CSE 156 | Lecture 8: Pretraining: Encoders

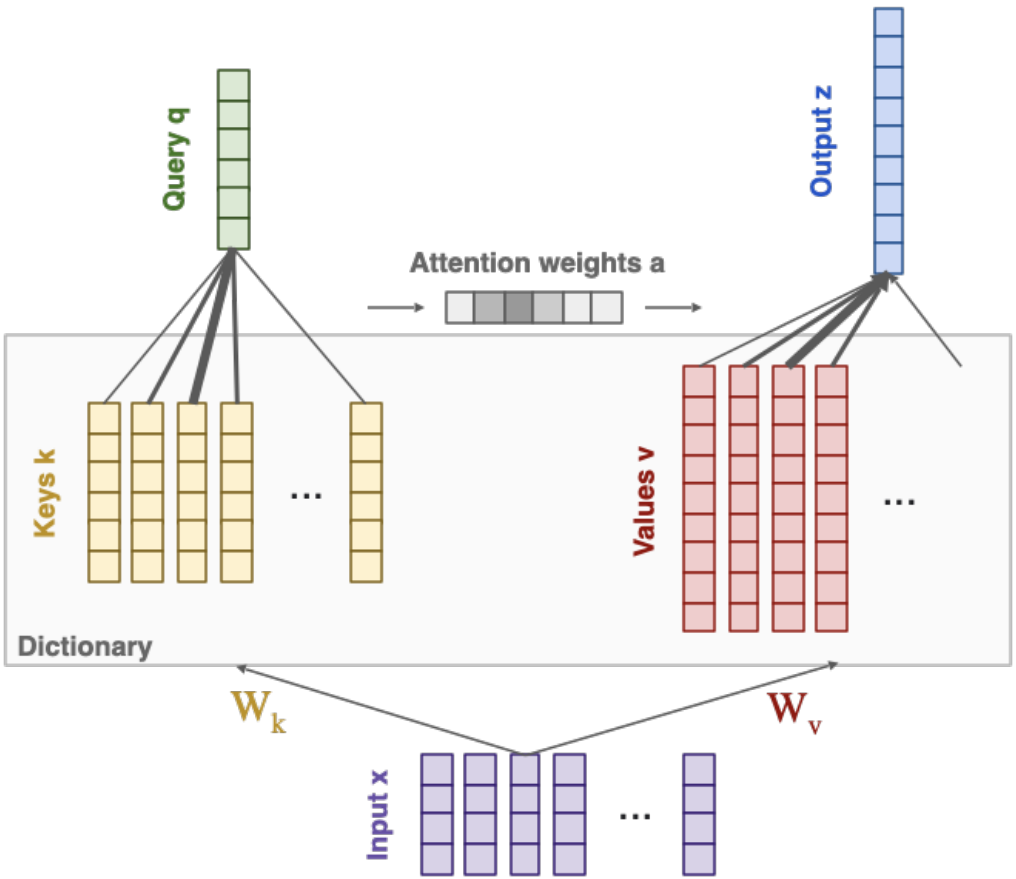
Ndapa Nakashole

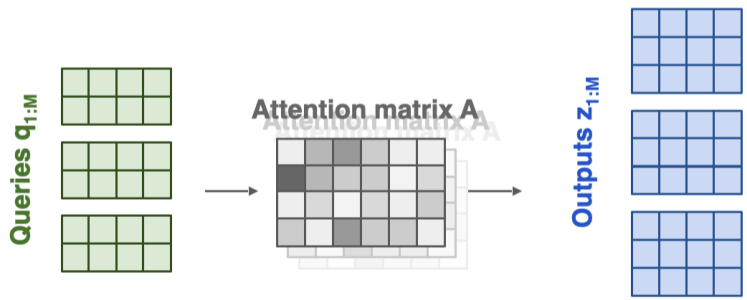
October 24, 2024

Administrative matters

- **Quiz 1**: out tomorrow

Recap: Self-Attention, Multi-Head Attention

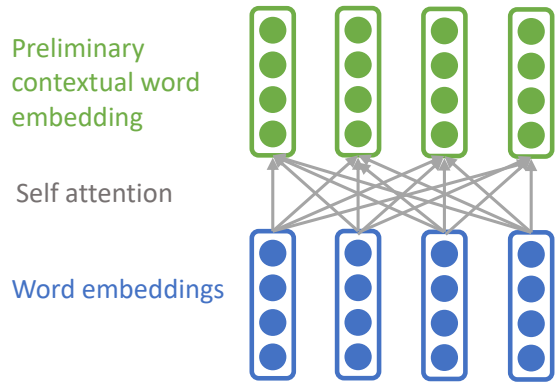




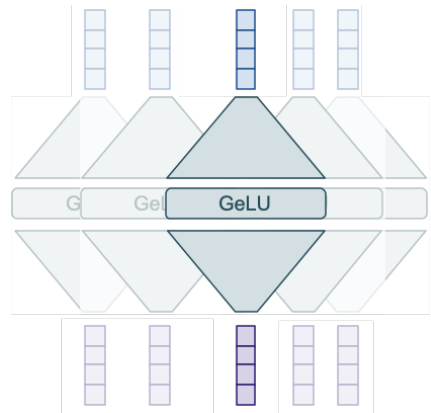
Recap: Stacked Transformer Blocks

- **Multi-Head Self-Attention**
  - Aggregates information from different representation subspaces in parallel
  - $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
- **Position-wise Feedforward Networks**
  - Two linear transformations with an activation g in between.
  - $\text{FFN}(x) = W_2(GeLU(W_1 x + b_1)) + b_2$

Attention is the "communication" stage



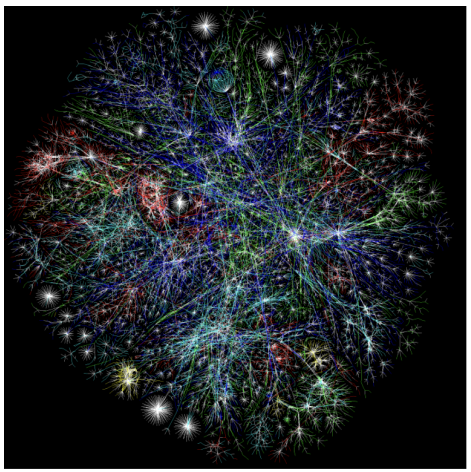
Feedforward is the "think" stage



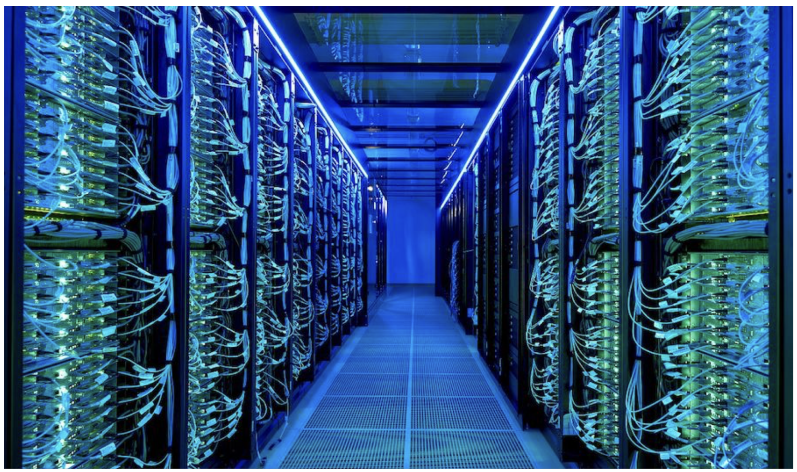
Today

1. Overview of LM pretraining
2. Representative LMs (encoder-only)
   - ELMo
   - BERT
   - DeBERTa
   - ...
3. MidTerm Information

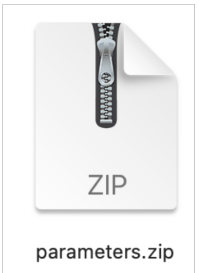
Pretraining ≈ compressing the internet



Chunk of the internet,
~10TB of text



6,000 GPUs for 12 days, ~$2M
~1e24 FLOPS



~140GB file

---

* Numbers for Llama2; 70B model

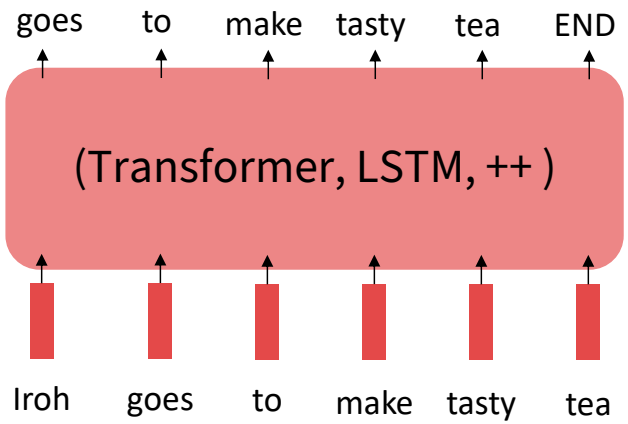
credit: A. Kaparthy

The Pretraining / Finetuning Paradigm

- **Pretraining**: can improve NLP applications by serving as **parameter initialization**

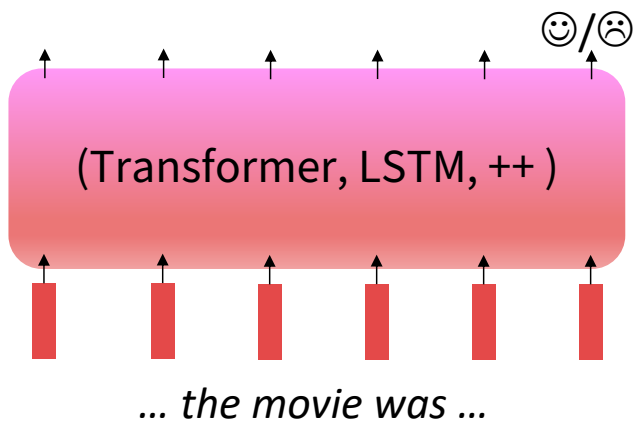
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



- **Pretraining**: The model learns general knowledge from a large, diverse dataset.
- **Finetuning**: The model adapts this general knowledge to a specific task with a smaller, task-specific dataset.

Pretraining and Finetuning

From a "training neural nets" perspective:

- Pretraining provides parameters $\hat{\theta}$ by approximating

$$\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta)$$

  - The **pretraining loss** $\mathcal{L}_{\text{pretrain}}$ is used to adjust pretrained parameters θ.

- Then, finetuning approximates

$$\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta)$$

  starting at $\hat{\theta}$.

  - The **finetuning loss** $\mathcal{L}_{\text{finetune}}$ fine-tunes the parameters θ, starting from pretrained $\hat{\theta}$.

Where does this data come from?

Model	**Training Data**
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed

Learning Representations

- **Representation learning**: map raw data to more useful forms
- **Supervised learning**: Learn representations on large labeled datasets
  - Image classifiers learn broadly useful features; rerepresenting raw pixels as edges, textures, and objects (Krizhevsky et al., 2012); (Zeiler & Fergus, 2014)
- **Unsupervised learning**: scale beyond specific tasks or domains
  - But what is the right **objective**?

Learning Representations: Early OpenAI experiment

LSTM character Language Model, Trained on 82 million product reviews

Learning to Generate Reviews and Discovering Sentiment

Alec Radford [1] **Rafal Jozefowicz** [1] **Ilya Sutskever** [1]

Abstract

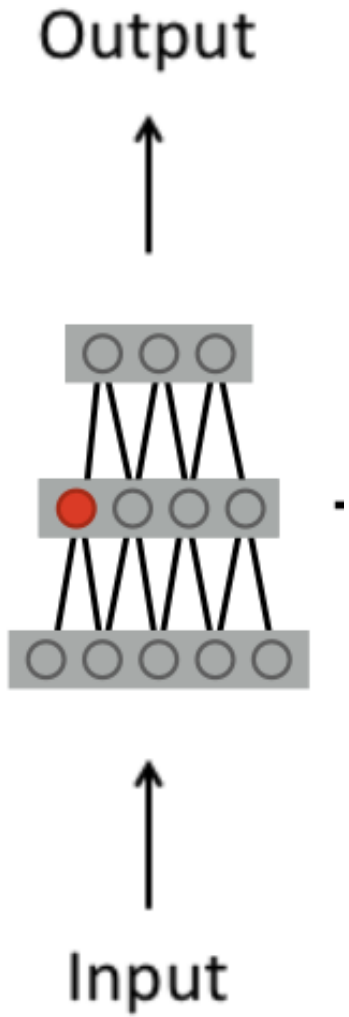
We explore the properties of byte-level recurrent language models. When given sufficient amounts of capacity, training data, and compute time, the representations learned by these models include disentangled features corresponding to high-level concepts. Specifically, we find a single unit which performs sentiment analysis. These representations, learned in an unsupervised manner, achieve state of the art on the binary subset of

it is now commonplace to reuse these representations on a broad suite of related tasks - one of the most successful examples of transfer learning to date (Oquab et al., 2014).

There is also a long history of unsupervised representation learning (Olshausen & Field, 1997). Much of the early research into modern deep learning was developed and validated via this approach (Hinton & Salakhutdinov, 2006) (Huang et al., 2007) (Vincent et al., 2008) (Coates et al., 2010) (Le, 2013). Unsupervised learning is promising due to its ability to scale beyond only the subsets and domains

with better coverage of target tasks. Finally, our work encourages further research into language modelling as it demonstrates that the standard language modelling objective with no modifications is sufficient to learn high-quality representations.

What is in a Neuron?



Learning Representations: the Sentiment Neuron

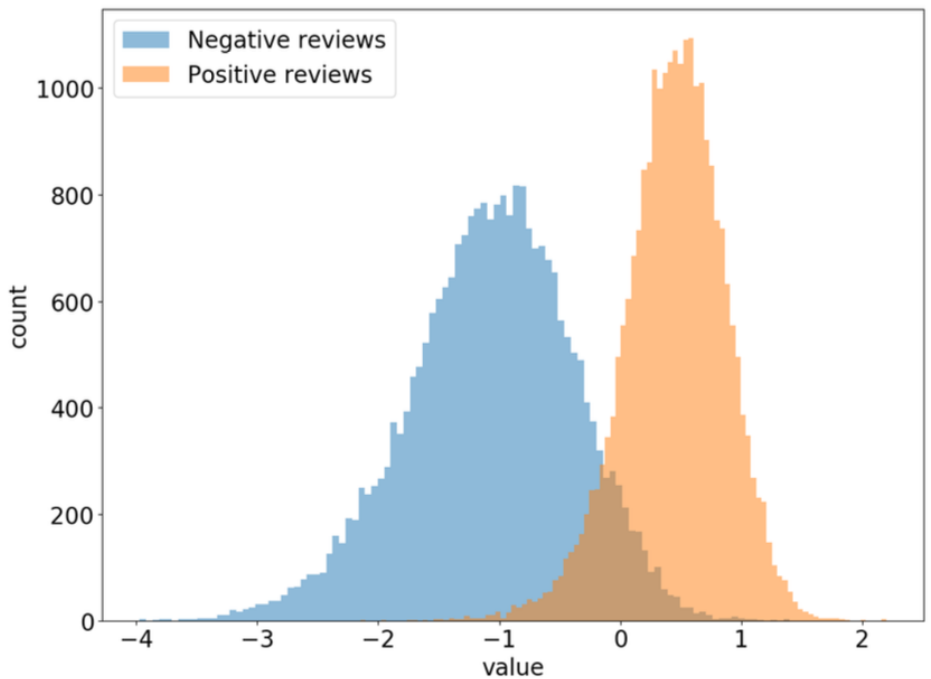


Figure 3. Histogram of cell activation values for the sentiment unit on IMDB reviews.

- **Sentiment Neuron**: a single unit which performs sentiment analysis
- **Very data efficient**: matches the performance of strong baselines trained on full datasets with only a handful of labeled examples
- 92.30% test accuracy on the binary Stanford Sentiment Treebank (PA1 dataset)

Learning to Generate Reviews and Discovering Sentiment, Radford et al. 2017

Sentiment Neuron

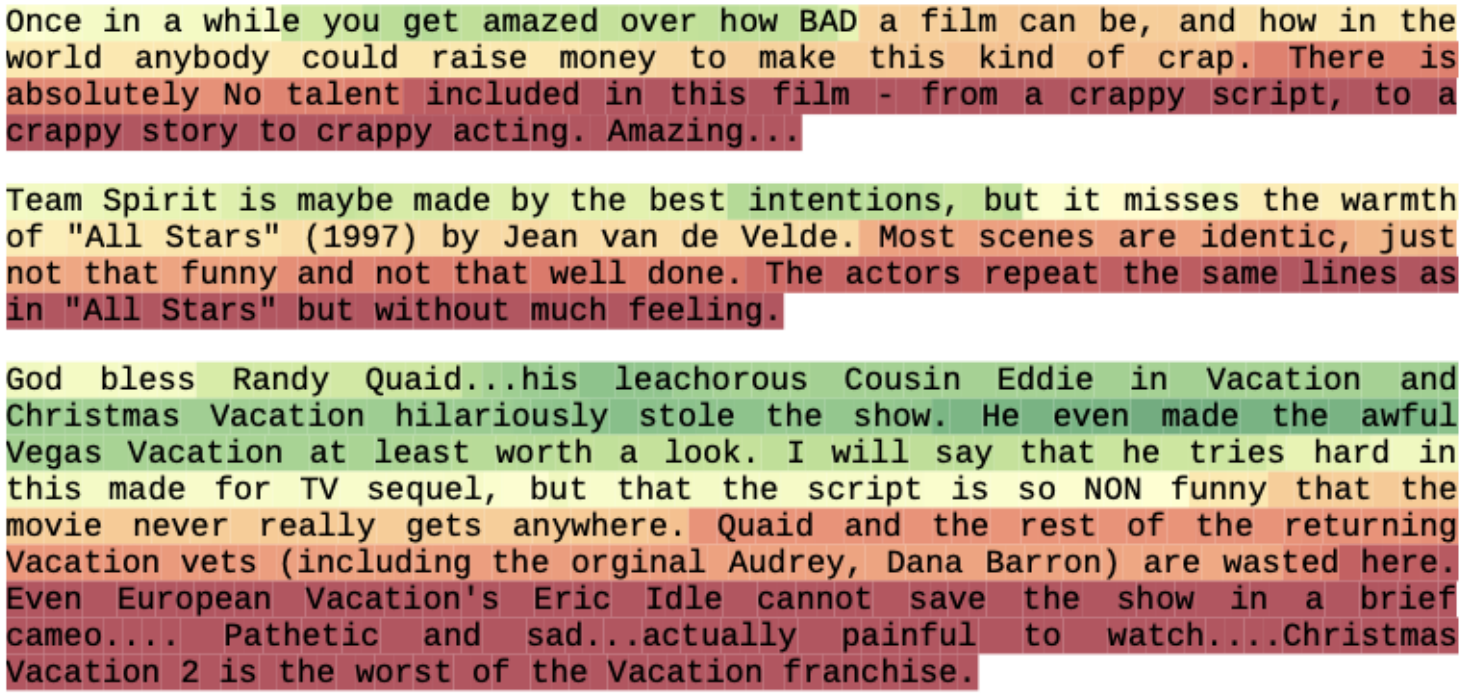
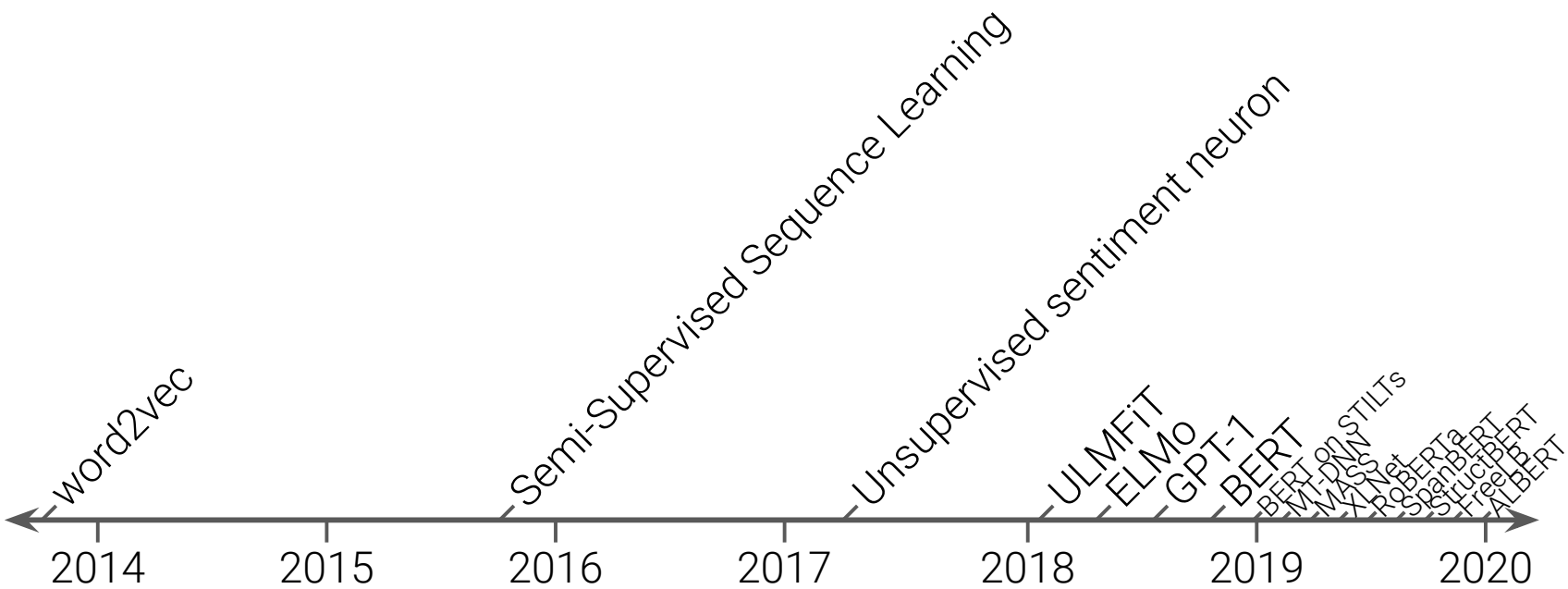


Figure 4. Visualizing the value of the sentiment cell as it processes six randomly selected high contrast IMDB reviews. Red indicates negative sentiment while green indicates positive sentiment. Best seen in color.

Learning Representations



Learning Representations: what task to train on?

- NLP: want **language representations** of words, sentences, ..., that capture their **meaning**
- **Guess the missing word**: requires knowing a lot about language
- **Word2Vec, Glove, ...,** use a lot of data but in a weak way

Meaning is really Context-Dependent

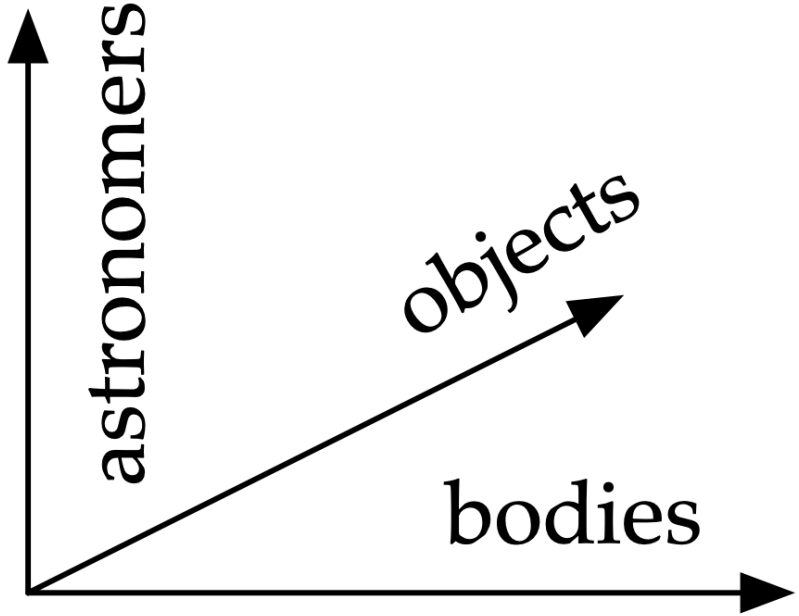
Recall the adage we mentioned at the beginning of the course: *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)

But . . .

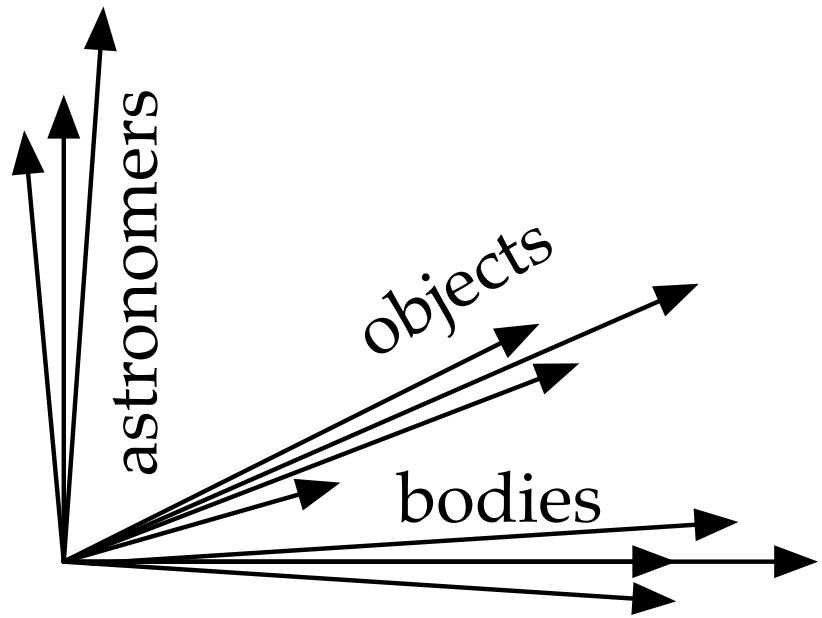
(J. R. Firth 1935)

"... the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously"

Contextualized Word Embeddings



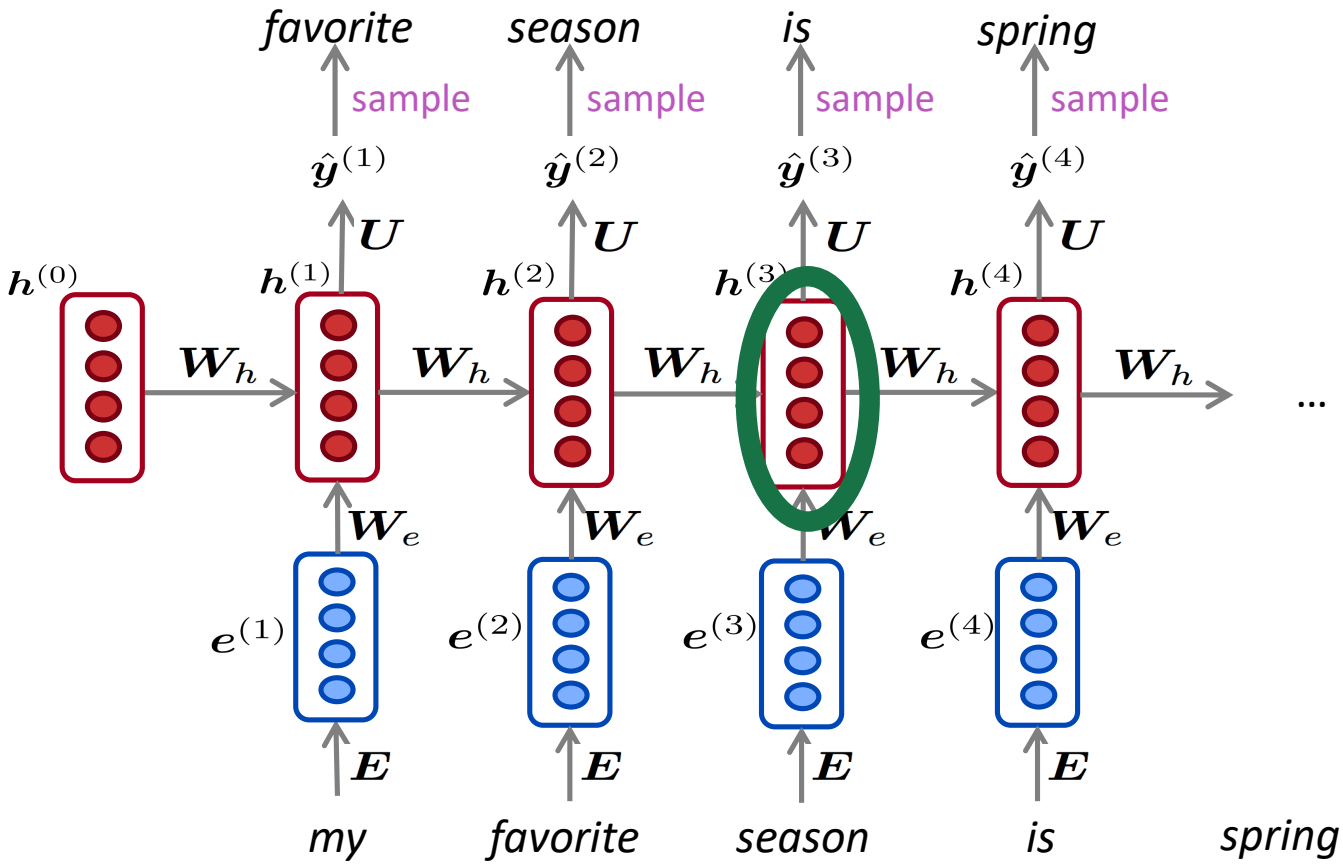
Word2Vec, Glove, etc: give a single embedding for each word regardless of context



really wanted: different embedding for each word in each context it appears

Insight: We have had the solution all along!

- LMs are producing context-specific word representations at each position — **hidden states** at each step (**ELMo** Peters et al. 2018)

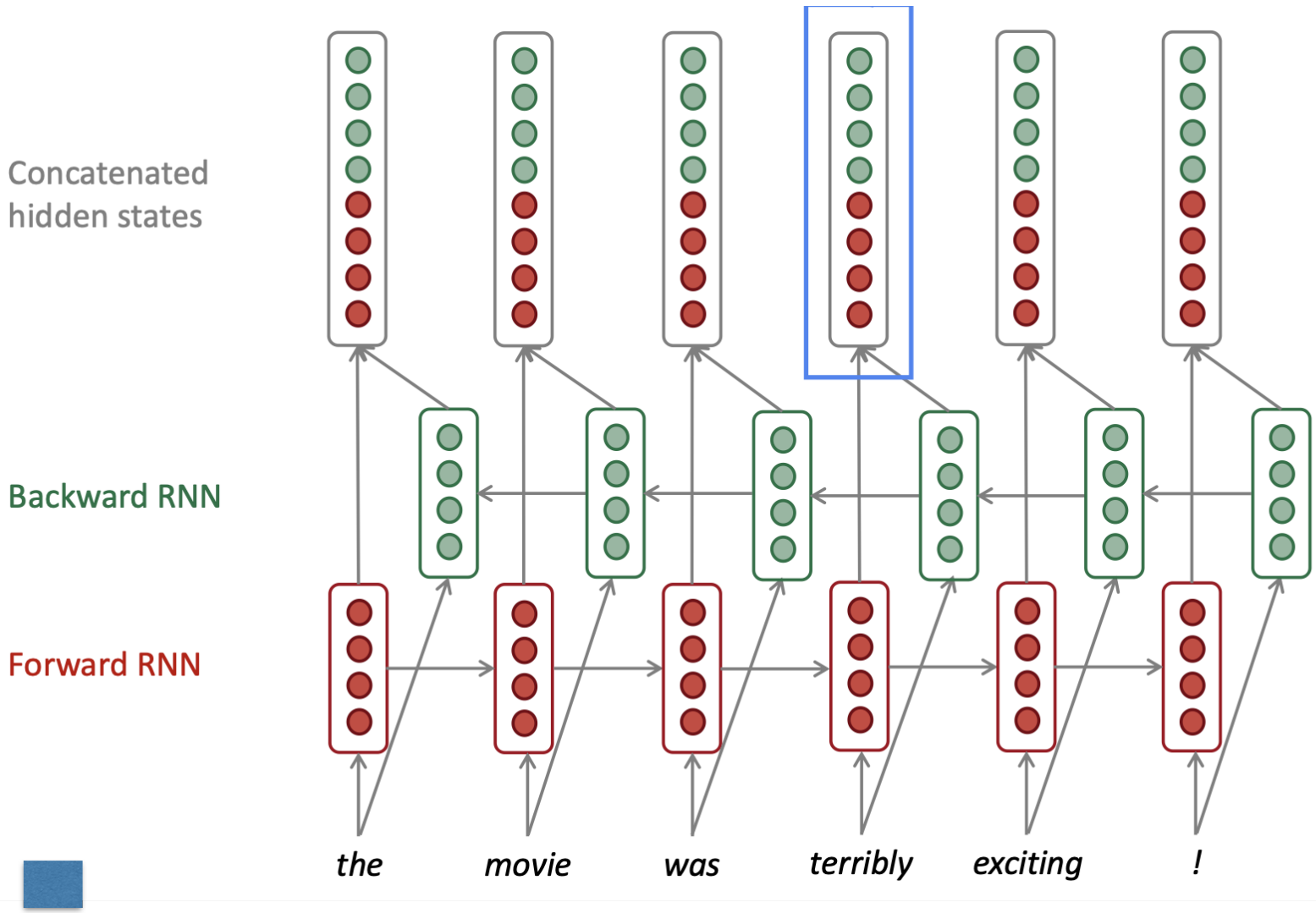


ELMo

- Learn word token vectors using **long contexts**, trained on LM objective
- Train a deep Bi-LSTM, freeze and use all its layers in prediction
  - Use the embeddings as a drop-in replacement for Word2Vec, Glove, etc.

Deep contextualized word representations, Peters et al. 2018

Recall deep bi-LSMTs



ELMo Architecture

- **ELMo** is built using a deep **Bi-LSTM** model, where each word token's embedding is a function of:

$$\mathbf{h}_k^{LM} = [\overrightarrow{\mathbf{h}_k^{LM}}; \overleftarrow{\mathbf{h}_k^{LM}}]$$

- $\overrightarrow{\mathbf{h}_k^{LM}}$ and $\overleftarrow{\mathbf{h}_k^{LM}}$ represent the hidden states of forward and backward LSTMs for word k.

$$\mathbf{ELMo}_k = \gamma \sum_{j=0}^{L} s_j \mathbf{h}_{k,j}^{LM}$$

Where:

- $\mathbf{h}_{k,j}^{LM}$ is the j-th layer of the LSTM for word k.
- s_j: trainable scalar weights.
- γ: trainable scalar parameter to scale the final embedding.

AI2's ELMo came out Spring 2018: race to scale up!

- ELMo: Huge gains across many high-profile tasks: NER, question answering, semantic role labeling (similar to parsing), etc.

GPT	BERT	GPT-2
June 2018	Oct 2018	Feb 2019
Training	Training	Training
240 GPU days	256 TPU days	~2048 TPU v3 days according to a reddit thread
	~320–560 GPU days	
OpenAI	Google AI	OpenAI

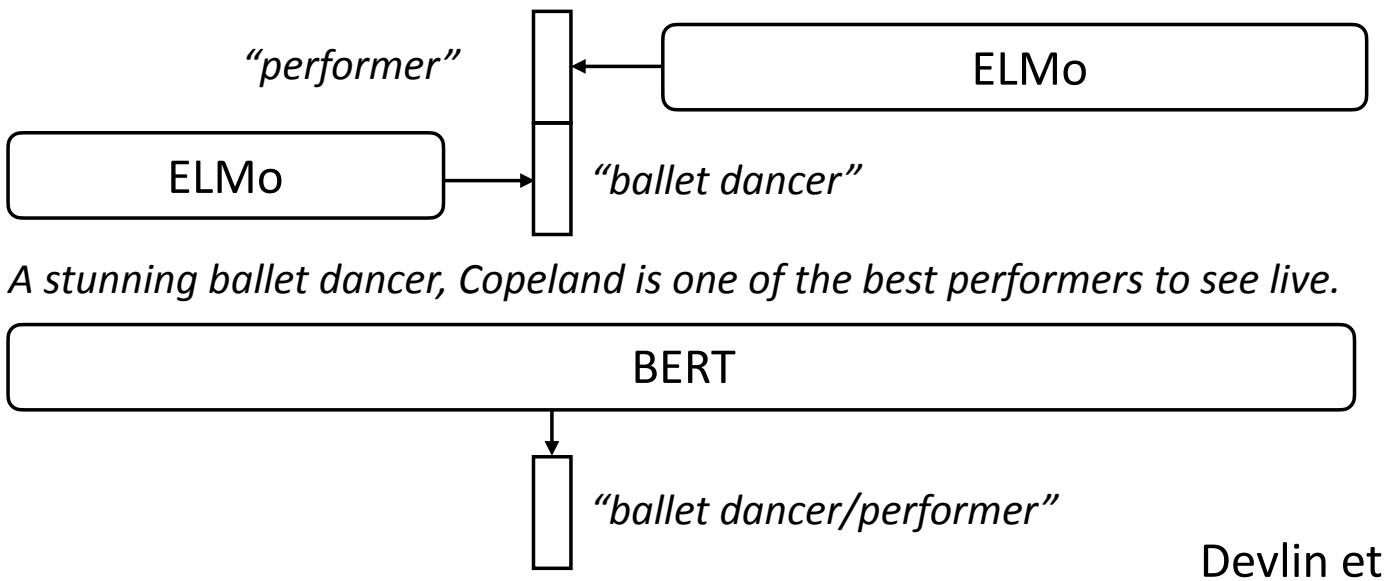
GPU day → amount of processing time required to complete a task when using one GPU.

BERT

- What if pretraining is **not just for the embeddings?**
- Four major changes compared to ELMo:
  - Transformers instead of LSTMs
  - Bidirectional model with "Masked LM" objective instead of standard LM
  - Fine-tune instead of freeze at test time (not just a source of word embeddings!)

BERT

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

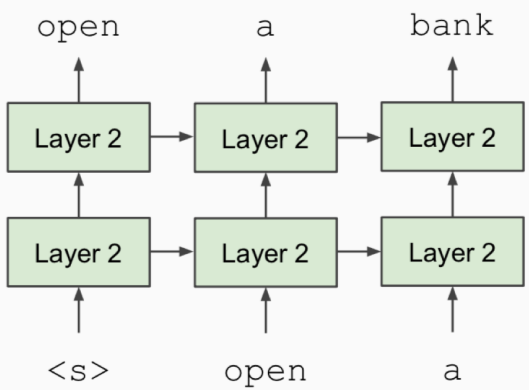


Devlin et al. (2019)

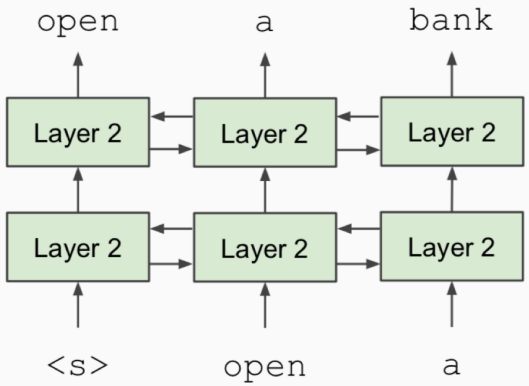
Bidirectional Context w/o Cheating

- Encoders get bidirectional context, so we can't do language modeling!

Unidirectional context
Build representation incrementally



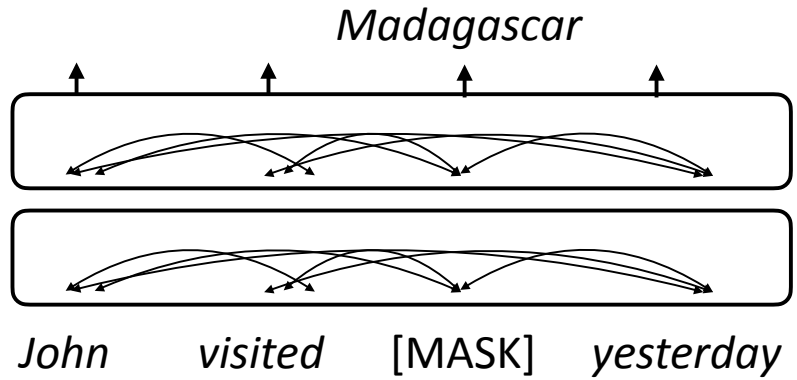
Bidirectional context
Words can "see themselves"



BERT: Masked Language Modeling

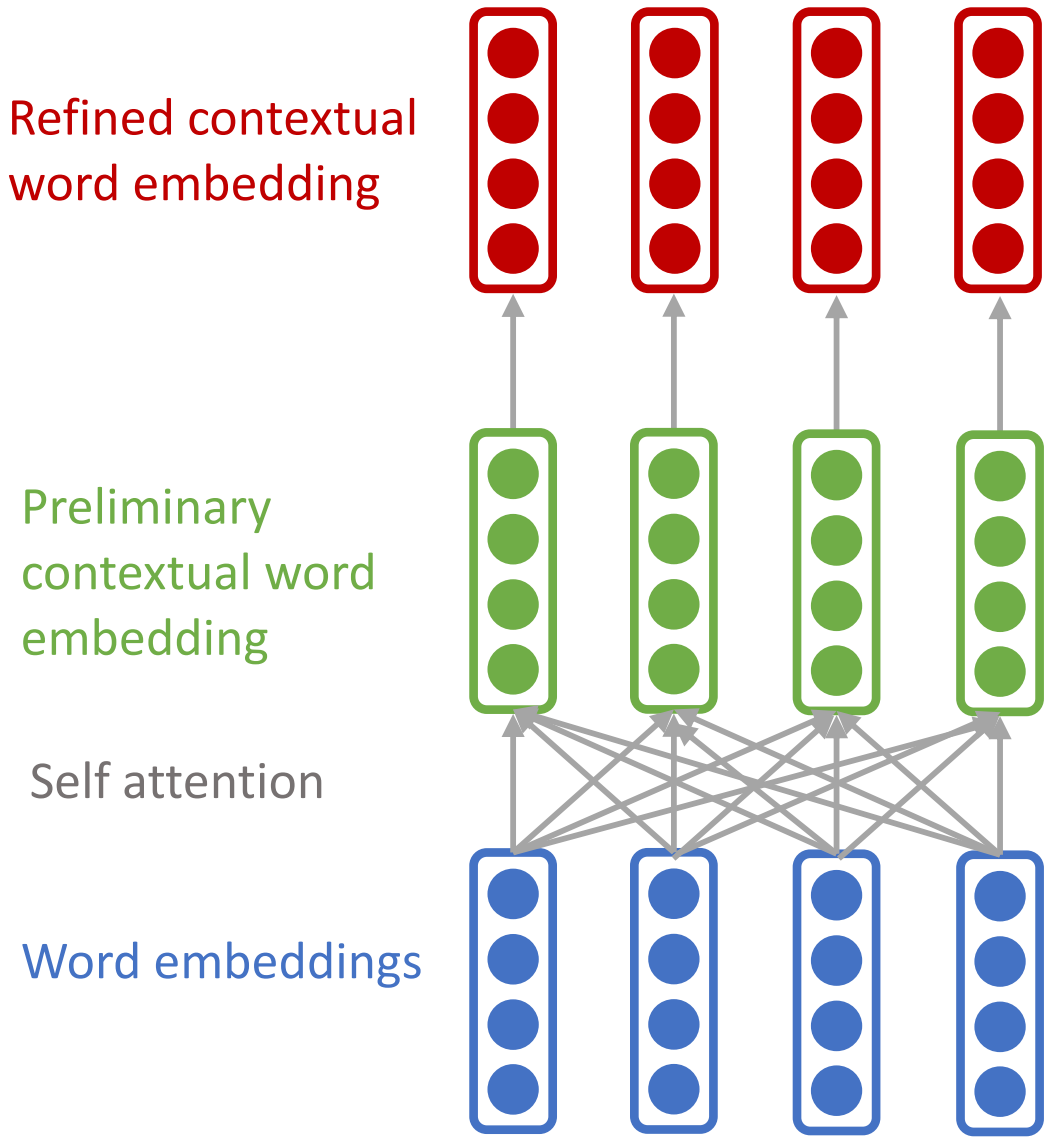
Devlin et al., 2018 proposed the "Masked LM" objective and **released the weights of a pretrained Transformer**

- Take a chunk of text, mask out 15% of the tokens, and try to predict them



Optimize: $\mathrm{P}($ Madagascar | John visited [MASK] yesterday)

Transformer Hidden States



Pre-training BERT: Masked Language Model

Masked Language Model (MLM) is one of the two pre-training tasks in BERT.

- Given an input sentence, BERT uses the context of surrounding words to predict the masked word:

$$P(\text{word}_i \mid \text{context}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_i)$$

- Where:
  - $\mathbf{h}_i$: The hidden state of the Transformer for word i.
  - $\mathbf{W}$: Output projection matrix.

BERT Masking Strategy

store gallon

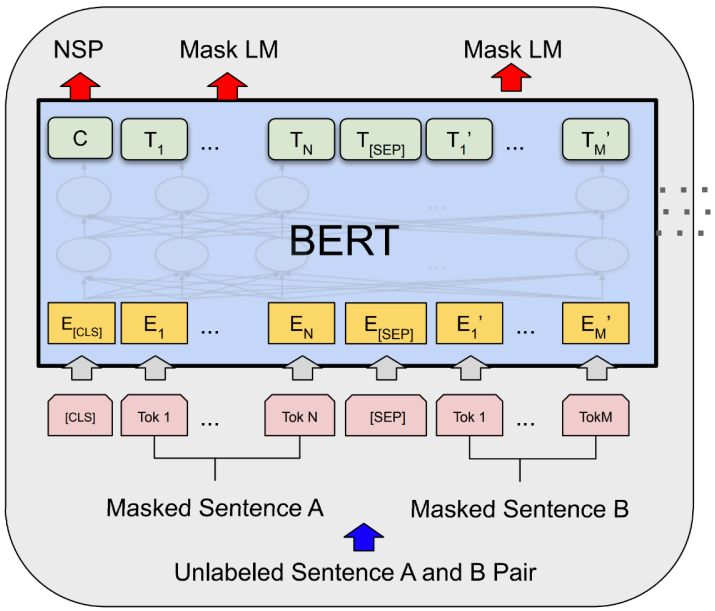
↑ ↑

the man went to the [MASK] to buy a [MASK] of milk

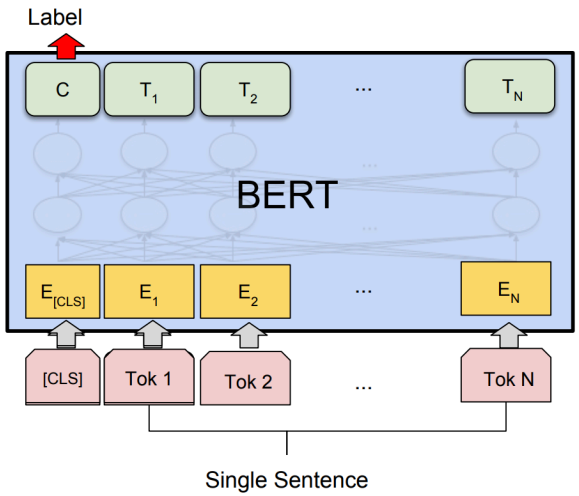
- Too little masking: Too expensive to train
- Too much masking: Not enough context

BERT Architecture: Base and Large

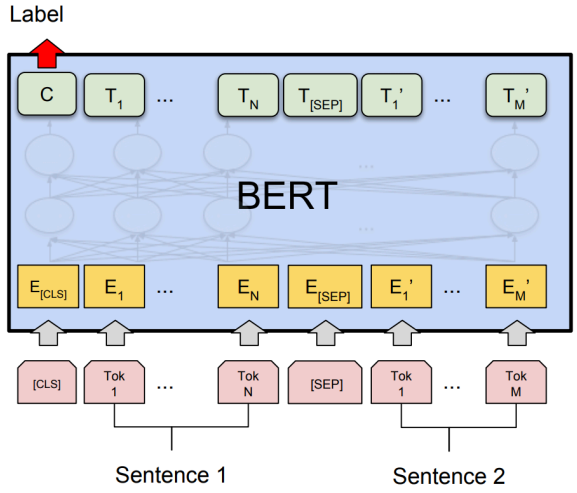
- 12/24 layers, 768/1024 hidden units, 12/16 attention heads
- 110M/340M parameters
- 512 tokens per sequence
- 16/32 sequences per batch
- Positional embeddings and segment embeddings, 30k word pieces



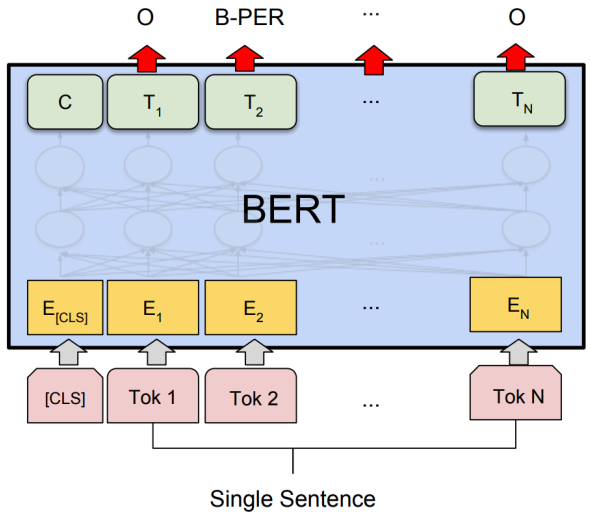
Fine-tuning BERT on Downstream Tasks



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



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



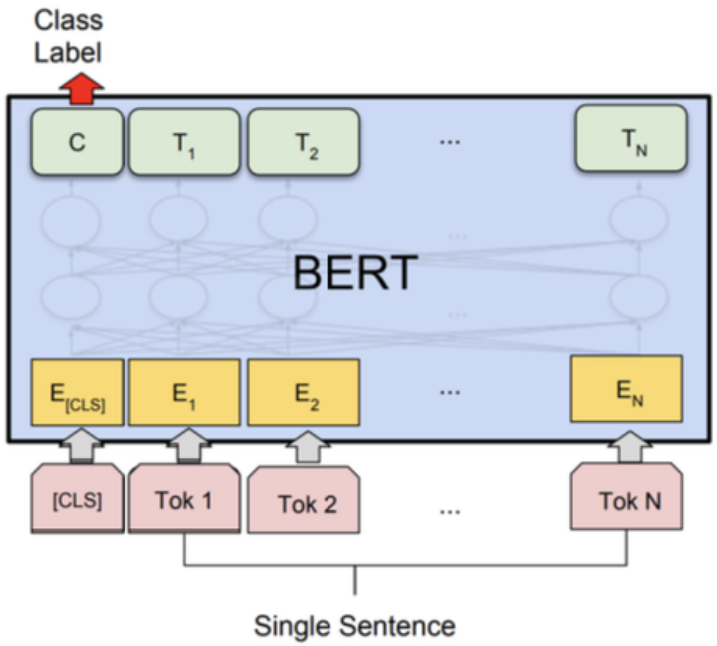
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

- Artificial [CLS] token is used as the vector to do classification
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

What can BERT NOT do?

- BERT **cannot generate** text (at least not in an obvious way)
- Could put MASK at the end repeatedly, but this is slow and in practice lacks coherence
  - **Example:** "The cat sat on the [MASK]" $\rightarrow$ "The cat sat on the mat"
- Masked language models are intended to be used primarily for "analysis" tasks

Fine-tuning BERT



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

- Finetune all parameters (both the newly added layer and the pretrained weights)
- Use a small learning rate (e.g., 1e-5)
- Train for a small number of epochs (e.g, 3 epochs)
- Led to SOTA results on many NLU tasks

Results: GLUE Benchmark

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

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**

- Huge improvements over prior work (even compared to ELMo)
- Effective at "sentence pair" tasks

Devlin et al. 2018

BERT Variants

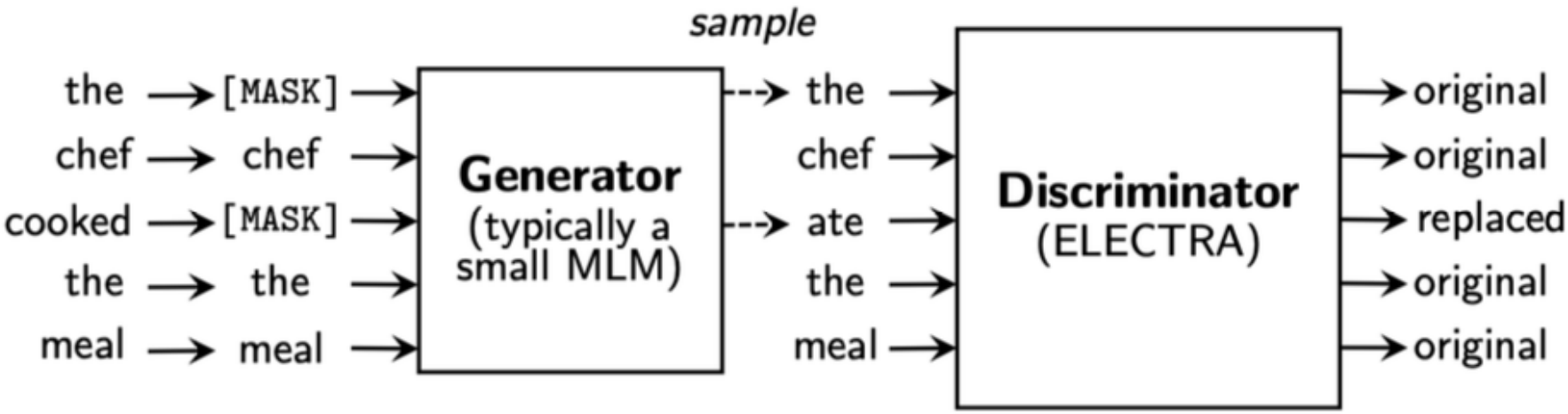
RoBERTa ("Robustly optimized BERT approach")

- Train BERT with more data, longer, bigger batches, and more training steps. 160GB of data instead of 16 GB
- Dynamic masking: 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

Liu et al. (2019)

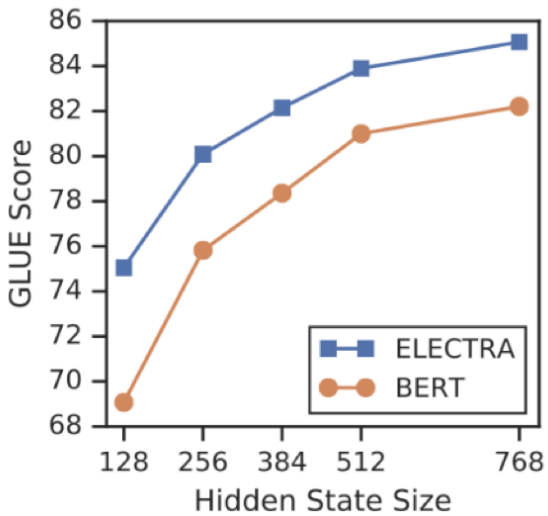
ELECTRA

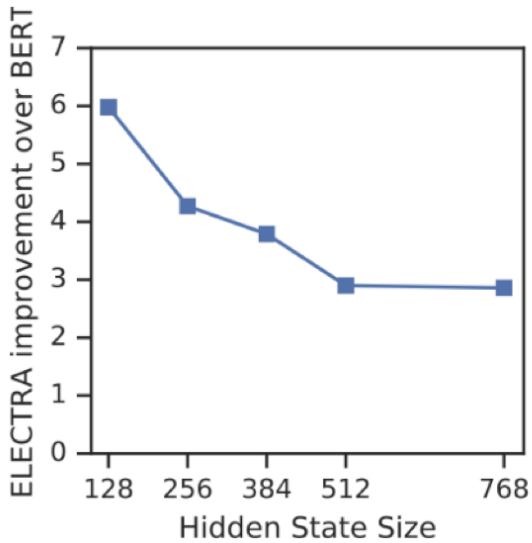


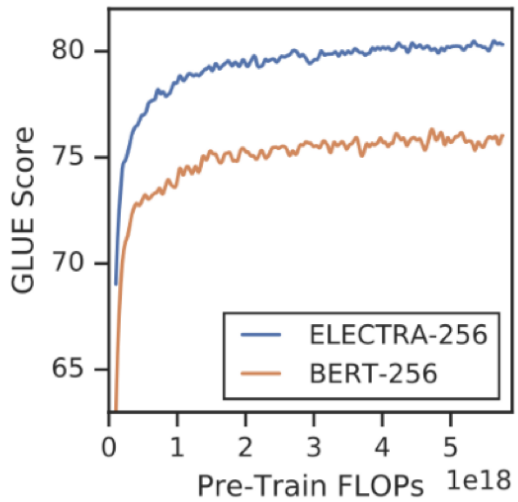
- BERT is inefficient, loss only computed for 15% of tokens, 85 % of tokens are wasted
- Have a prediction for every token, not just the masked ones
- Instead of predicting the masked word, predict whether a word is replaced by a different word

Clark et al. (2020)

ELECTRA Results







- Large improvements with smaller models
- Faster training
- Effective approach if you don't have large compute for pretraining

Clark et al. (2020)

DeBERTa

He et al. 2021: DeBERTa: Decoding-enhanced BERT with Disentangled Attention

- Each word is **represented by two vectors**: one for content, one for position; attention weights are based on content and relative positions.

$$\boldsymbol{Q_c} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{q},\boldsymbol{c}}, \quad \boldsymbol{K_c} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{k},\boldsymbol{c}}, \quad \boldsymbol{V_c} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{v},\boldsymbol{c}},$$

$$\boldsymbol{Q}_r = \boldsymbol{P}\boldsymbol{W}_{\boldsymbol{q},r}, \quad \boldsymbol{K}_r = \boldsymbol{P}\boldsymbol{W}_{\boldsymbol{k},r}$$

$$\tilde{A}_{i,j} = \underbrace{\boldsymbol{Q}_i^{\boldsymbol{c}} \cdot \boldsymbol{K}_j^{\boldsymbol{c}}(\boldsymbol{b})}_{\text{(a) content-to-content}}$$

$$+ \underbrace{\boldsymbol{Q}_i^{\boldsymbol{c}} \cdot \boldsymbol{K}_{\boldsymbol{\delta}(i,j)}^{\top}}_{\text{content-to-position}}$$

$$+ \underbrace{\boldsymbol{K}_j^{\boldsymbol{c}} \cdot \boldsymbol{Q}_{\delta(j,i)}^{\top}}_{\text{(c) position-to-content}}$$

$$\boldsymbol{H}_o = \operatorname{softmax}\left(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}}\right)\boldsymbol{V}_c$$

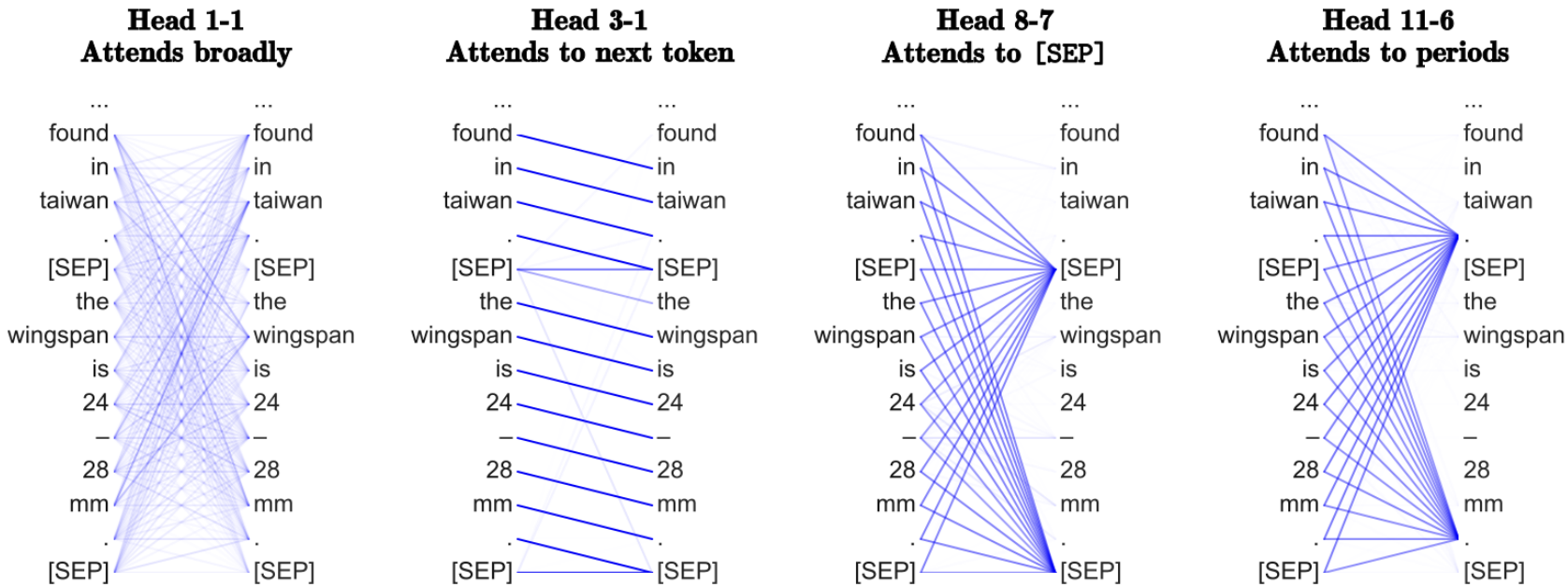
DeBERTa Results

Model	CoLA Mcc	QQP Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT$_{large}$	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa$_{large}$	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet$_{large}$	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
ELECTRA$_{\text{large}}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa$_{large}$	**70.5**	92.3	**91.1/91.1**	96.8	**92.8**	**95.3**	**88.3**	**91.9**	**90.00**

He et al. (2021)

What does BERT look at?: positional and "no-op" heads

- Strong results suggest pretraining teaches models abiut language, but **what specific linguistic features do they learn?**: Analyse the *144 attention heads* in BERT-base (12 layers x 12 heads)



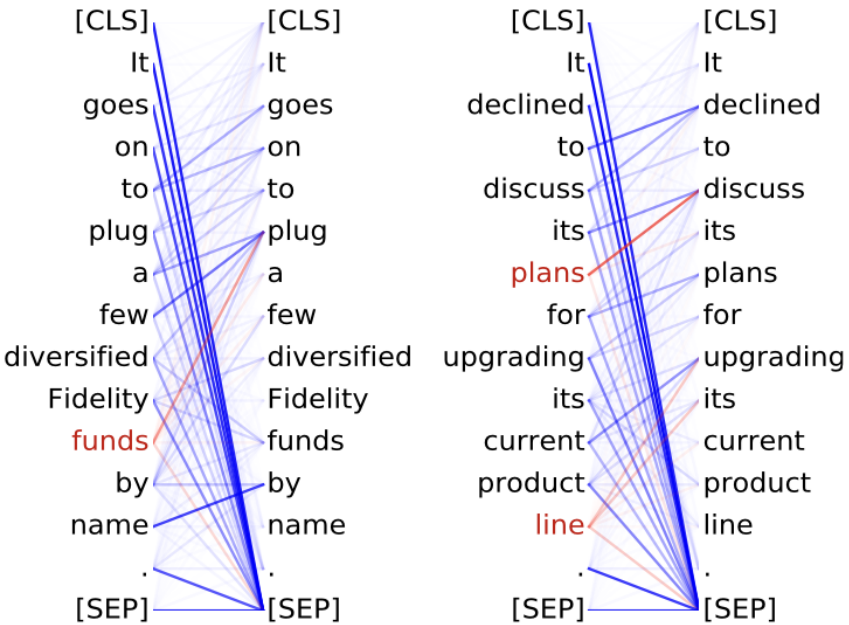
- Most heads pay **little attention on the current token**. Some heads specialize in attending to **the next or previous** token

What Does BERT Look at? An Analysis of BERT's Attention, Clark et al. (2019)

What does BERT look at?: syntactic depedency heads

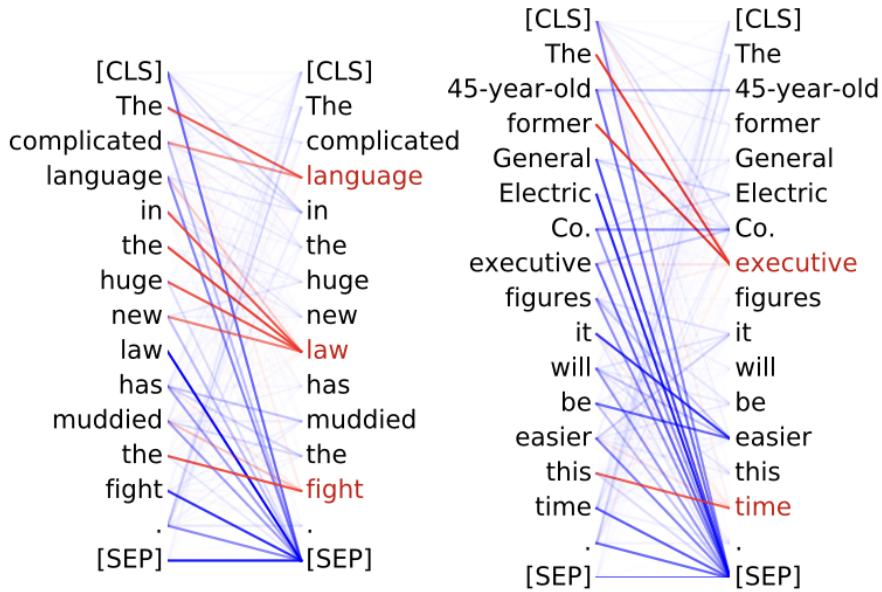
Head 8-10

- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation



Head 8-11

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation



Clark et al. (2019)

What does BERT look at?: syntactic depedency heads

- Different heads specialize in different aspects of syntactic relationships

Relation	Head	Accuracy	Baseline
All	7-6	34.5	26.3 (1)
prep	7-4	66.7	61.8 (-1)
pobj	9-6	**76.3**	34.6 (-2)
det	8-11	**94.3**	51.7 (1)
nn	4-10	70.4	70.2 (1)
nsubj	8-2	58.5	45.5 (1)
amod	4-10	75.6	68.3 (1)
dobj	8-10	**86.8**	40.0 (-2)
advmod	7-6	48.8	40.2 (1)
aux	4-10	81.1	71.5 (1)
poss	7-6	**80.5**	47.7 (1)
auxpass	4-10	**82.5**	40.5 (1)
ccomp	8-1	**48.8**	12.4 (-2)
mark	8-2	**50.7**	14.5 (2)
prt	6-7	**99.1**	91.4 (-1)

Pretraining Data and Models

What to Train on?

"... commercial language models rarely provide any information about their data; even open models rarely release datasets they are trained on, or an exact recipe to reproduce them. "

Luca Soldaini[♥α] Rodney Kinney[♥α] Akshita Bhagia[♥α] Dustin Schwenk[♥α]
David Atkinson[α] Russell Authur[α] Ben Bogin[αω] Khyathi Chandu[α]
Jennifer Dumas[α] Yanai Elazar[αω] Valentin Hofmann[α] Ananya Harsh Jha[α]
Sachin Kumar[α] Li Lucy[β] Xinxi Lyu[ω] Nathan Lambert[α] Ian Magnusson[α]
Jacob Morrison[α] Niklas Muennighoff Aakanksha Naik[α] Crystal Nam[α]
Matthew E. Peters[σ] Abhilasha Ravichander[α] Kyle Richardson[α] Zejiang Shen[τ]
Emma Strubell[χα] Nishant Subramani[χα] Oyvind Tafjord[α] Pete Walsh[α]
Luke Zettlemoyer[ω] Noah A. Smith[αω] Hannaneh Hajishirzi[αω]
Iz Beltagy[α] Dirk Groeneveld[α] Jesse Dodge[α]

Kyle Lo[♥α]

[α]Allen Institute for AI [β]University of California, Berkeley [χ]Carnegie Mellon University
[σ]Spiffy AI [τ]Massachusetts Institute of Technology [ω]University of Washington

{lucas,kylel}@allenai.org

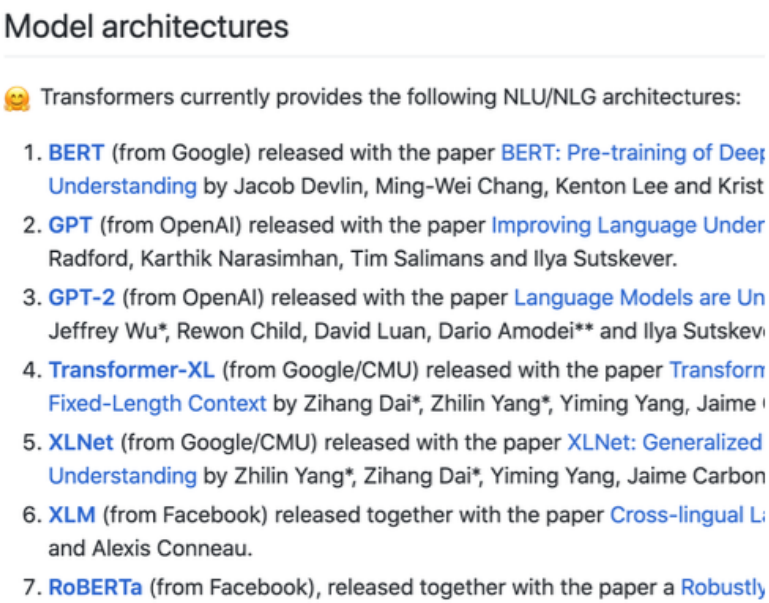
Dolma: 3T Tokens

Source	Doc Type	UTF-8 bytes *(GB)*	Documents *(millions)*	Unicode words *(billions)*	Llama tokens *(billions)*
Common Crawl	web pages	9,022	3,370	1,775	2,281
The Stack	code	1,043	210	260	411
C4	web pages	790	364	153	198
Reddit	social media	339	377	72	89
PeS2o	STEM papers	268	38.8	50	70
Project Gutenberg	books	20.4	0.056	4.0	6.0
Wikipedia, Wikibooks	encyclopedic	16.2	6.2	3.7	4.3
Total		**11,519**	**4,367**	**2,318**	**3,059**

Using BERT, DeBERTa, ...

HuggingFace Transformers: big open-source library with most pre-trained architectures implemented, weights available

Lots of standard models...

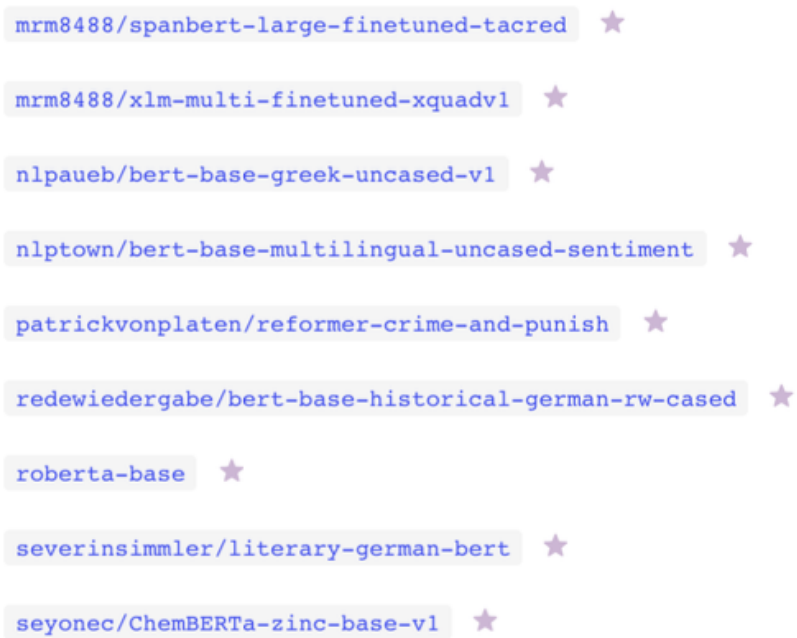


Model architectures

🤗 Transformers currently provides the following NLU/NLG architectures:

1. **BERT** (from Google) released with the paper BERT: Pre-training of Deep Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Krist
2. **GPT** (from OpenAI) released with the paper Improving Language Under Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
3. **GPT-2** (from OpenAI) released with the paper Language Models are Un Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskev
4. **Transformer-XL** (from Google/CMU) released with the paper Transform Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime
5. **XLNet** (from Google/CMU) released with the paper XLNet: Generalized Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbon
6. **XLM** (from Facebook) released together with the paper Cross-lingual L and Alexis Conneau.
7. **RoBERTa** (from Facebook), released together with the paper a Robustly

and "community models"

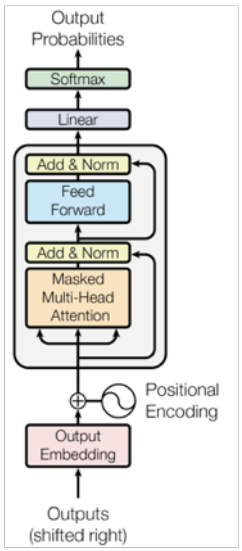


- `mrm8488/spanbert-large-finetuned-tacred`
- `mrm8488/xlm-multi-finetuned-xquadv1`
- `nlpaueb/bert-base-greek-uncased-v1`
- `nlptown/bert-base-multilingual-uncased-sentiment`
- `patrickvonplaten/reformer-crime-and-punish`
- `redewiedergabe/bert-base-historical-german-rw-cased`
- `roberta-base`
- `severinsimmler/literary-german-bert`
- `seyonec/ChemBERTa-zinc-base-v1`

Next: decoders and encoder-decoder models

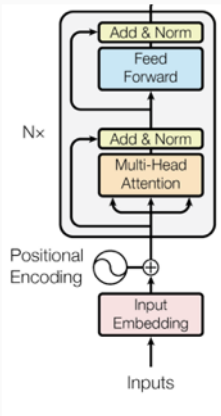
Decoder-only GPT

[sat_]



Encoder-only BERT

[*] [*] **[sat_]** [*] **[the_]** [*]



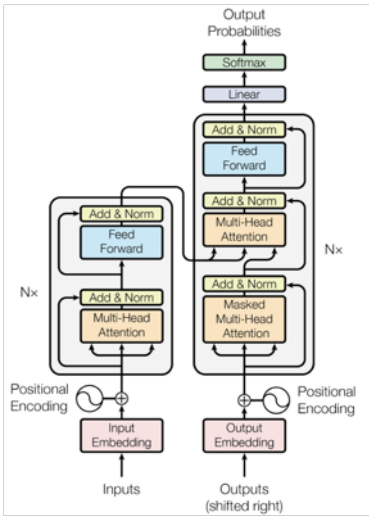
[The_] [cat_] **[MASK]** [on_] **[MASK]** [mat_]

Enc-Dec T5

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



Translate EN-DE: This is good.

Summarize: state authorities dispatched…

Is this toxic: You look beautiful today!

Takeaways

- BERT and its variants are powerful models for a range of NLP tasks
- These models have enabled big advances in NLI and QA specifically
- Next: pre-trained decoders (GPT-i) and encoder-decoder models (T5)

Midterm Exam Information

Midterm Exam Information

- The midterm exam will be held on **Tuesday, November 12th, 2024** during the regular class time.
- It will be **closed book** but you can bring a **one-page notes sheet** (front and back).
- It **must be completed in one sitting**.
- Calculators and electronic devices are **not allowed**.
  - Final grade contribution: 25%

Midterm Exam Topics: 1/3

Foundations of Neural NLP

- **Softmax classifer**: model, training objective, gradient update
- **Classification**: how weights work in this setting
- **Optimization**: stochastic gradient descent, impact of step size on optimization
- **Feedforward neural networks**: definition, activation functions
- **Training neural networks**
- **Word embeddings**: skip-gram model, definition, properties
- **Subword tokenization**:
- **Deep averaging networks**: model from PA1, limitations of the model

Midterm Exam Topics: 2/3

Modern Language Models: Background

- n-**gram language modeling**: basic definition, count-based parameter estimation
- **Recurrent neural networks**: model, training RNNs
- **Sequence to sequence models**: model, training, inference

Midterm Exam Topics: 3/3

Modern Language Models: Key Ingredients

- **Self-attention**: mathematical definition, how it works
- **Transformers**: architecture details
- **Transformer language modeling**: everything from PA2, including training, inference, and evaluation (perplexity)
- **Encoders**: masked language modeling and BERT
- **Decoders**: causal language modeling, training, inference
- **Inference in decoders**: nucleus sampling, beam search

That's it!

NLP Projects Information

There is no final project in this course - CSE 156. But given excitement in the field, you might be thinking of doing a project perhaps on your own - here are some ideas.

How to Find an Interesting Place to Start?

- **ACL Anthology** for NLP papers (see past 3 years): https://aclanthology.org/
- Online proceedings of major ML conferences:
  - **NeurIPS**: https://papers.nips.cc
  - **ICLR**: https://openreview.net/group?id=ICLR.cc
  - **ICML**
- Check out online **preprint servers**, especially: https://arxiv.org
- Even better: find an interesting problem in the world!
  - **Hal Varian:** How to Build an Economic Model in Your Spare Time: https://people.ischool.berkeley.edu/ hal/Papers/how.pdf

Old Deep Learning for NLP vs New Deep Learning for NLP

2010-2018: Early Deep Learning Revival

- **Focus:** Defining and exploring better deep learning architectures.
- E.g., Paper: A new summarization architecture that uses **copying attention**, a probabilistic gate is added to allow the model to directly copy words from the input to the output (Abigail See et al., ACL 2017).

2019-2024: Modern Deep Learning in NLP

- **Focus:** Fine-tuning pre-trained models (which fixes the architecture).
- The action has shifted from architecture design to optimizing for specific tasks and domains.

2024 NLP ... recommended for practical projects

pip install transformers # By **Huggingface** 🤗

```
# not quite runnable code but gives the general idea….
from transformers import BertForSequenceClassification, AutoTokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased’)
model.train()
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased’)
fine_tuner = Trainer( model=model,  args=training_args, train_dataset=train_dataset,
    eval_dataset=test_dataset )
fine_tuner.train()
eval_dataset = load_and_cache_examples(args, eval_task, tokenizer, evaluate=True)
results = evaluate(model, tokenizer, eval_dataset, args)
```

Wanting to do LLM inference on local machine, see Llama.cpp

llama.cpp

Roadmap / Project status / Manifesto / ggml

Inference of Meta's LLaMA model (and others) in pure C/C++

Types of Projects

This is not an exhaustive list, but a starting point for ideas.

1. **Task/Application Exploration:**
   - Find a task of interest and explore how to solve it, often with an existing model.
   - This could involve a real-world task, or **Kaggle** competition or **shared task** dataset.
2. **Complex Neural Architecture Implementation:**
   - Implement a complex (perhaps novel) neural architecture and explore its performance on some dataset.
3. **Prompting or In-context Learning:**
   - Explore the use of **prompting** of LLMs, find effective prompts, etc
   - This category expects experiments with numbers and ablations.

Project Ideas Continued

4. **Model Adaptation:**
   - Adapt a model to a new task or domain, or explore how to make a model more robust to certain types of errors or inputs (more on this ...)

5. **Data Augmentation:**
   - Explore how data augmentation can improve model performance, or investigate the types of augmentation that are most effective.

6. **Parameter-efficient fine-tuning:**
   - Techniques like **low-rank adaptation** and **parameter-efficient fine-tuning**.

7. **Ethical/Social Impact:**
   - Investigate the ethical or social implications of a model or application, or explore ways to mitigate bias or other ethical concerns.

Project Ideas Continued - Scaling up and down

Scaling models up and down is a key focus in NLP research.

- **Scaling models up:**
  - Building large models is impactful but requires substantial compute resources.
  - **be realistic** about the compute you can access.
- **Scaling models down:**
  - Building small, performant models is equally valuable. This could be a great project idea!
  - Techniques include **model pruning** and **model quantization**.
  - How well can you do QA in a **6GB** or **500MB** model? See EfficientQA.

Behavioral Evaluation

- There is interest in evaluating and improving models for **something other than accuracy**
- E.g., **Robustness**: How well models perform when faced with noisy, unexpected, or adversarial input, ...

Evaluating Models

- Evaluation of ML models involves assessing their ability to perform specific tasks
- **Typical Approach:** models are evaluated by computing aggregate metrics on a test set
- **Limitations of Aggregate Metrics:**
  - Similar to how an IQ test roughly estimates human intellect, **aggregate metrics provide a rough estimate of model performance**
  - They may conceal systematic failures

High accuracy on test data does not guarantee avoidance of systematic failures

Behavioral Evaluation in Machine Learning

Behavioral Evaluation

Assessing how well a model performs under a variety of real-world conditions and on specific types of input data

Example: Sentiment Analysis

Example: When creating a sentiment classification model, might check that model works for:

- **Double negatives**
- **Sarcasm**
- **Emojis**

Key Aspects of Behavioral Evaluation

- **Robustness Checks**: how well does model perform on inputs that are designed to **test the limits of its capabilities or simulate adversarial conditions**.
- **Slice-based Analysis**: Looking at model **performance on specific, meaningful subsets of data** to identify if there are any discrepancies that need attention.
- **Fairness and Bias:** ML systems can perpetuate societal biases and safety concerns, with issues like:
  - Systematic misclassification in medical imaging
  - Buolamwini and Gebru's study shows poorer performance of facial classification models on darker-skinned people compared to lighter-skinned people

Challenge of Behavioral Evaluation

- Behavioral evaluation is important but challenging, **requires discovering patterns and systematic failures** in model behavior
- First must decide which **capabilities a model should have**
  - There can be a large number of requirements in complex domains, which would be impossible to list and test
  - Instead, **might need domain experts** to define the capabilities that a model should have
  - As endusers interact with the model in products and services, they also provide feedback on the limitations or expected behaviors
- Very good example paper: Ribeiro et al ACL 2020

Beyond Accuracy: Behavioral Testing of NLP Models with CheckList

Marco Tulio Ribeiro[1] **Tongshuang Wu**[2] **Carlos Guestrin**[2] **Sameer Singh**[3]
[1]Microsoft Research [2]University of Washington [3]University of California, Irvine
marcotcr@gmail.com {wtshuang,guestrin}@cs.uw.edu sameer@uci.edu

Abstract

Although measuring held-out accuracy has been the primary approach to evaluate generalization, it often overestimates the performance of NLP models, while alternative approaches for evaluating models either focus on individual tasks or on specific behaviors. Inspired by principles of behavioral testing in software engineering, we introduce CheckList, a task-agnostic methodology for testing NLP models. CheckList includes a matrix of general linguistic *capabilities* and *test types* that facilitate comprehensive test ideation, as well as a software tool to generate a large and diverse number of test cases quickly. We illustrate the utility of CheckList with tests for three tasks, identifying critical failures in both commercial and state-of-art models. In a user study, a team responsible for a commercial sentiment analysis model found new and actionable bugs in an extensively tested model. In another user

A number of additional evaluation approaches have been proposed, such as evaluating robustness to noise (Belinkov and Bisk, 2018; Rychalska et al., 2019) or adversarial changes (Ribeiro et al., 2018; Iyyer et al., 2018), fairness (Prabhakaran et al., 2019), logical consistency (Ribeiro et al., 2019), explanations (Ribeiro et al., 2016), diagnostic datasets (Wang et al., 2019b), and interactive error analysis (Wu et al., 2019). However, these approaches focus either on individual tasks such as Question Answering or Natural Language Inference, or on a few capabilities (e.g. robustness), and thus do not provide comprehensive guidance on how to evaluate models. Software engineering research, on the other hand, has proposed a variety of paradigms and tools for *testing* complex software systems. In particular, "behavioral testing" (also known as black-box testing) is concerned with testing different capabilities of a system by validating

Inspired by principles of behavioral testing in software engineering

Tools for Behavioral Evaluation, see ZENO, Cabrera et al 2023

Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning

Ángel Alexander Cabrera
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Erica Fu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

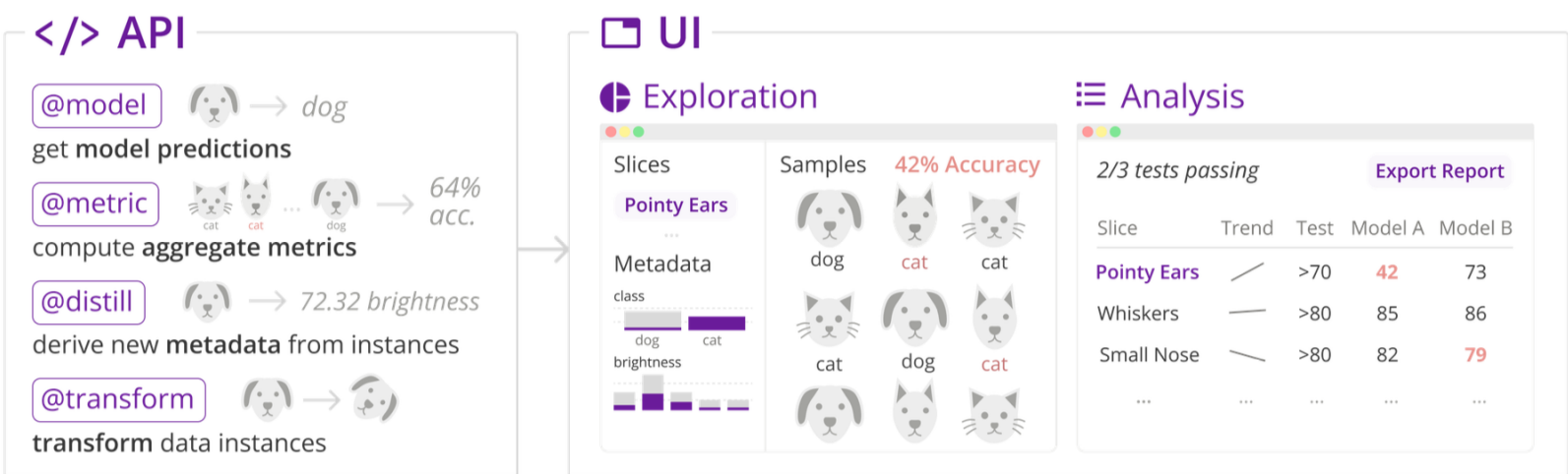
Donald Bertucci
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Kenneth Holstein
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Ameet Talwalkar
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Jason I. Hong
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Adam Perer
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA



zenoml.com

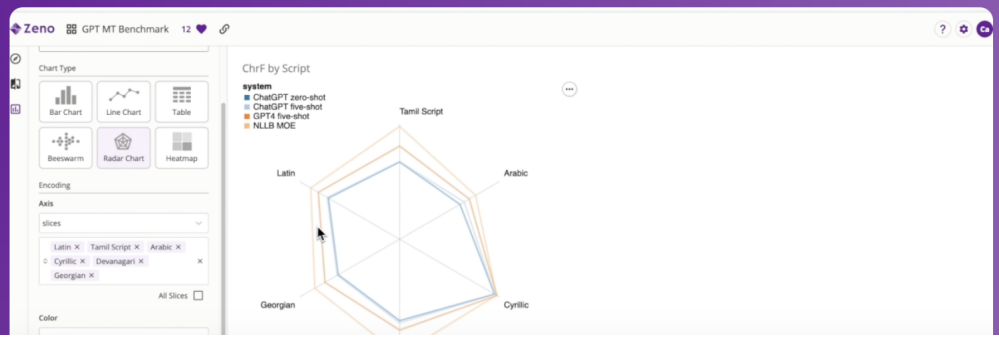
New Chrome available

Zeno Docs Blog About FAQ Sign up Sign In

AI Evaluation Made Easy

Discover how your AI performs with **Zeno**.
Explore your data, uncover failures, and create beautiful, interactive charts.

Get Started Explore Projects



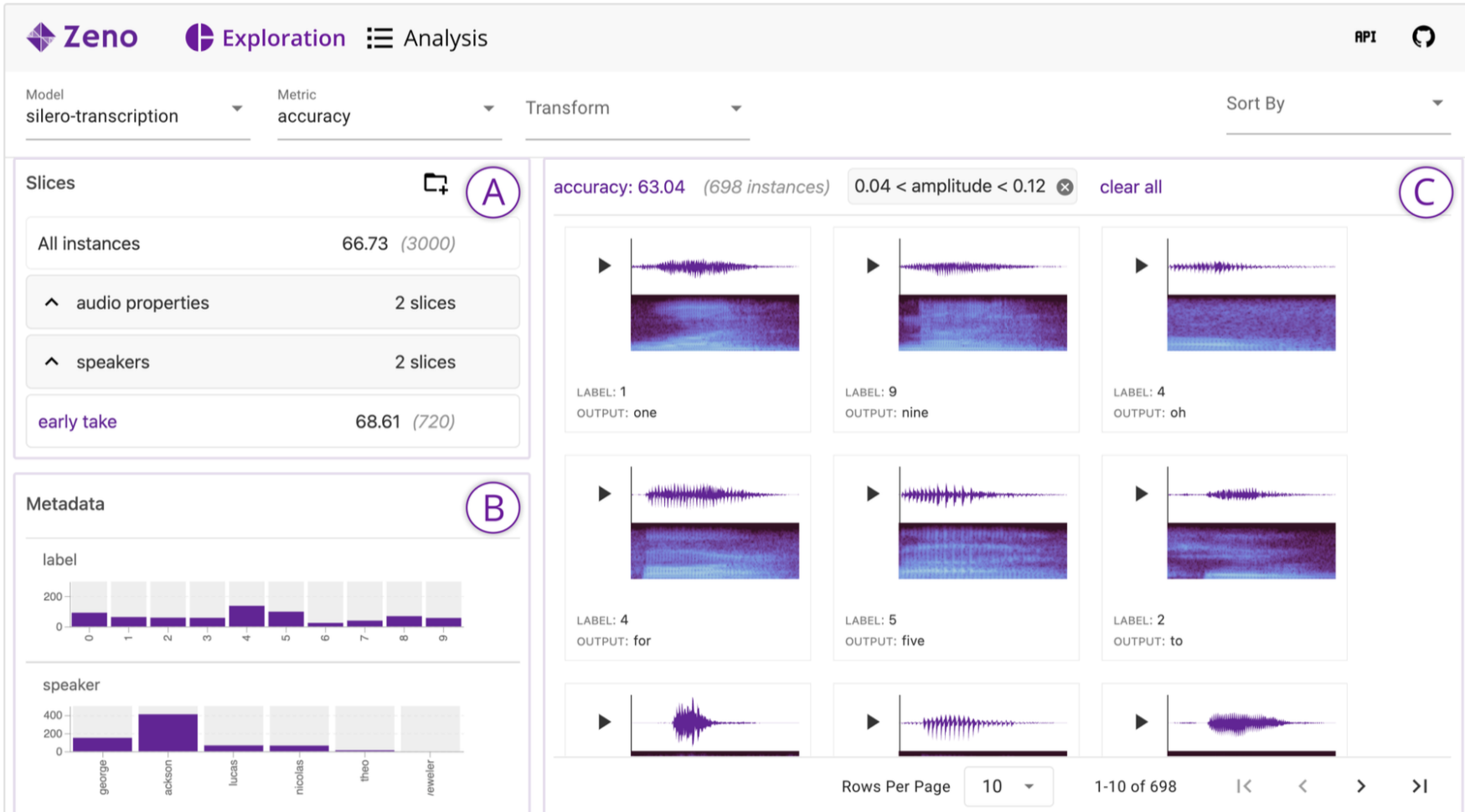


Figure 4: The Exploration UI allows users to see data instances and model outputs and investigate model performance. In the figure, ZENO is shown for the audio transcription example described in Section 5. The interface has two components, the Metadata Panel (A & B) and the Samples View (C). The Metadata Panel shows the metadata distributions of the dataset (B) and the slices and folders a user has created (A). The metadata widgets are cross-filtered, with the purple bars showing the filtered table distribution. The Samples View (C) shows the filtered data instances and outputs, currently those with *0.04 < amplitude < 0.12*, along with the selected metric, in this case, accuracy.

LLMs (e.g., ChatGPT, Clause, Gemini, ...) and Your Final Project

- You are **welcome** to use models like ChatGPT, Claude, or Gemini in your final project.
  - Need your own API access
  - There are many exciting projects that can be done using these models:
    - Analysis projects: Understand model behavior.
    - Prompt engineering: Learn to design effective prompts.
    - Chain of thought reasoning: Explore advanced reasoning with large models.
- **Important reminder:** You will be evaluated based on **your work**, not just the impressive output of ChatGPT or similar models.
  - Showing impressive results (e.g., *"Look, it works zero-shot!"*) is not sufficient if the project lacks your own contribution.

2024 NLP ... recommended for practical projects

pip install transformers # By **Huggingface** 🤗

```
# not quite runnable code but gives the general idea….
from transformers import BertForSequenceClassification, AutoTokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased’)
model.train()
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased’)
fine_tuner = Trainer( model=model,  args=training_args, train_dataset=train_dataset,
    eval_dataset=test_dataset )
fine_tuner.train()
eval_dataset = load_and_cache_examples(args, eval_task, tokenizer, evaluate=True)
results = evaluate(model, tokenizer, eval_dataset, args)
```