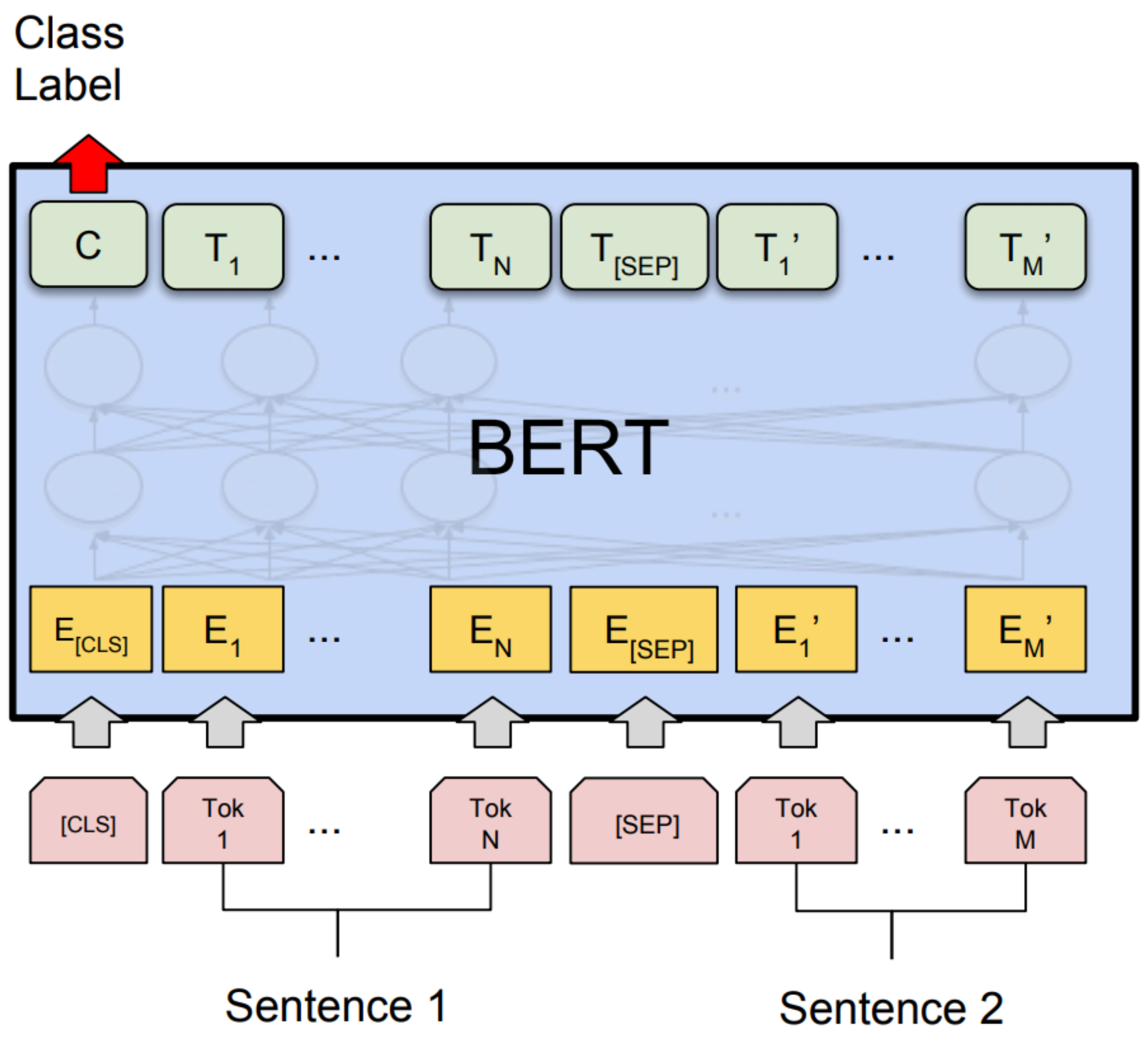
- CLS token is used to provide classification decisions
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

# What can BERT do?




[CLS] A boy plays in the snow [SEP] A boy is outside



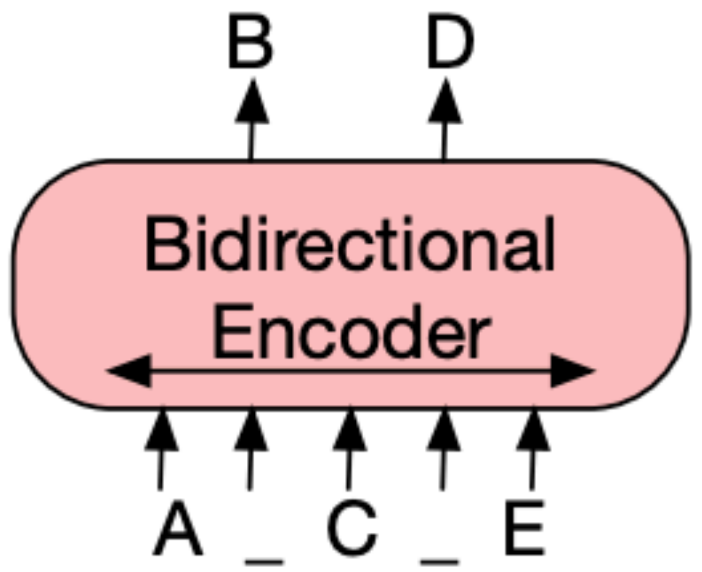

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

- How does BERT model this sentence pair stuff?
- Transformers can capture interactions between the two sentences, even though the NSP objective doesn't really cause this to happen

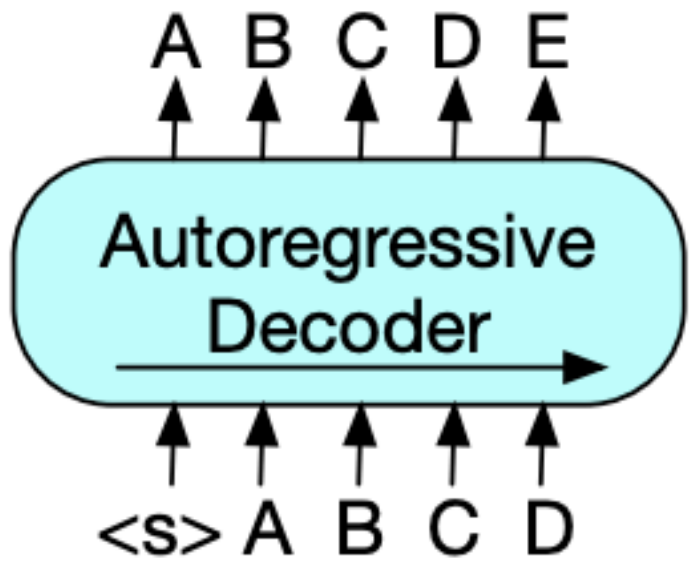
# What can BERT NOT do?

- BERT **cannot** generate text (at least not in an obvious way)
  - Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat

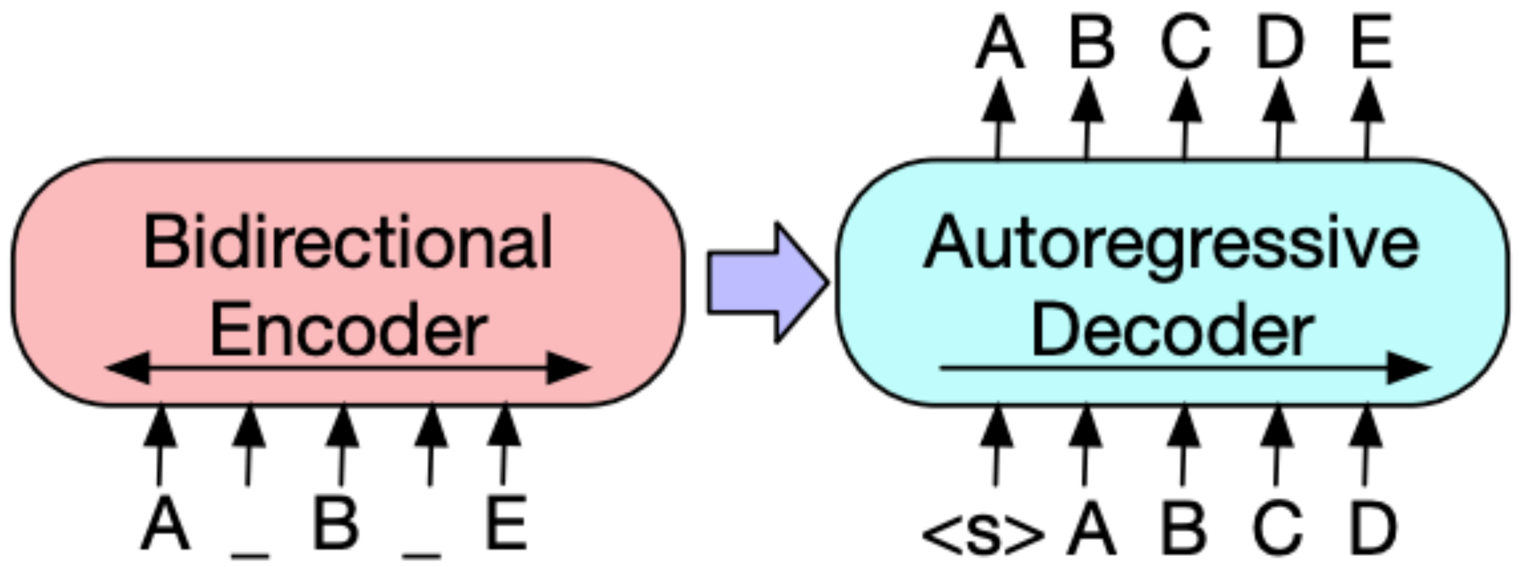
# What can BERT NOT do?




(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.




(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



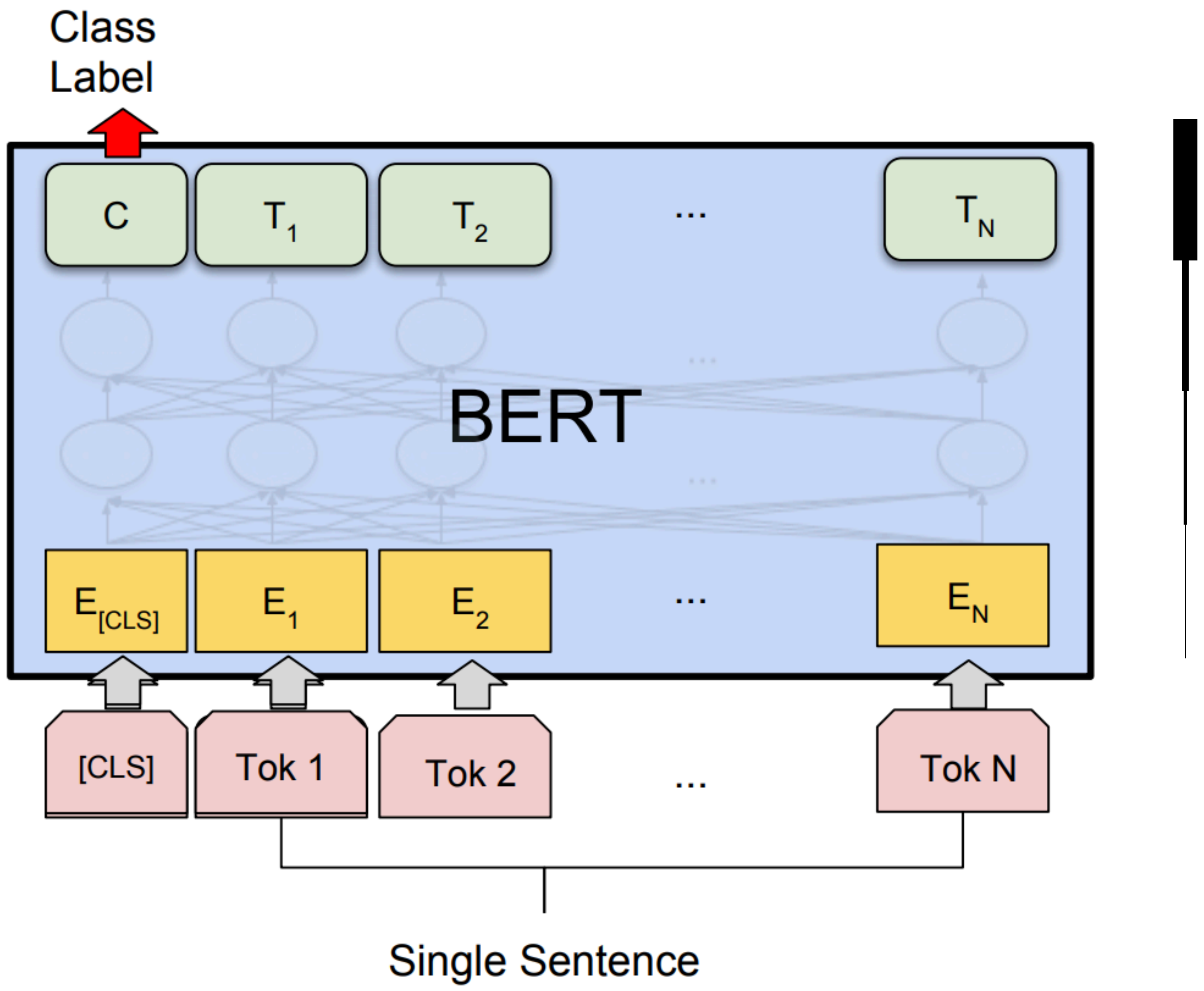

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

Lewis et al. (2019)

# Fine-tuning BERT

- Fine-tune for 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5




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

- Large changes to weights up here (particularly in last layer to route the right information to [CLS])
- Smaller changes to weights lower down in the transformer
- Small LR and short fine-tuning schedule mean weights don’t change much
- More complex “triangular learning rate” schemes exist

# Fine-tuning BERT

| Pretraining | Adaptation | NER<br>CoNLL 2003 | SA<br>SST-2 | Nat. lang. inference<br>MNLI | <br>SICK-E | Semantic textual similarity<br>SICK-R | <br>MRPC | <br>STS-B |
|---|---|---|---|---|---|---|---|---|
| Skip-thoughts | ❄️ | - | 81.8 | 62.9 | - | 86.6 | 75.8 | 71.8 |
| ELMo | ❄️ | 91.7 | **91.8** | **79.6** | **86.3** | **86.1** | **76.0** | **75.9** |
| | 🔥 | **91.9** | 91.2 | 76.4 | 83.3 | 83.3 | 74.7 | 75.5 |
| | Δ=🔥-❄️ | 0.2 | -0.6 | -3.2 | -3.3 | -2.8 | -1.3 | -0.4 |
| BERT-base | ❄️ | 92.2 | 93.0 | **84.6** | 84.8 | 86.4 | 78.1 | 82.9 |
| | 🔥 | **92.4** | **93.5** | **84.6** | **85.8** | **88.7** | **84.8** | **87.1** |
| | Δ=🔥-❄️ | 0.2 | 0.5 | 0.0 | 1.0 | 2.3 | 6.7 | 4.2 |

- BERT is typically better if the whole network is fine-tuned, unlike ELMo

Peters, Ruder, Smith (2019)

# Evaluation: GLUE

| Corpus | \|Train\| | \|Test\| | Task | Metrics | Domain |
|---|---|---|---|---|---|
| | | | Single-Sentence Tasks | | |
| CoLA | 8.5k | **1k** | acceptability | Matthews corr. | misc. |
| SST-2 | 67k | 1.8k | sentiment | acc. | movie reviews |
| | | | Similarity and Paraphrase Tasks | | |
| MRPC | 3.7k | 1.7k | paraphrase | acc./F1 | news |
| STS-B | 7k | 1.4k | sentence similarity | Pearson/Spearman corr. | misc. |
| QQP | 364k | **391k** | paraphrase | acc./F1 | social QA questions |
| | | | Inference Tasks | | |
| MNLI | 393k | **20k** | NLI | matched acc./mismatched acc. | misc. |
| QNLI | 105k | 5.4k | QA/NLI | acc. | Wikipedia |
| RTE | 2.5k | 3k | NLI | acc. | news, Wikipedia |
| WNLI | 634 | **146** | coreference/NLI | acc. | fiction books |

Wang et al. (2019)

# Results

| 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 | 79.9 | 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** |

- Huge improvements over prior work (even compared to ELMo)
- Effective at "sentence pair" tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Devlin et al. (2018)

# RoBERTa

- "Robustly optimized BERT"
- 160GB of data instead of 16 GB
- Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them

| **Model** | **data** | **bsz** | **steps** | **SQuAD** (v1.1/2.0) | **MNLI-m** | **SST-2** |
|---|---|---|---|---|---|---|
| RoBERTa | | | | | | |
| with BOOKS + WIKI | 16GB | 8K | 100K | 93.6/87.3 | 89.0 | 95.3 |
| + additional data (§3.2) | 160GB | 8K | 100K | 94.0/87.7 | 89.3 | 95.6 |
| + pretrain longer | 160GB | 8K | 300K | 94.4/88.7 | 90.0 | 96.1 |
| + pretrain even longer | 160GB | 8K | 500K | **94.6/89.4** | **90.2** | **96.4** |
| $\text{BERT}_{\text{LARGE}}$ | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 90.9/81.8 | 86.6 | 93.7 |

- New training + more data = better performance

Liu et al. (2019)

# GPT/GPT2/GPT3

# OpenAI GPT/GPT2

- "ELMo with transformers" (works better than ELMo)
- Train a single unidirectional transformer LM on long contexts
- GPT2: trained on 40GB of text collected from upvoted links from reddit
- 1.5B parameters — the largest of these models trained as of March 2019

| Parameters | Layers | $d_{model}$ |
|---|---|---|
| 117M | 12 | 768 |
| 345M | 24 | 1024 |
| 762M | 36 | 1280 |
| 1542M | 48 | 1600 |

- Because it's a language model, we can **generate** from it

Radford et al. (2019)

# OpenAI GPT2

| | |
|---|---|
| SYSTEM PROMPT (HUMAN-WRITTEN) | *Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.* |
| MODEL COMPLETION (MACHINE-WRITTEN, SECOND TRY) | The 19-year-old singer was caught on camera being escorted out of the store by security guards.<br><br>The singer was wearing a black hoodie with the label ‘Blurred Lines’ on the front and ‘Fashion Police’ on the back.<br><br>Scroll down for video<br><br>Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured)<br><br>The singer was wearing a black hoodie with the label ‘Blurred Lines’ on the front and ‘Fashion Police’ on the back<br><br>The singer was also wearing a pair of black-rimmed glasses, a black jacket, black jeans and black sandals.<br><br>She was carrying a pair of black and white striped gloves and a small black bag. |

slide credit: OpenAI

# GPT3

**Wei Xu**
@cocoweixu

I am finally jumping on the bandwagon of GPT-3 and read the 72-page long paper released by @OpenAI. Here is a summary of some technical details:

Model: largely the same as GPT-2, but alternate between dense/space attentions as in Sparse Transformers (OpenAI's 2019 work).

1/n

8:05 PM · Jul 21, 2020 · Twitter Web App

View Tweet activity

**81** Retweets **6** Quote Tweets **397** Likes

**Wei Xu** @cocoweixu · Jul 21, 2020

Replying to @cocoweixu

Model (cont'): sparse factorizations of the attention matrix to reduce computing time and memory use. trained 8 different sizes of models varying from 125M parameters (w/ 12 layers) to 175B parameters (w/ 96 layers). context window is set to 2048 tokens.

2/n

1 1 14

**Wei Xu** @cocoweixu · Jul 21, 2020

Data: filtered Common Crawl (410B tokens downsampled x0.44) + WebText dataset (19B x2.9) + two Internet-based book corpora (12Bx1.9, 55Bx0.43) + English Wiki (3B upsampled x3.4). efforts were made to remove overlap with evaluation datasets but unfortunately there was a bug.

3/n

1 2 15

**Wei Xu** @cocoweixu · Jul 21, 2020

https://twitter.com/cocoweixu/status/1285727605568811011

# GPT3

**Wei Xu** @cocoweixu · Jul 21, 2020

Training: a formidable amount of 3640 petaflop/s-days to train the largest GPT-3 language model (175B parameters). 1 petaflop/s-day is equivalent to 8 V100 GPUs at full efficiency of a day. gradually increased batch size, learning rate warmup/decay, and parallelism.

5/n

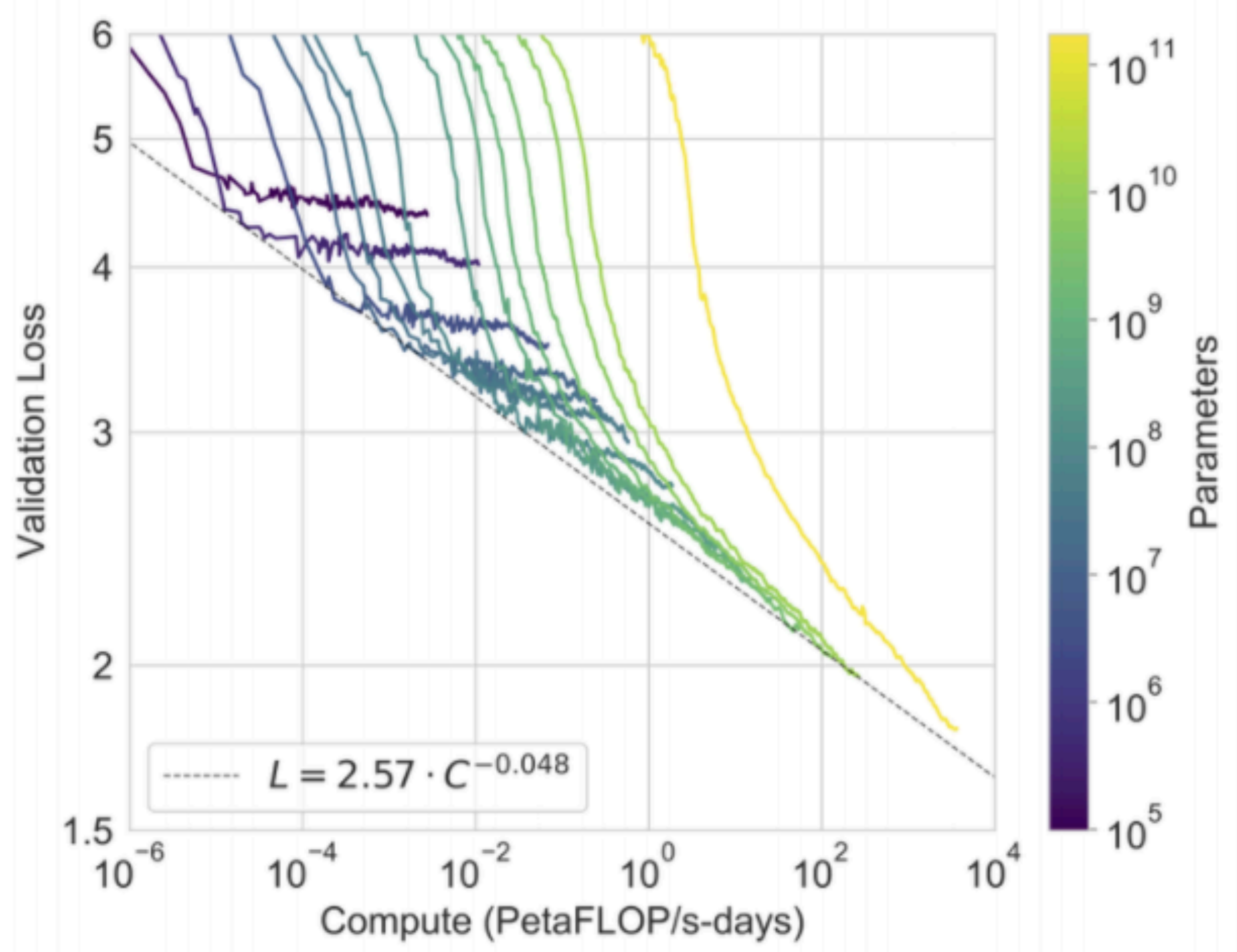



**Figure 3.1: Smooth scaling of performance with compute.** Performance (measured in terms of cross-entropy

1 4 18

https://twitter.com/cocoweixu/status/1285727605568811011

# Pre-Training Cost (with Google/AWS)

- BERT: Base $500, Large $7000
- Grover-MEGA: $25,000
- XLNet (BERT variant): $30,000 — $60,000 (unclear)
- This is for a single pre-training run…developing new pre-training techniques may require many runs
- *Fine-tuning* these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets)

https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

# Pre-training Cost

nytimes.com/2020/11/24/science/artificial-intelligence-ai-gpt3.html

The New York Times | **SCIENCE** | Meet GPT-3. It Has Learned to Code (and Blog and Argue).

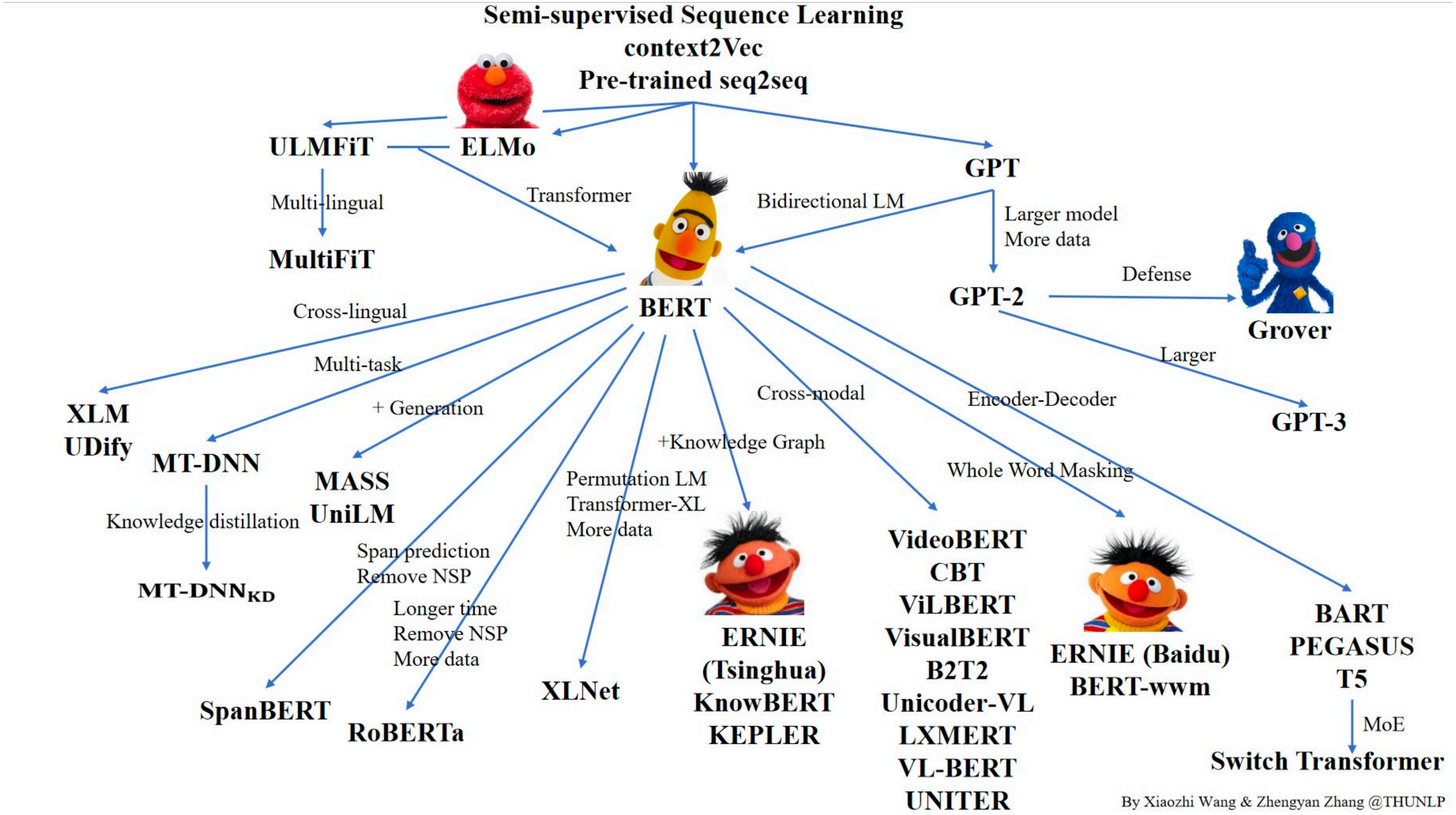
solely on language, they say, the system could already reach into other areas, whether computer programming, playing chess or generating guitar tabs.

But continuing to improve this technology is far from trivial. Processing all of that internet data requires a specialized supercomputer running for months on end, an undertaking that is enormously expensive. When asked if such a project ran into the millions of dollars, Sam Altman, OpenAI's chief executive, said the costs were actually "higher," running into the tens of millions.

Mr. Amodei, OpenAI's vice president for research, said there was still room to improve the technique, using more processing power to analyze more data. But he also said the approach might be close to running out of "juice."

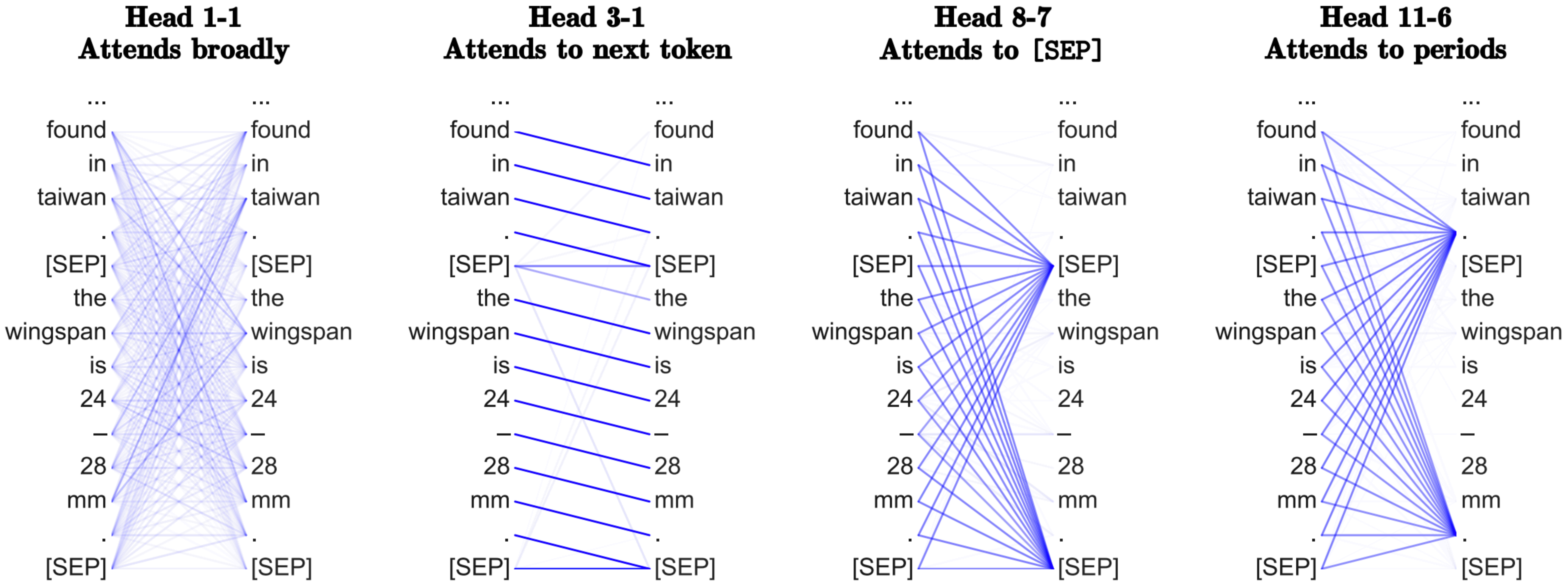
At the very least, GPT-3 is a new tool for a world of A.I. researchers and entrepreneurs, a way of building all sorts of new technologies and new products. Mr. Wrigley, the computer programmer, recently

# And a lot more ...



# Analysis

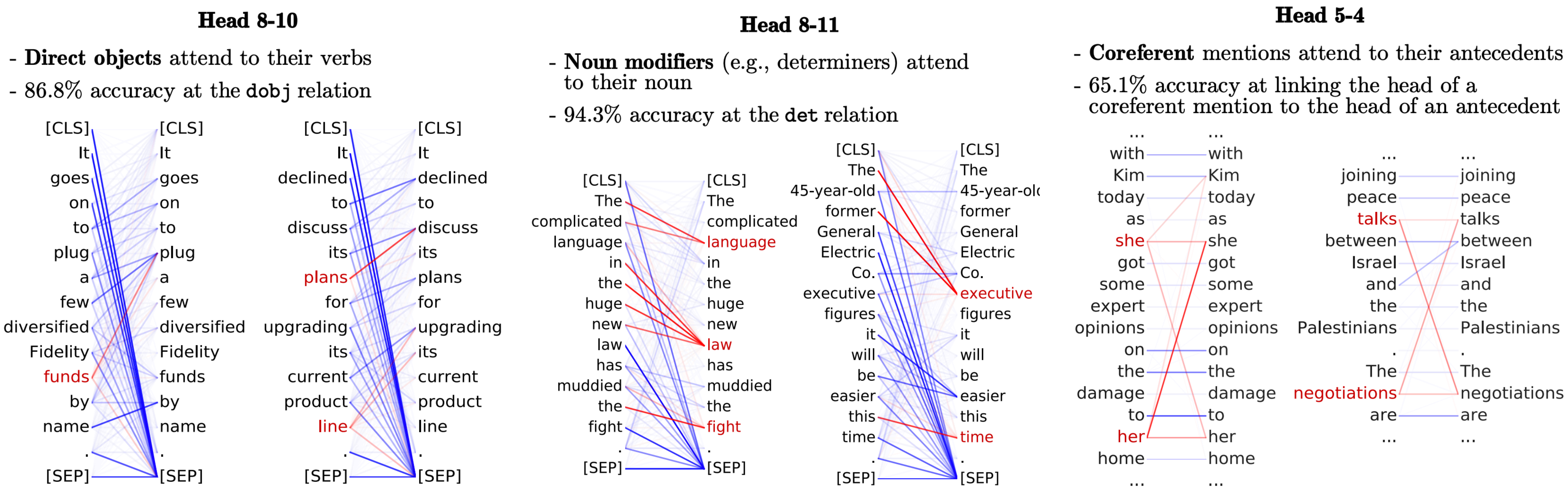
# What does BERT learn?




- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

# What does BERT learn?




- Still way worse than what supervised systems can do, but interesting that this is learned organically

Clark et al. (2019)

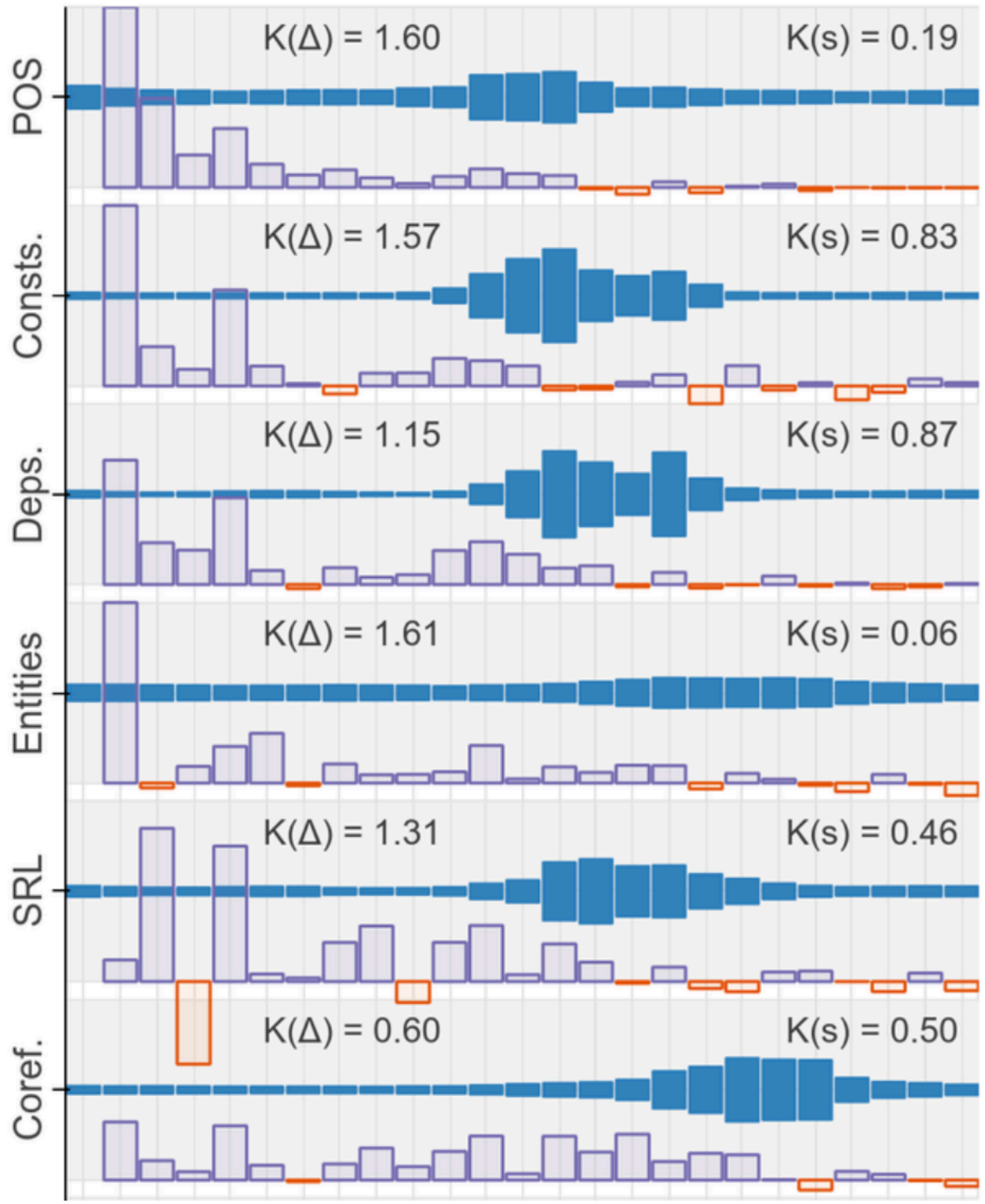
# Probing BERT

- Try to predict POS, etc. from each layer. Learn mixing weights

$$\mathbf{h}_{i,\tau} = \gamma_\tau \sum_{\ell=0}^{L} s_\tau^{(\ell)} \mathbf{h}_i^{(\ell)}$$

representation of wordpiece *i* for task $\tau$

- Plot shows *s* weights (blue) and performance deltas when an additional layer is incorporated (purple)
- BERT "rediscovers the classical NLP pipeline": first syntactic tasks then semantic ones

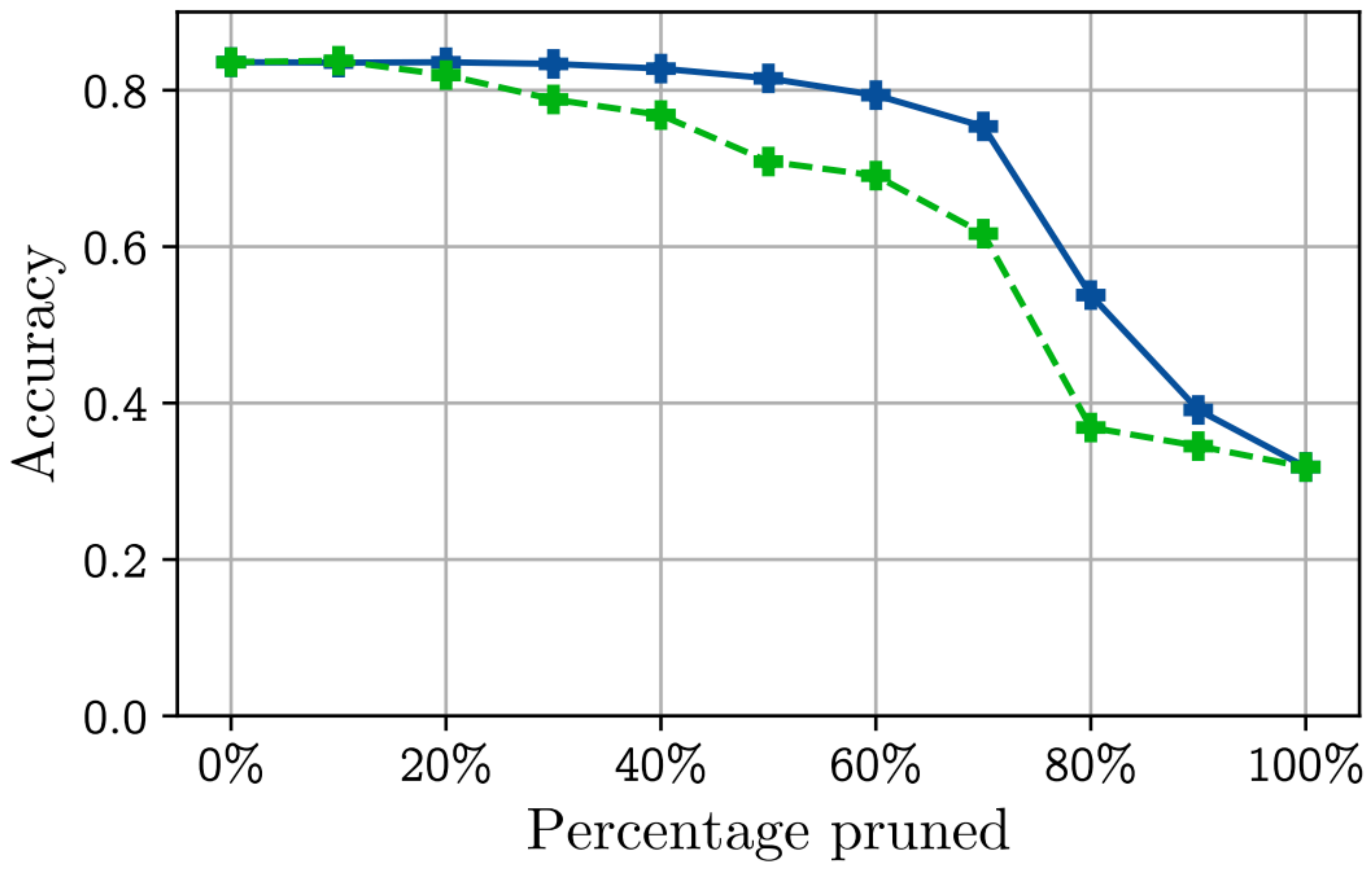



Tenney et al. (2019)

# Compressing BERT

- Remove 60+% of BERT's heads with minimal drop in performance

- DistilBERT (Sanh et al., 2019): nearly as good with half the parameters of BERT (via knowledge distillation)




(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT according to $I_h$ (solid blue) and accuracy difference (dashed green).

Michel et al. (2019)

# Open Questions

- BERT-based systems are state-of-the-art for nearly every major text analysis task
- These techniques are here to stay, unclear what form will win out
- Role of academia vs. industry: no major pretrained model has come purely from academia
- Cost/carbon footprint: a single model costs $10,000+ to train (though this cost should come down)