Natural Language Processing with Deep Learning

CS224N/Ling284

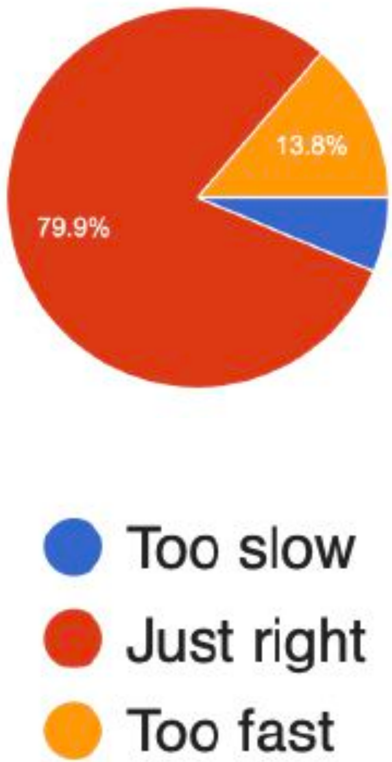
Christopher Manning

Lecture 14: More on Contextual Word Representations and Pretraining

Thanks for your Feedback!

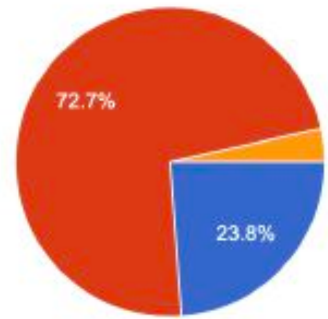
How's the pace of lectures so far?

458 responses



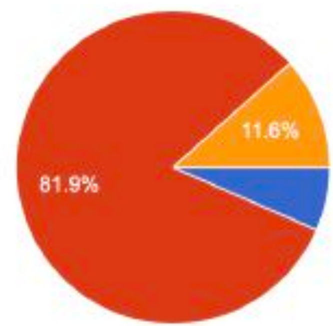
How challenging was Assignment 1?

458 responses



How challenging was Assignment 3?

458 responses



Thanks for your Feedback!

What do you most want to learn about in the remaining lectures?

More cutting-edge research directions
BERT embeddings
a survey about what the different NLP techniques beyond what we've learned
I want to dive further into cutting edge NLP techniques like transformers
transformers, bert, more state-of-the-art models in nlp
BERT, GPT-2 and derivative models
How different techniques/models tackle various linguistic challenges/complexities
Image captioning
GPT-2?
Program synthesis applications from natural language
I think it would be really helpful to understand how to go about building a model from scratch and understanding what techniques to leverage in certain problems.
BERT
I am interested in learning about different contexts these models can be applied to
Guest lecture

Announcements

- Assignment 5 is due today
- We're handing back project proposal feedback today
- Project milestone – due in 12 days…

Lecture Plan

Lecture 14: Contextual Word Representations and Pretraining

1. Reflections on word representations (5 mins)
2. Pre-ELMo and ELMO (20 mins)
3. ULMfit and onward (10 mins)
4. Transformer architectures (20 mins)
5. BERT (15 mins)
6. How’s the weather? (5 mins)

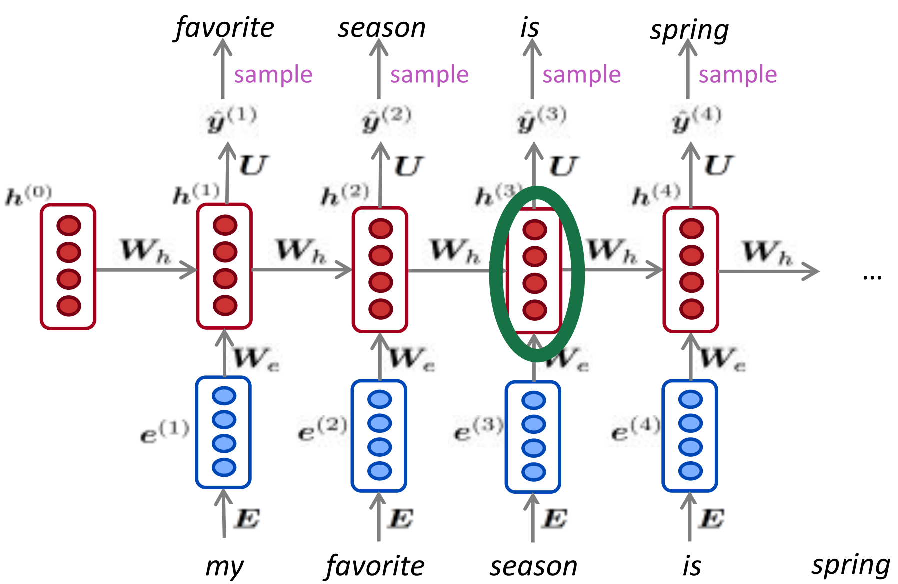
1. Representations for a word

- Originally, we basically had one representation of words:
  - The word vectors that we learned about at the beginning
    - Word2vec, GloVe, fastText

- These have two problems:
  - Always the same representation for a **word type** regardless of the context in which a **word token** occurs
    - We might want very fine-grained word sense disambiguation
  - We just have one representation for a word, but words have different **aspects**, including semantics, syntactic behavior, and register/connotations

Did we all along have a solution to this problem?

- In, an NLM, we immediately stuck word vectors (perhaps only trained on the corpus) through LSTM layers
- Those LSTM layers are trained to predict the next word
- But those language models are producing context-specific word representations at each position!



Context-free vs. contextual similarity

Model	Source	Nearest Neighbor(s)
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
BiLM	Chico Ruiz made a spectacular play on Alusik 's grounder {. . . }	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {. . . }	{. . . } they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

From Peters et al. 2018 Deep contextualized word representations (ELMo paper)

2. Pre-ELMo and ELMo

Dai and Le (2015) https://arxiv.org/abs/1511.01432

- Why don't we do semi-supervised approach where we train NLM sequence model on large unlabeled corpus, rather than just word vectors and use as pretraining for sequence model

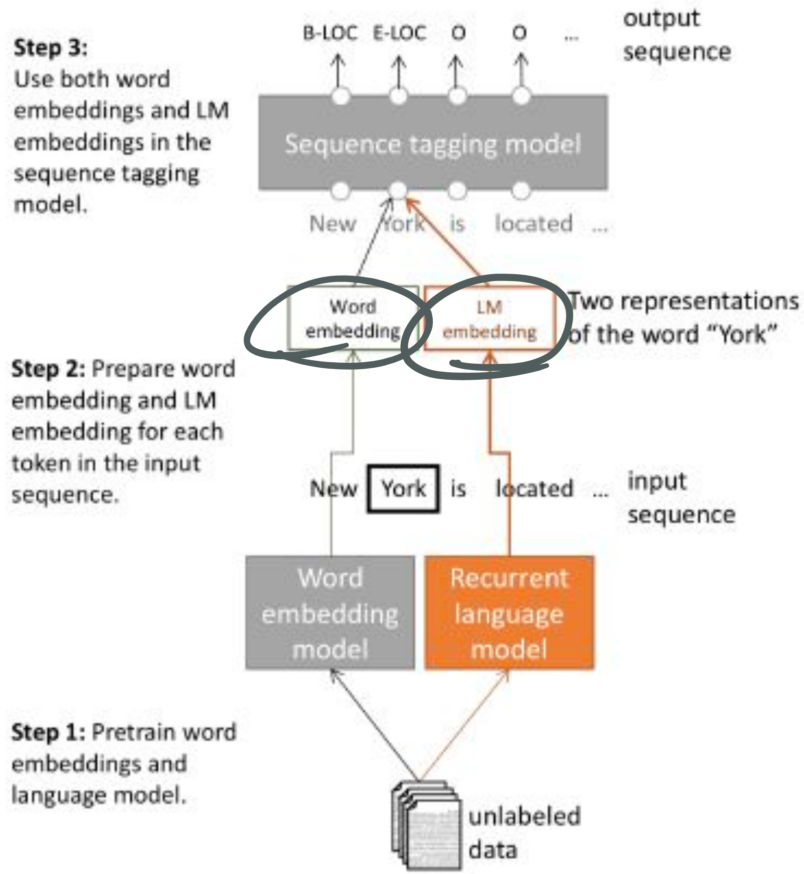
Peters et al. (2017) https://arxiv.org/pdf/1705.00108.pdf

- Idea: Want meaning of word in context, but standardly learn task RNN only on small task-labeled data (e.g., NER)

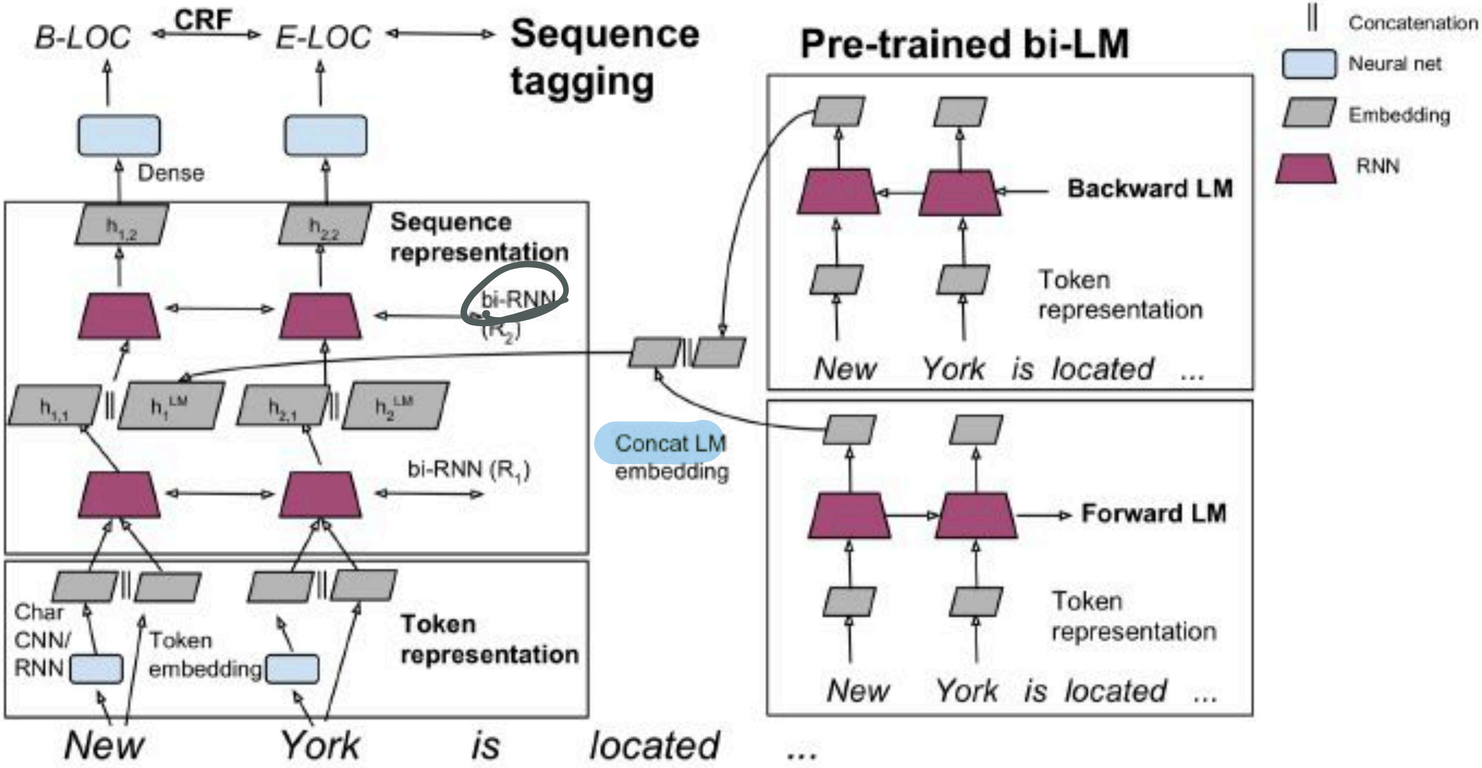
Howard and Ruder (2018) Universal Language Model Fine-tuning for Text Classification. https://arxiv.org/pdf/1801.06146.pdf

- Same general idea of transferring NLM knowledge
- Here applied to text classification

Tag LM (Peters et al. 2017)



Tag LM



$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}].$$

Named Entity Recognition (NER)

- Find and classify names in text, for example:
  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation

CoNLL 2003 Named Entity Recognition (en news testb)

Name	Description	Year	F1
TagLM Peters	LSTM BiLM in BiLSTM tagger	2017	91.93
Ma + Hovy	BiLSTM + char CNN + CRF layer	2016	91.21
Tagger Peters	BiLSTM + char CNN + CRF layer	2017	90.87
Ratinov + Roth	Categorical CRF+Wikipeda+word cls	2009	90.80
Finkel et al.	Categorical feature CRF	2005	86.86
IBM Florian	Linear/softmax/TBL/HMM ensemble, gazettes++	2003	88.76
Stanford Klein	MEMM softmax markov model	2003	86.07

Peters et al. (2017): TagLM – "Pre-ELMo"

Language model is trained on 800 million training words of "Billion word benchmark"

Language model observations

- An LM trained on supervised data does not help
- Having a bidirectional LM helps over only forward, by about 0.2
- Having a huge LM design (ppl 30) helps over a smaller model (ppl 48) by about 0.3

Task-specific BiLSTM observations

- Using just the LM embeddings to predict isn't great: 88.17 F1
  - Well below just using an BiLSTM tagger on labeled data

Peters et al. (2018): ELMo: Embeddings from Language Models

Deep contextualized word representations. NAACL 2018. https://arxiv.org/abs/1802.05365

- Initial breakout version of **word token vectors** or **contextual word vectors**
- Learn word token vectors using long contexts not context windows (here, whole sentence, could be longer)
- Learn a deep Bi-NLM and use all its layers in prediction

Peters et al. (2018): ELMo: Embeddings from Language Models

- Train a bidirectional LM
- Aim at performant but not overly large LM:
  - Use 2 biLSTM layers
  - Use character CNN to build initial word representation (only)
    - 2048 char n-gram filters and 2 highway layers, 512 dim projection
  - Use 4096 dim hidden/cell LSTM states with 512 dim projections to next input
  - Use a residual connection
  - Tie parameters of token input and output (softmax) and tie these between forward and backward LMs

Peters et al. (2018): ELMo: Embeddings from Language Models

- ELMo learns task-specific combo of biLM layer representations
- This is an innovation that improves on just using top layer of LSTM stack

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

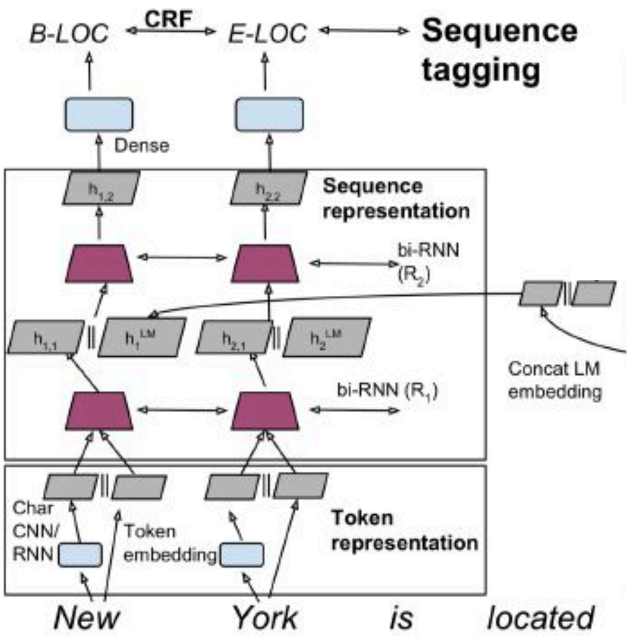
- γ^{task} scales overall usefulness of ELMo to task;
- $\mathbf{s}^{\text{task}}$ are softmax-normalized mixture model weights

Peters et al. (2018): ELMo: Use with a task

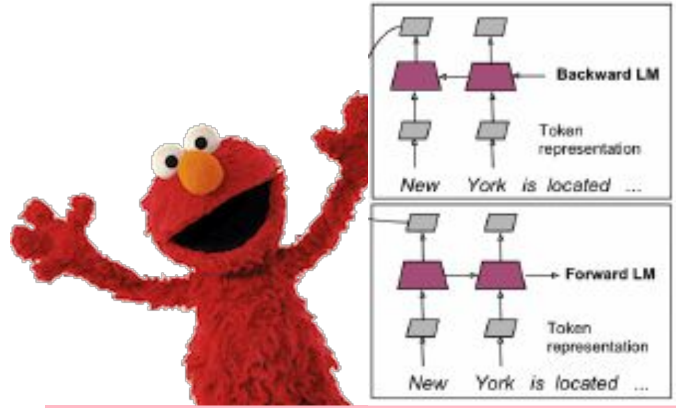
- First run biLM to get representations for each word
- Then let (whatever) end-task model use them
  - Freeze weights of ELMo for purposes of supervised model
  - Concatenate ELMo weights into task-specific model
    - Details depend on task
      - Concatenating into intermediate layer as for TagLM is typical
      - Can provide ELMo representations again when producing outputs, as in a question answering system

ELMo used in an NER tagger

Breakout version of **deep contextual word vectors**

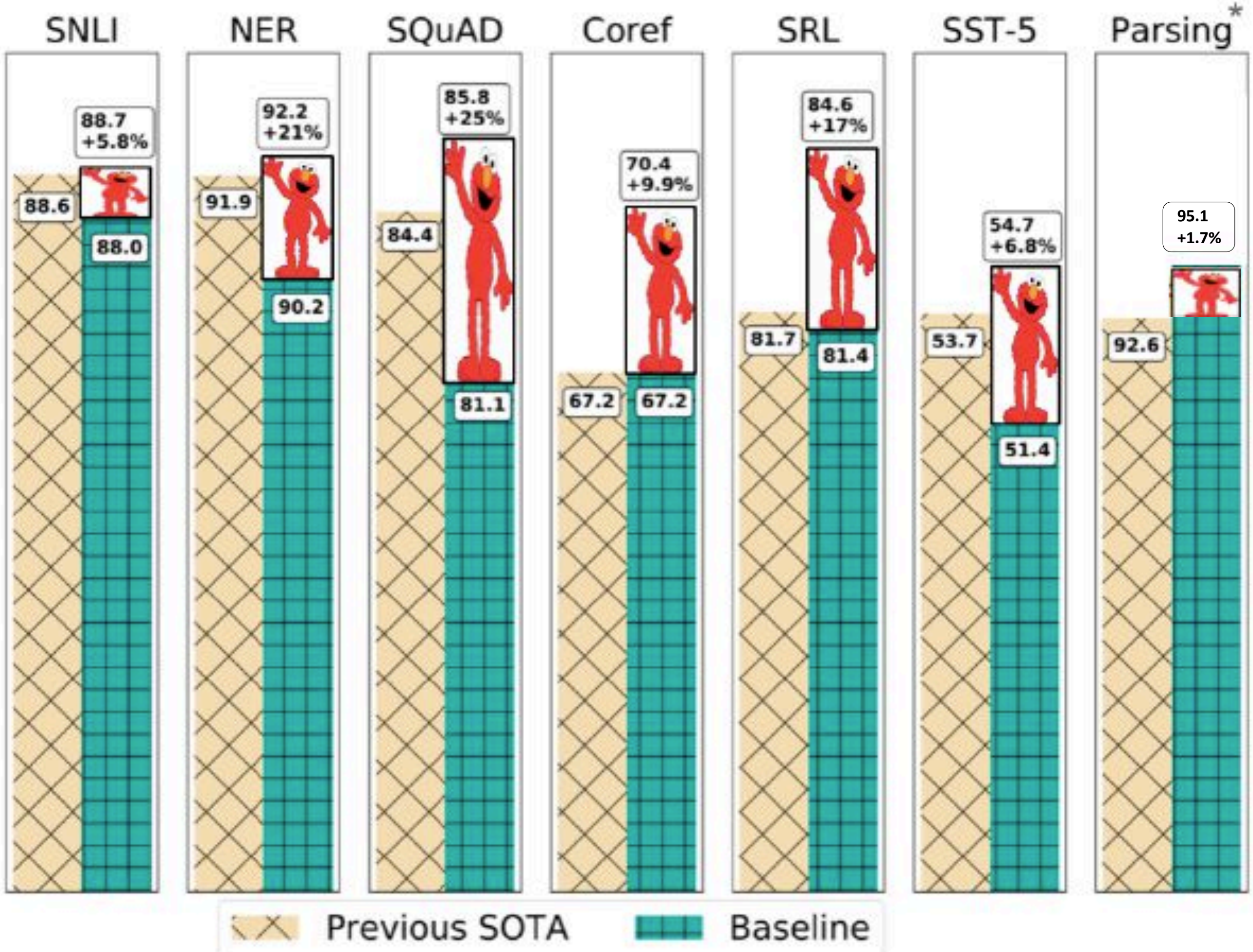


$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]$$



ELMo representation:
A deep bidirectional neural LM

Use learned, task-weighted average of (2) hidden layers



*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)

CoNLL 2003 Named Entity Recognition (en news testb)

Name	Description	Year	F1
ELMo	ELMo in BiLSTM	2018	92.22
TagLM Peters	LSTM BiLM in BiLSTM tagger	2017	91.93
Ma + Hovy	BiLSTM + char CNN + CRF layer	2016	91.21
Tagger Peters	BiLSTM + char CNN + CRF layer	2017	90.87
Ratinov + Roth	Categorical CRF+Wikipeda+word cls	2009	90.80
Finkel et al.	Categorical feature CRF	2005	86.86
IBM Florian	Linear/softmax/TBL/HMM ensemble, gazettes++	2003	88.76
Stanford	MEMM softmax markov model	2003	86.07

ELMo: Weighting of layers

- The two biLSTM NLM layers have differentiated uses/meanings
  - Lower layer is better for lower-level syntax, etc.
    - Part-of-speech tagging, syntactic dependencies, NER
  - Higher layer is better for higher-level semantics
    - Sentiment, Semantic role labeling, question answering, SNLI

- This seems interesting, but it'd seem more interesting to see how it pans out with more than two layers of network

3. Also in the air: McCann et al. 2017: CoVe

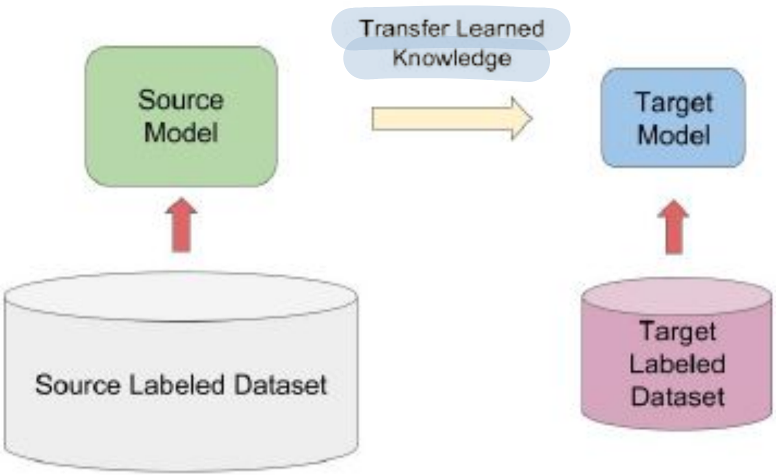
https://arxiv.org/pdf/1708.00107.pdf

- Also has idea of using a trained sequence model to provide context to other NLP models
- Idea: Machine translation is meant to preserve meaning, so maybe that's a good objective?
- Use a 2-layer bi-LSTM that is the encoder of seq2seq + attention NMT system as the context provider
- The resulting CoVe vectors do outperform GloVe vectors on various tasks
- But, the results aren't as strong as the simpler NLM training described in the rest of these slides so seems abandoned
  - Maybe NMT is just harder than language modeling?
  - Maybe someday this idea will return?

Also around: ULMfit

Howard and Ruder (2018) Universal Language Model Fine-tuning for Text Classification. https://arxiv.org/pdf/1801.06146.pdf

- Same general idea of transferring NLM knowledge
- Here applied to text classification

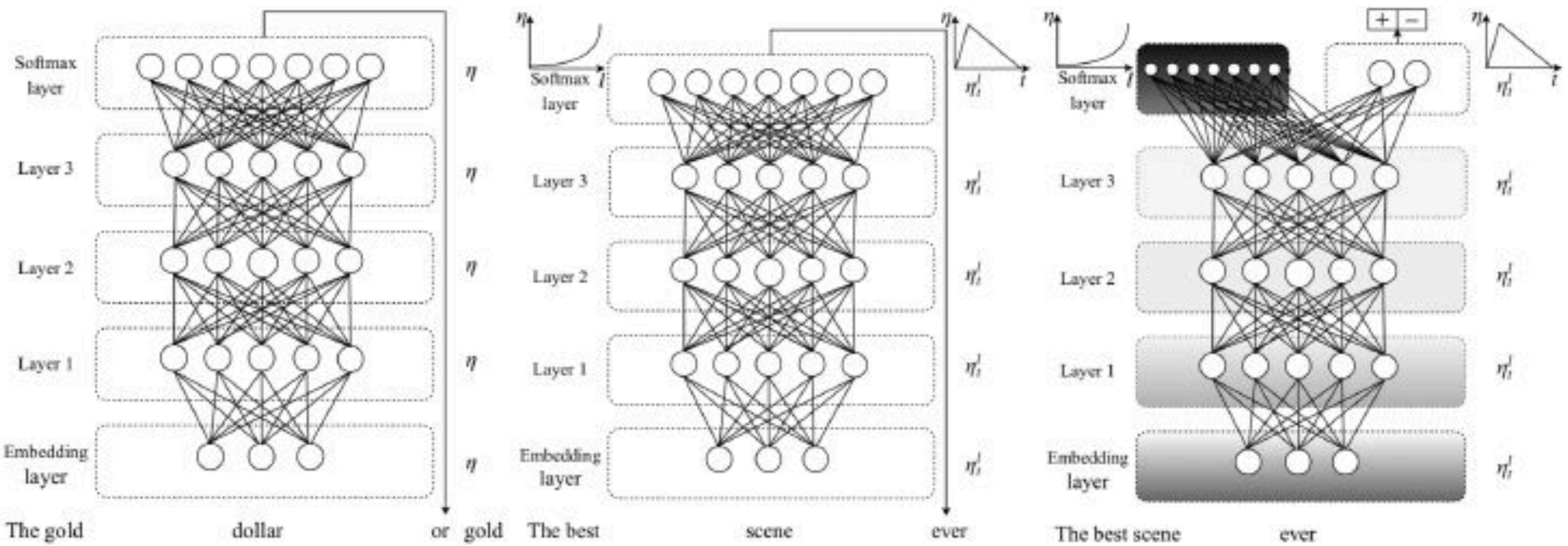


ULMfit

Train LM on big general domain corpus (use biLM)

Tune LM on target task data

Fine-tune as classifier on target task



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

ULMfit emphases

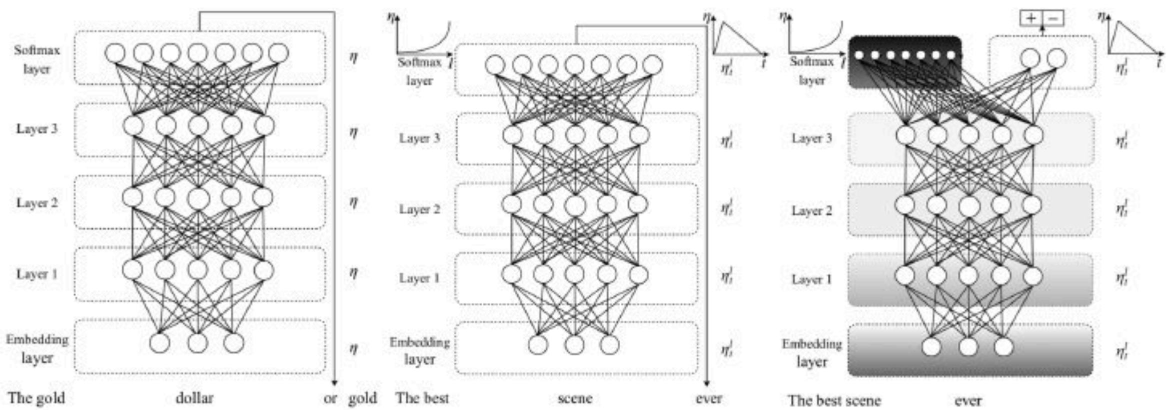
Use reasonable-size "1 GPU" language model not really huge one

A lot of care in LM fine-tuning

- Different per-layer learning rates
- Slanted triangular learning rate (STLR) schedule

Gradual layer unfreezing and STLR when learning classifier

Classify using concatenation $[h_T, \mathrm{maxpool}(\mathbf{h}), \mathrm{meanpool}(\mathbf{h})]$



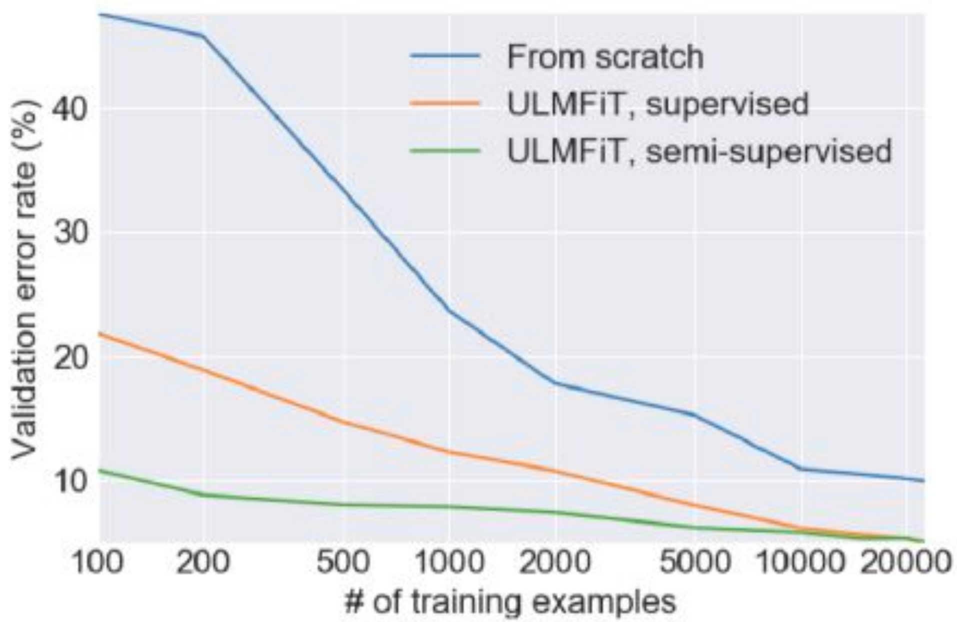
(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

ULMfit performance

- Text classifier error rates

	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**

ULMfit transfer learning



Let's scale it up!

ULMfit

Jan 2018

Training:

1 GPU day

GPT

June 2018

Training

240 GPU days

BERT

Oct 2018

Training

256 TPU days

~320–560 GPU days

GPT-2

Feb 2019

Training

~2048 TPU v3 days according to a reddit thread

fast.ai

OpenAI

Google AI

OpenAI

GPT-2 language model (cherry-picked) output

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. …

METRO NEWS... BUT NOT AS YOU KNOW IT

NEWS SPORT ENTERTAINMENT SOAPS MORE TRENDING

UK WORLD WEIRD TECH

Elon Musk's OpenAI builds artificial intelligence so powerful it must be kept locked up for the good of humanity

Jasper Hamill Friday 15 Feb 2019 10:06 am

272 SHARES

Elon Musk's scientists have announced the creation of a terrifying artificial intelligence that's so smart they refused to release it to the public.

OpenAI's GPT-2 is designed to write just like a human and is an impressive leap forward capable of penning chillingly convincing text.

It was 'trained' by analysing eight million web pages and is capable of writing large tracts based upon a 'prompt' written by a real person.

But the machine mind will not be released in its fully-fledged form because of the risk of it being used for 'malicious purposes' such as generating fake news, impersonating people online, automating the production of spam or churning out 'abusive or faked content to post on social media'.

OpenAI wrote: 'Due to our concerns about malicious applications of the technology, we are not releasing the trained model.

Elon Musk @elonmusk

Follow

Replying to @georgezachary

To clarify, I've not been involved closely with OpenAI for over a year & don't have mgmt or board oversight

8:19 PM - 16 Feb 2019

500 Retweets 14,573 Likes

229 500 15K

4. Transformer models

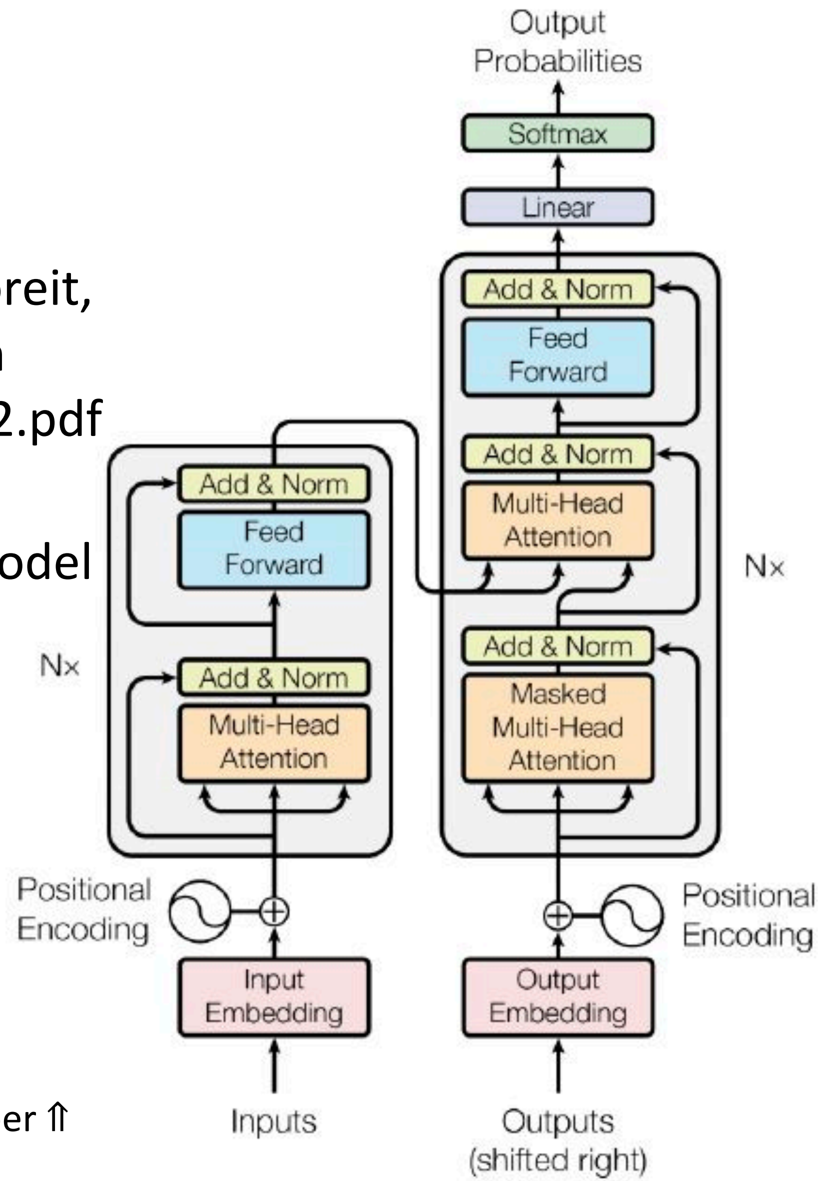
All of these models are Transformer models

ELMo	GPT	BERT	GPT-2	XL-Net, ERNIE, Grover RoBERTa, T5
Oct 2017	June 2018	Oct 2018	Feb 2019	July 2019—
Training: 800M words	Training 800M words	Training 3.3B words	Training 40B words	
42 GPU days	240 GPU days	256 TPU days ~320–560 GPU days	~2048 TPU v3 days according to [a reddit thread]	

The Transformer

Attention is all you need. 2017.
Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin
https://arxiv.org/pdf/1706.03762.pdf

- Non-recurrent sequence-to-sequence encoder-decoder model
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



This and related figures from paper ⇑

Transformer Basics

- Learning about transformers on your own?
  - Key recommended resource:
    - http://nlp.seas.harvard.edu/2018/04/03/attention.html
    - The Annotated Transformer by Sasha Rush
  - A Jupyter Notebook using PyTorch that explains everything!

Dot-Product Attention (Extending our previous def.)

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality d_k value have d_v

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

Dot-Product Attention – Matrix notation

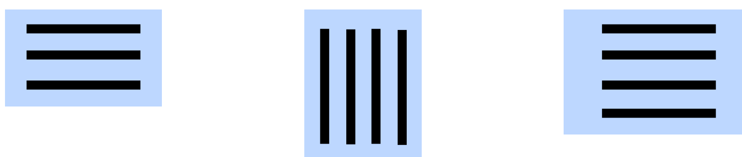
- When we have multiple queries q, we stack them in a matrix Q:

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- Becomes: $A(Q, K, V) = softmax(QK^T)V$

$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$

softmax
row-wise



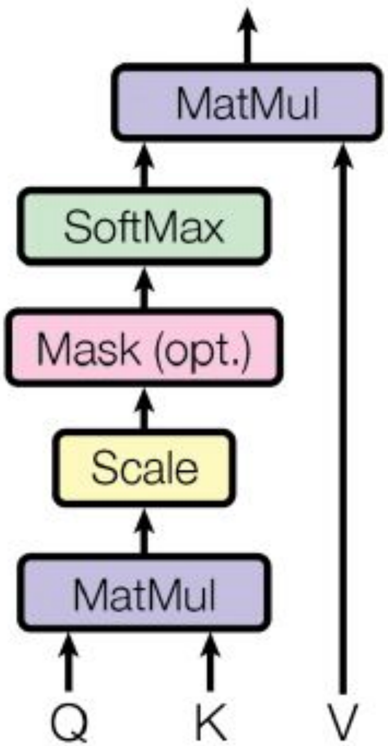
$= [|Q| \times d_v]$

Scaled Dot-Product Attention

- Problem: As d_k gets large, the variance of $q^T k$ increases → some values inside the softmax get large → the softmax gets very peaked → hence its gradient gets smaller.

- Solution: Scale by length of query/key vectors:

$$A(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



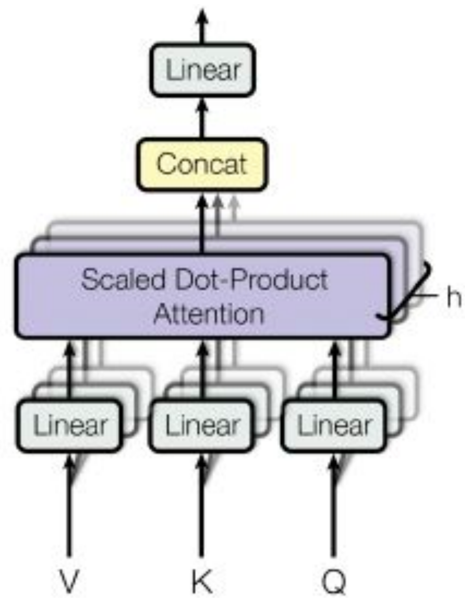
Self-attention in an encoder

- The input word vectors are the queries, keys and values
- In other words: the word vectors themselves select each other
- Word vector stack = Q = K = V
- They’re separated in the definition so you can different things
  - For an NMT decoder, you can do queries from the output with K/V from the encoder

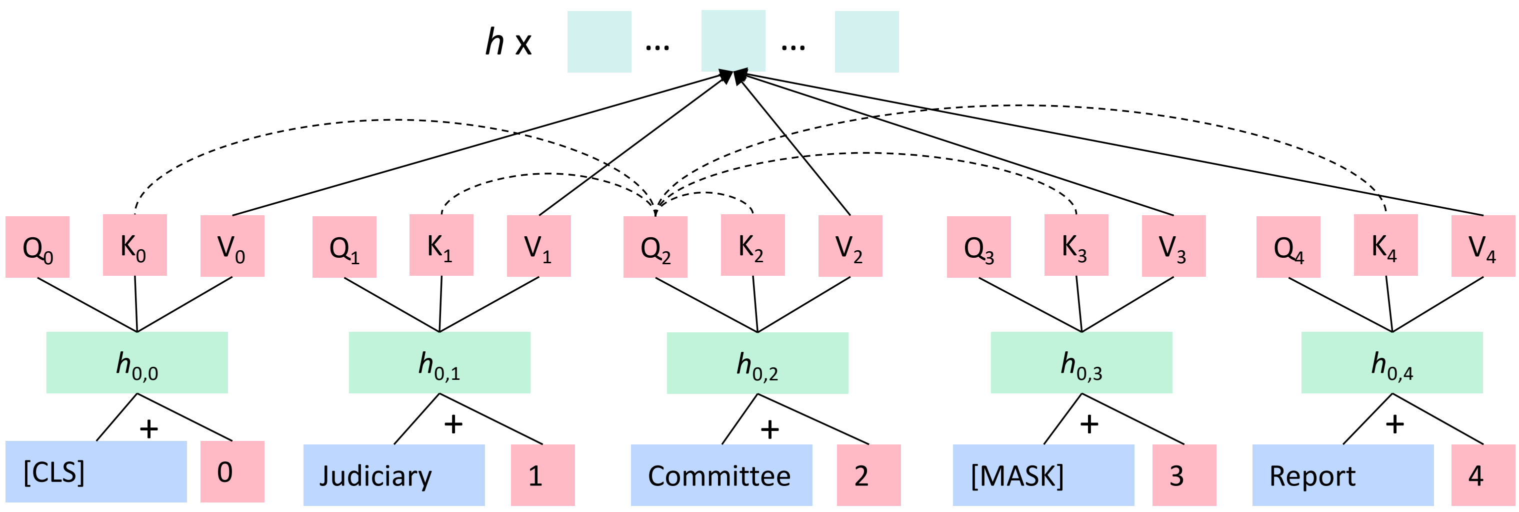
Multi-head attention

- Problem with simple self-attention:
- Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h=8 many lower dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Transformer (Vaswani et al. 2017)



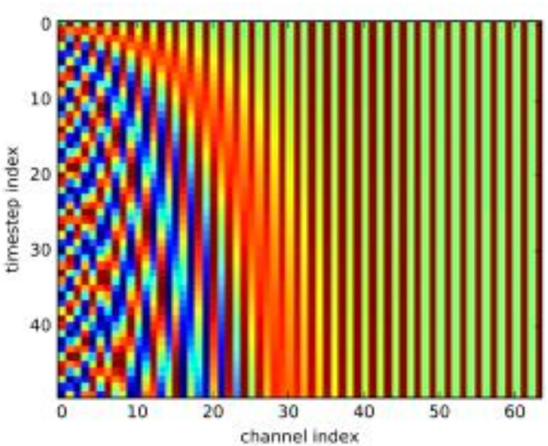
Encoder Input

- Actual word representations are byte-pair encodings
  - As in last lecture

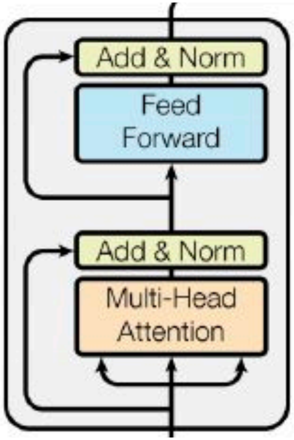
- Also added is a **positional encoding** so same words at different locations have different overall representations:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

or learned



Complete transformer block



Each block has two "sublayers"

1. Multihead attention
2. 2-layer feed-forward NNet (with ReLU)

Each of these two steps also has:

Residual (short-circuit) connection and LayerNorm

LayerNorm(x + Sublayer(x))

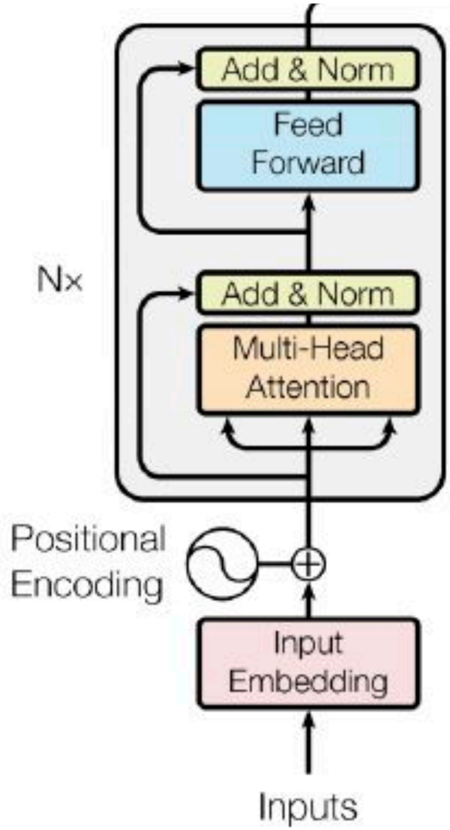
Layernorm changes input features to have mean 0 and variance 1 per layer (and adds two more parameters)

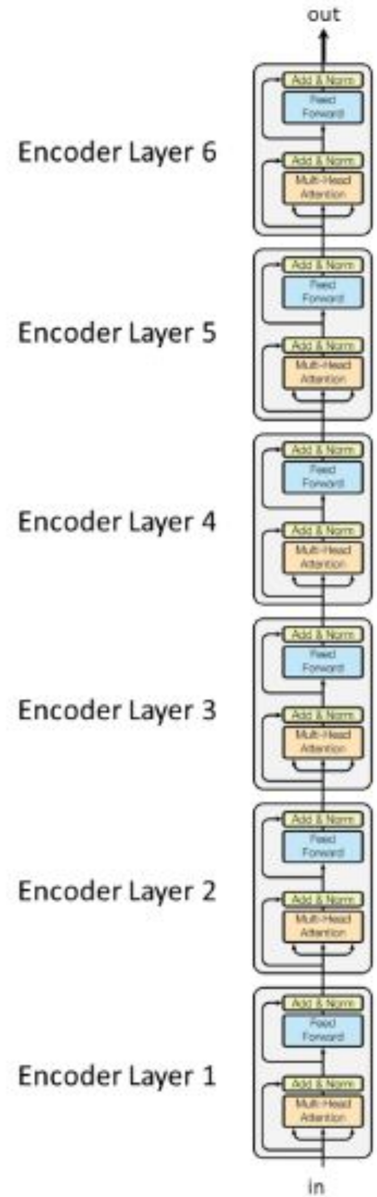
$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

Layer Normalization by Ba, Kiros and Hinton, https://arxiv.org/pdf/1607.06450.pdf

Complete Encoder

- Blocks are repeated 6 or more times
  - (in vertical stack)



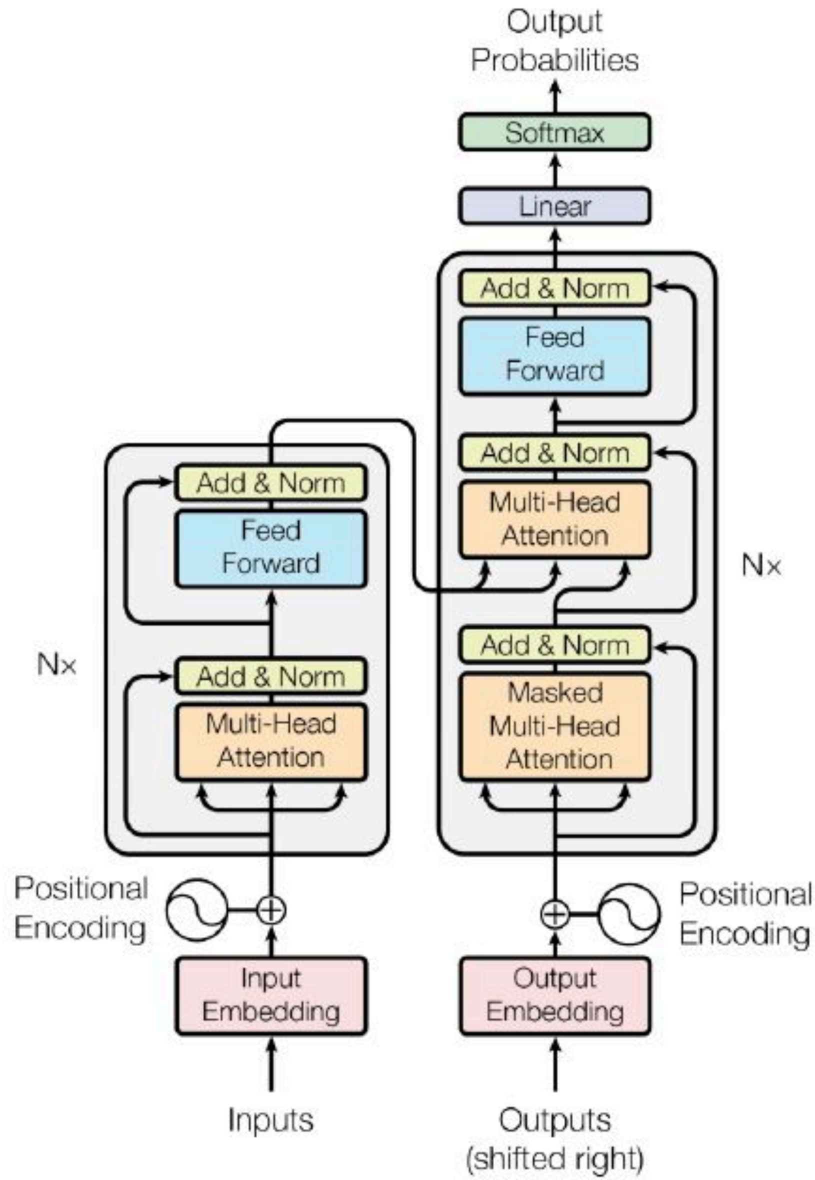


Transformer Decoder

- 2 sublayer changes in decoder
- Masked decoder self-attention on previously generated outputs:

- Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder

- Blocks repeated 6 times also



Experimental Results for MT

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

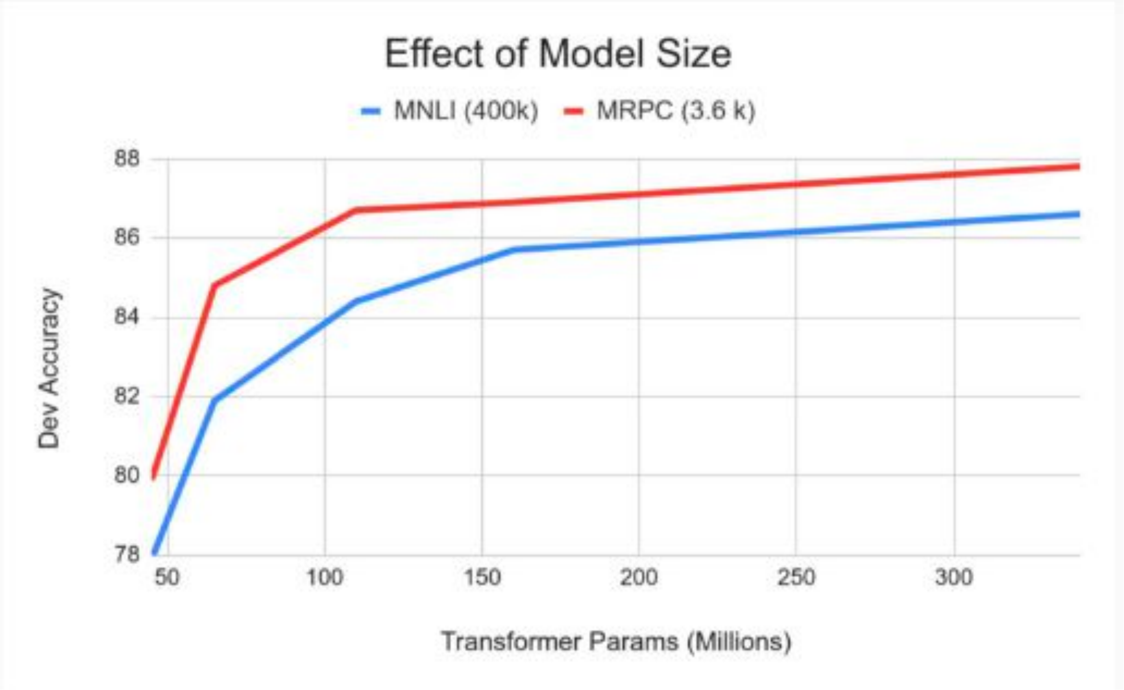
Some performance numbers: LM on WikiText-103

Model	# Params	Perplexity
Grave et al. (2016) – LSTM		48.7
Grave et al. (2016) – LSTM with cache		40.8
4-layer QRNN (Merity et al. 2018)	151M	33.0
LSTM + Hebbian + Cache + MbPA (Rae et al.)	151M	29.2
Transformer-XL Large (Dai et al. 2019)	**257M**	**18.3**
GPT-2 Large* (Radford et al. 2019)	1.5B	17.5

(For gray haired people)
A perplexity of 18 for Wikipedia text is just stunningly low!

Size matters

- Going from 110M to 340M parameters helps a lot
- Improvements have not yet asymptoted



5. BERT: Devlin, Chang, Lee, Toutanova (2018)

BERT (Bidirectional Encoder Representations from Transformers):

Pre-training of Deep Bidirectional Transformers for Language Understanding, which is then fine-tuned for a task

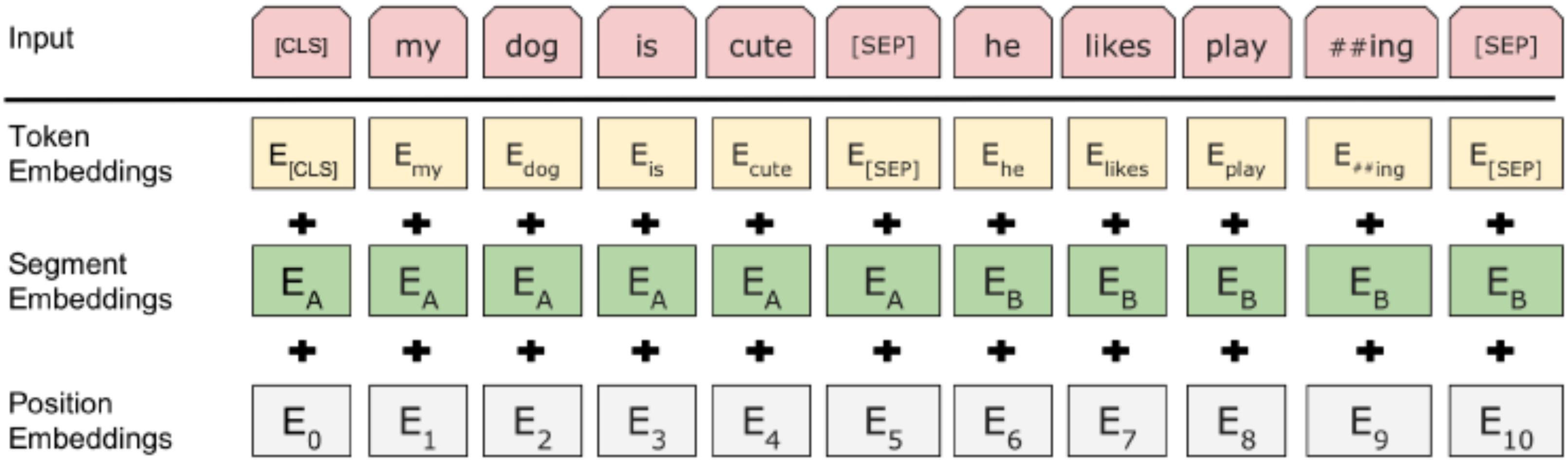
Want: truly bidirectional information flow without leakage in a deep model

Solution: Use a cloze task formulation where 15% of words are blanked out and predicted:

store ↑ gallon ↑

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

BERT sentence pair encoding



Token embeddings are word pieces
Learned segmented embedding represents each sentence
Positional embedding is as for other Transformer architectures

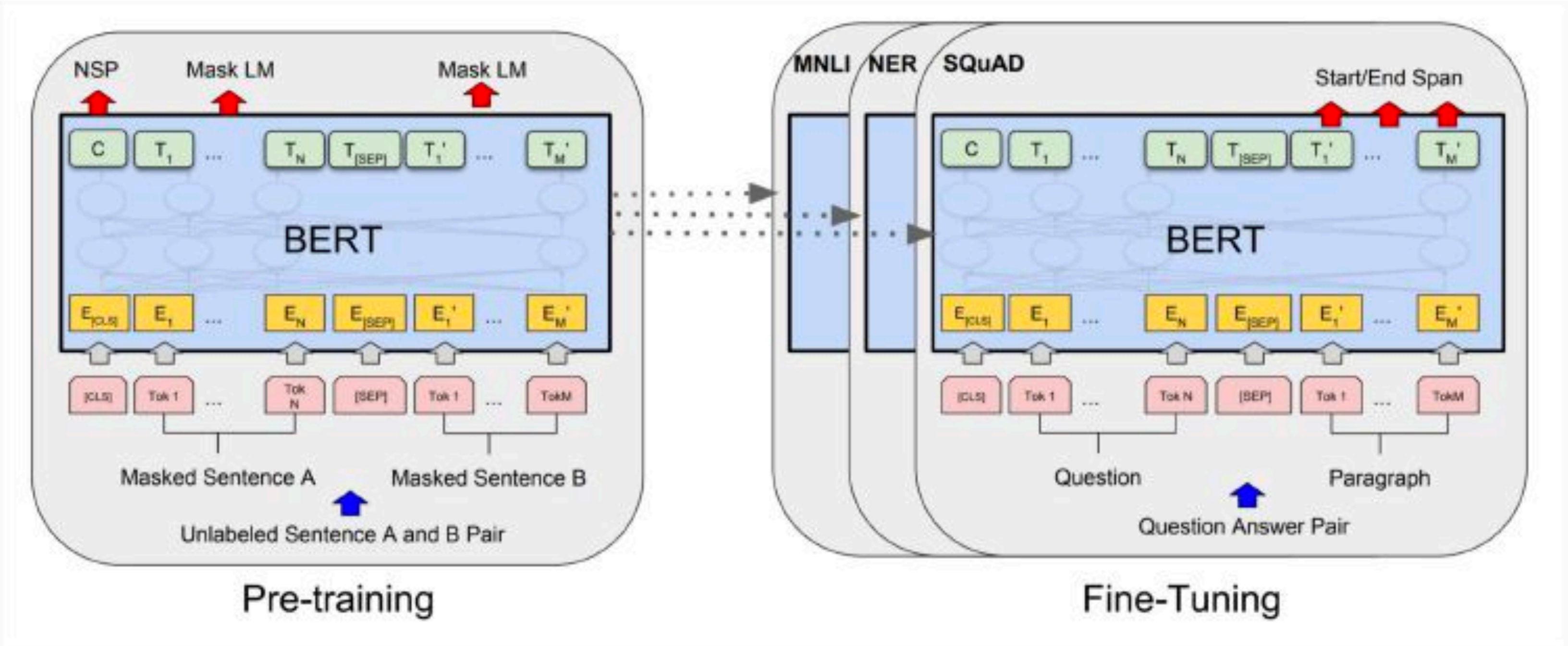
BERT model architecture and training

- Transformer encoder (as before)
- Self-attention ⇒ no locality bias
  - Long-distance context has “equal opportunity”
- Single multiplication per layer ⇒ efficiency on GPU/TPU

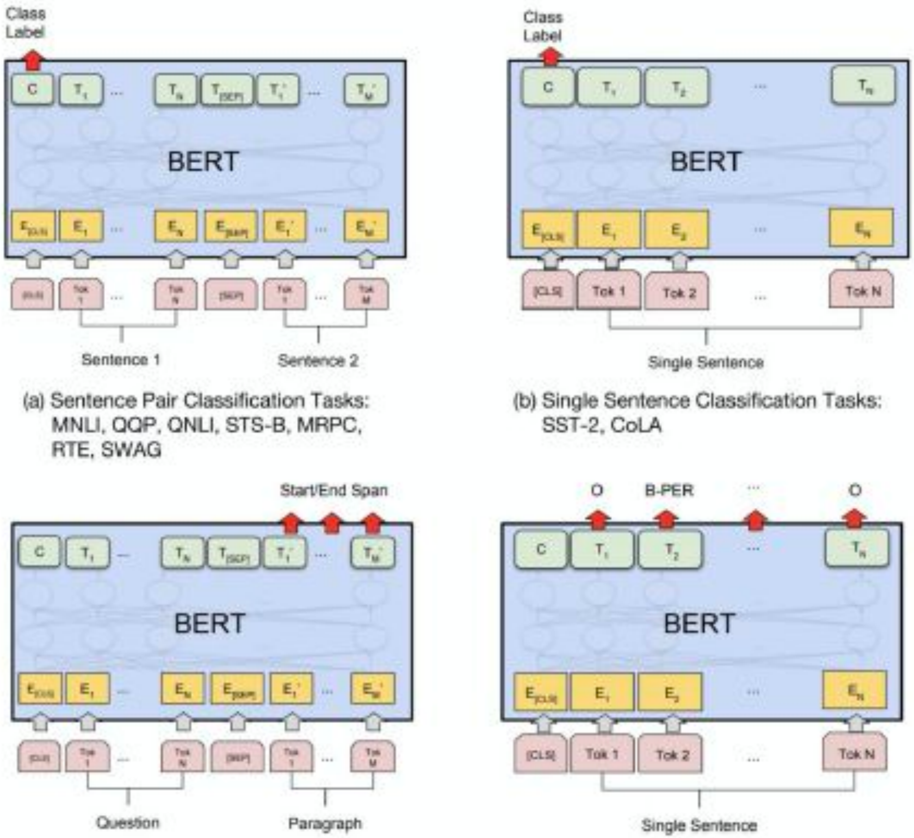
- Train on Wikipedia + BookCorpus
- Train 2 model sizes:
  - BERT-Base: 12-layer, 768-hidden, 12-head
  - BERT-Large: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

BERT model fine tuning

- Simply learn a classifier built on the top layer for each task that you fine tune for



BERT model fine tuning



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

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

(c) Question Answering Tasks: SQuAD v1.1

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

CoNLL 2003 Named Entity Recognition (en news testb)

Name	Description	Year	F1
Flair (Zalando)	Character-level language model	2018	93.09
BERT Large	Transformer bidi LM + fine tune	2018	92.8
CVT Clark	Cross-view training + multitask learn	2018	92.61
BERT Base	Transformer bidi LM + fine tune	2018	92.4
ELMo	ELMo in BiLSTM	2018	92.22
TagLM Peters	LSTM BiLM in BiLSTM tagger	2017	91.93
Ma + Hovy	BiLSTM + char CNN + CRF layer	2016	91.21
Tagger Peters	BiLSTM + char CNN + CRF layer	2017	90.87
Ratinov + Roth	Categorical CRF+Wikipeda+word cls	2009	90.80
Finkel et al.	Categorical feature CRF	2005	86.86
IBM Florian	Linear/softmax/TBL/HMM ensemble, gazettes++	2003	88.76
Stanford	MEMM softmax markov model	2003	86.07

AllenAI ARISTO: Answering Science Exam Questions

From 'F' to 'A' on the N.Y. Regents Science Exams: An Overview of the Aristo Project. Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, Michael Schmitz

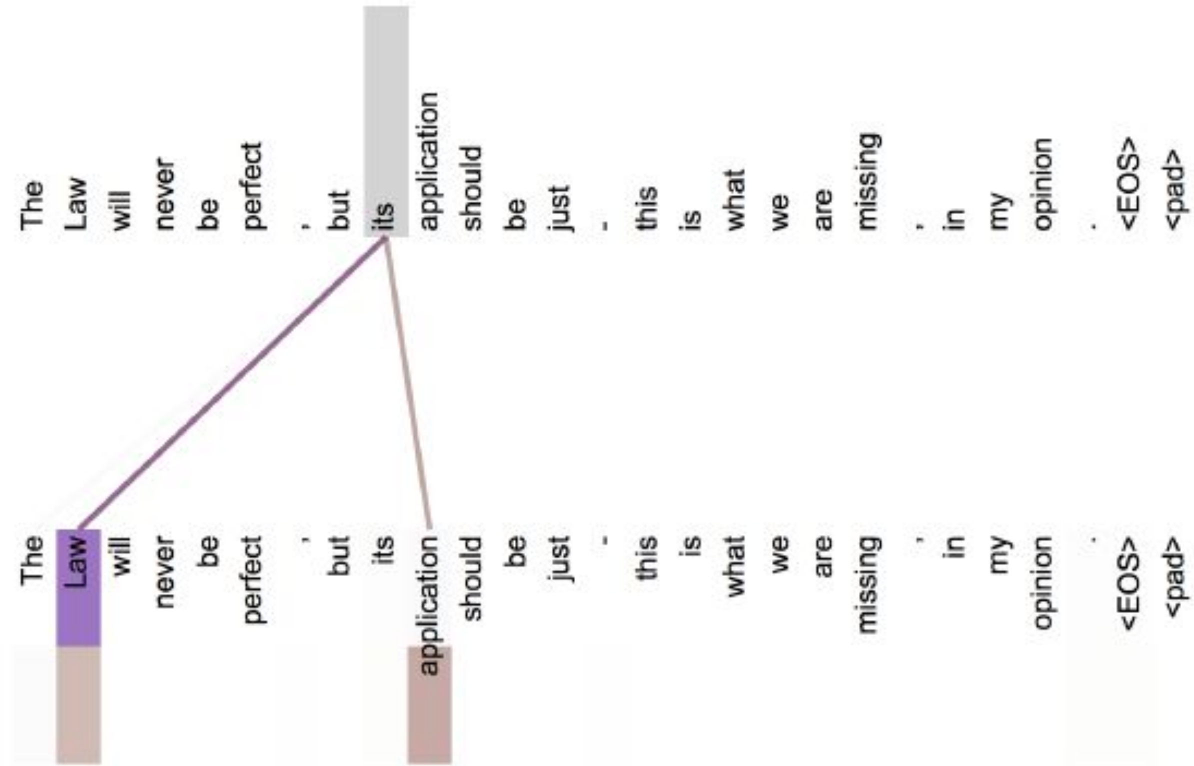
Which equipment will best separate a mixture of iron filings and black pepper? **(1)** magnet **(2)** filter paper **(3)** triplebeam balance **(4)** voltmeter

Which process in an apple tree primarily results from cell division? **(1)** growth **(2)** photosynthesis **(3)** gas exchange **(4)** waste removal

Test Set	IR	TupInf	Multee	AristoBERT	AristoRoBERTa	ARISTO
Regents 4th	64.5	63.5	69.7	86.2	88.1	**89.9**
Regents 8th	66.6	61.4	68.9	86.6	88.2	**91.6**
Regents 12th	41.2	35.4	56.0	75.5	82.3	**83.5**
ARC-Challenge	0.0	23.7	37.4	57.6	**64.6**	64.3

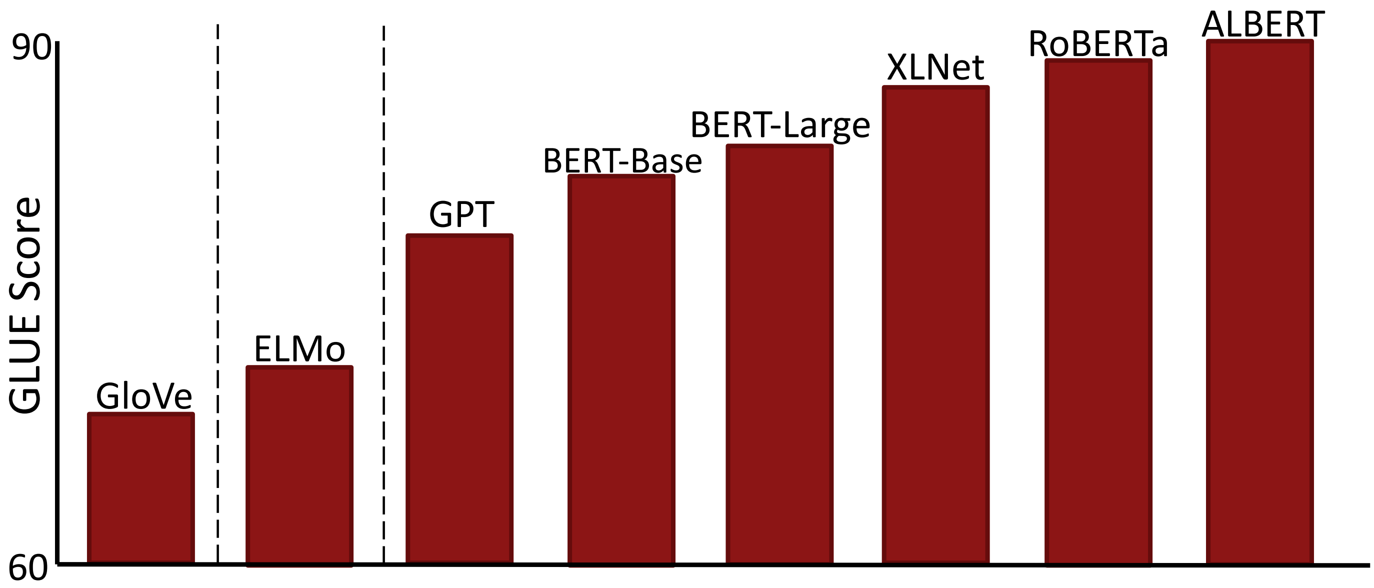
Attention visualization: Implicit anaphora resolution

Words start to pay attention to other words in sensible ways



In 5th layer. Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

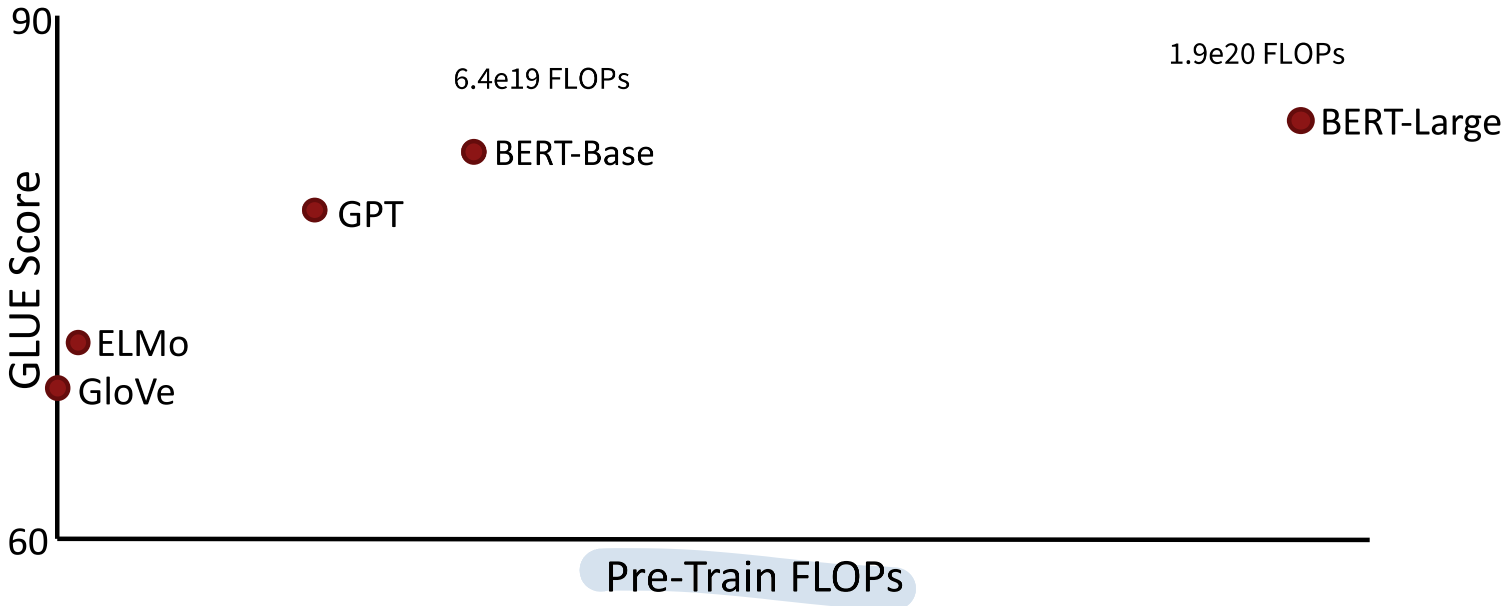
6. How's the weather?
Rapid Progress from Pre-Training (GLUE benchmark)



Over 3x reduction in error in 2 years, "superhuman" performance

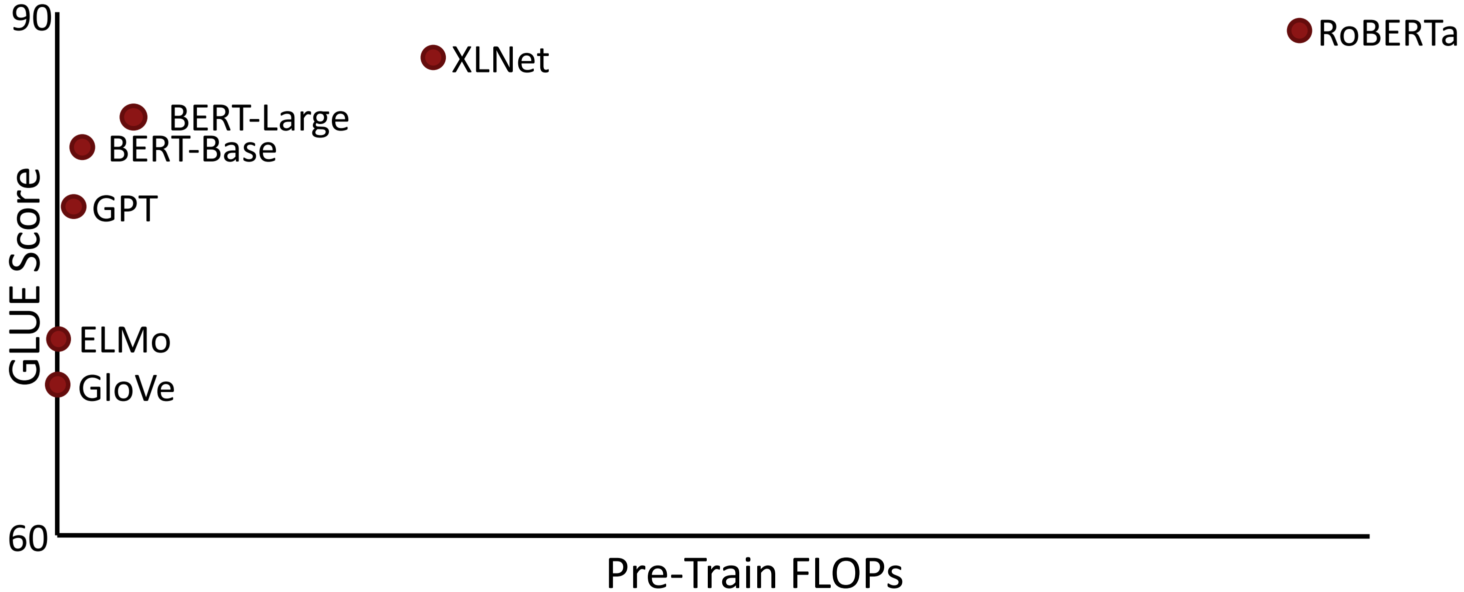
Yay! We now have strongly performing, deep, generic, pre-trained, neural network stacks for NLP that you can just load – in the same way vision has had for 5 years (ResNet, etc.)!

But let's change the x-axis to compute …



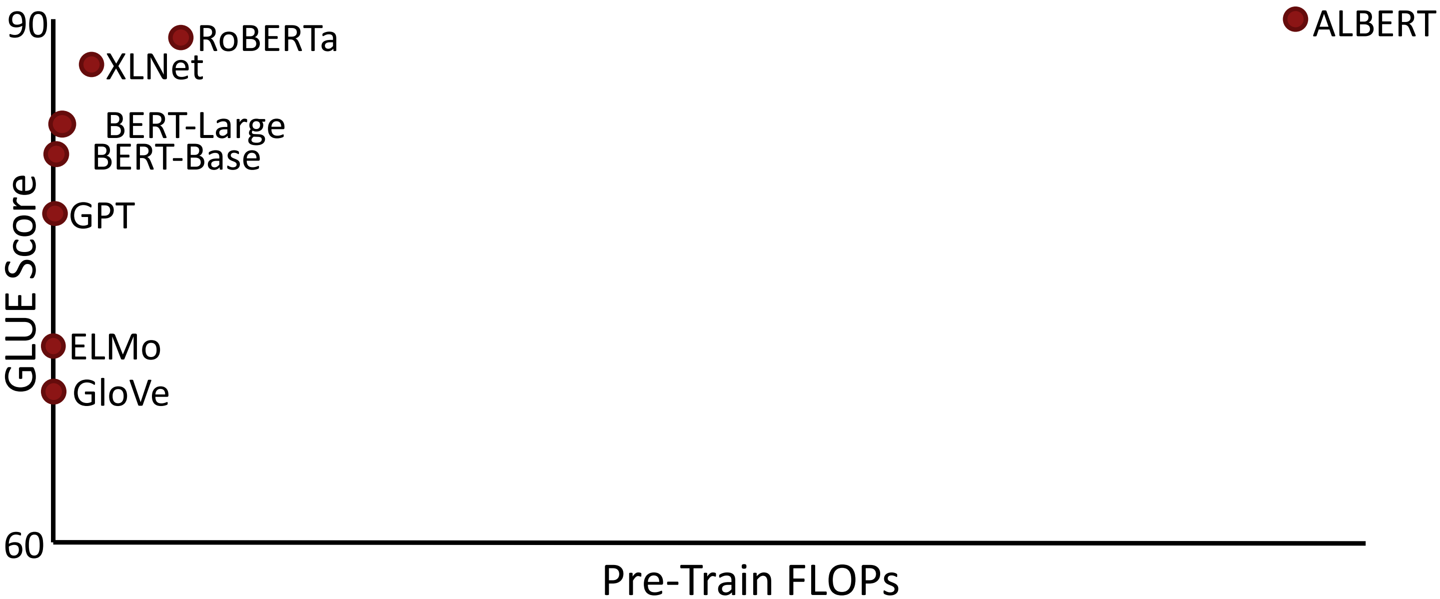
BERT-Large uses 60x more compute than ELMo

But let's change the x-axis to compute …



RoBERTa uses 16x more compute than BERT-Large

More compute, more better?



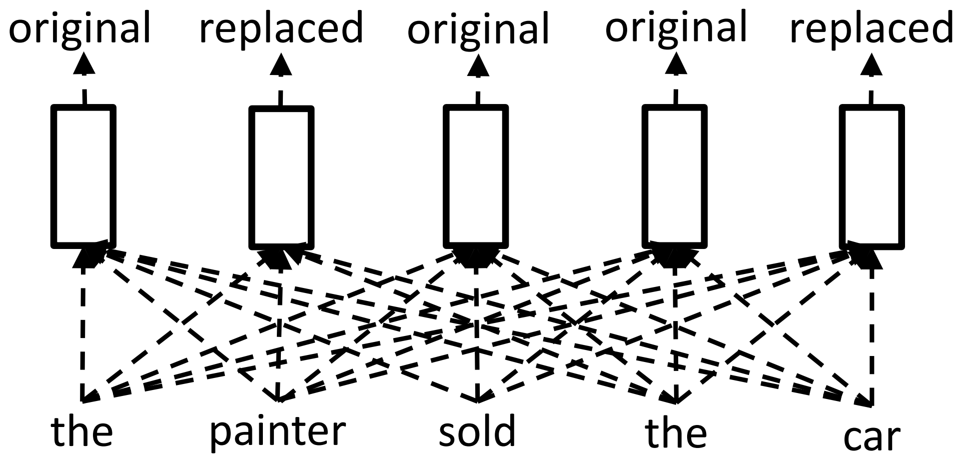
ALBERT uses 10x more compute than RoBERTa

The climate cost of modern deep learning

ELECTRA: "Efficiently Learning an Encoder to Classify Token Replacements Accurately"

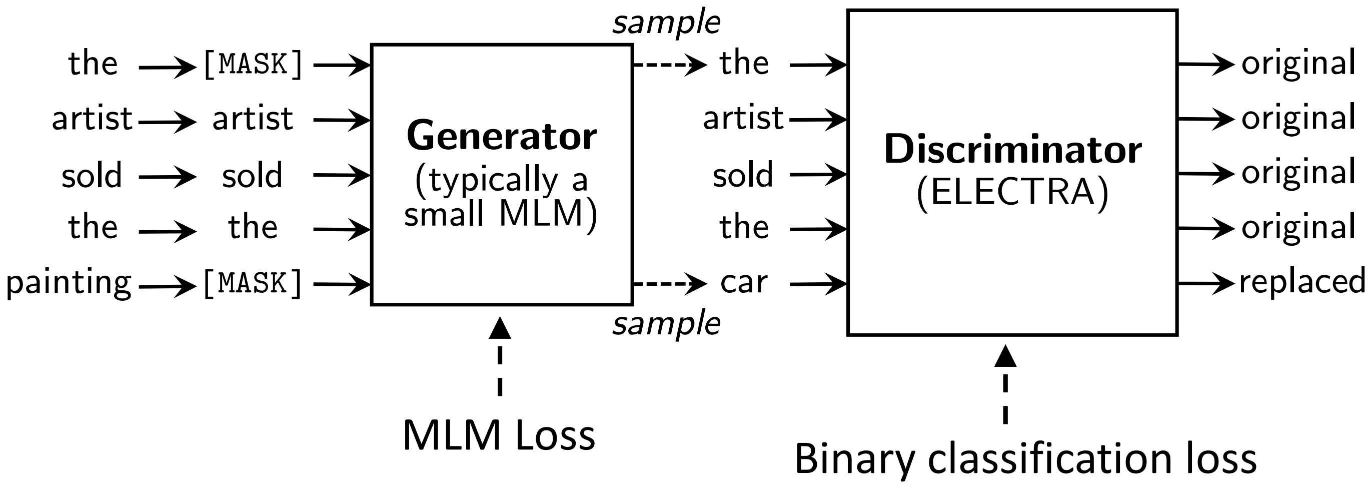
Clark, Luong, Le, and Manning (2020)

Bidirectional model but learn from all tokens



Generating Replacements

Plausible alternatives come from small masked language model (the "generator") trained jointly with ELECTRA



Results: GLUE Score vs Compute