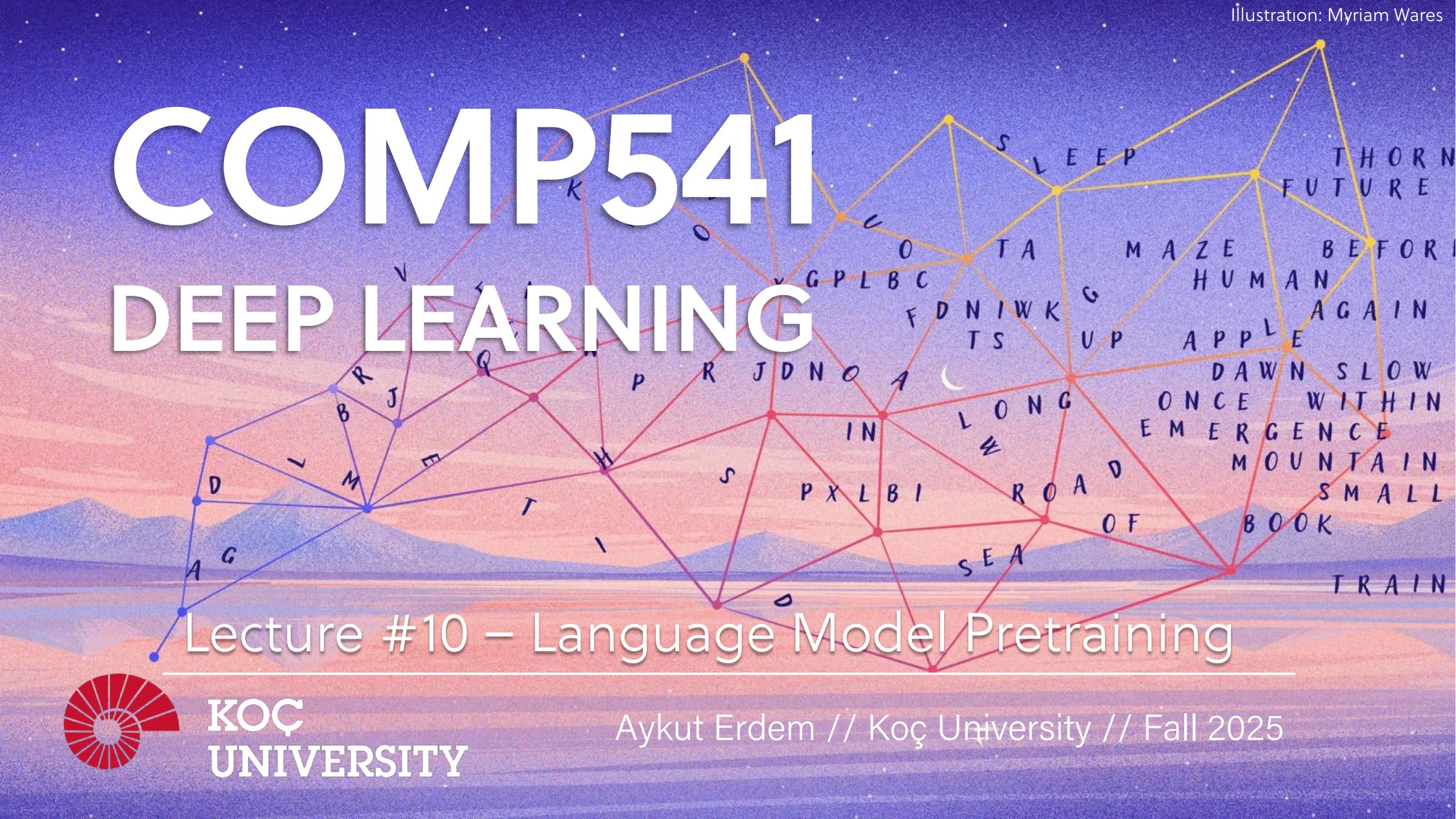
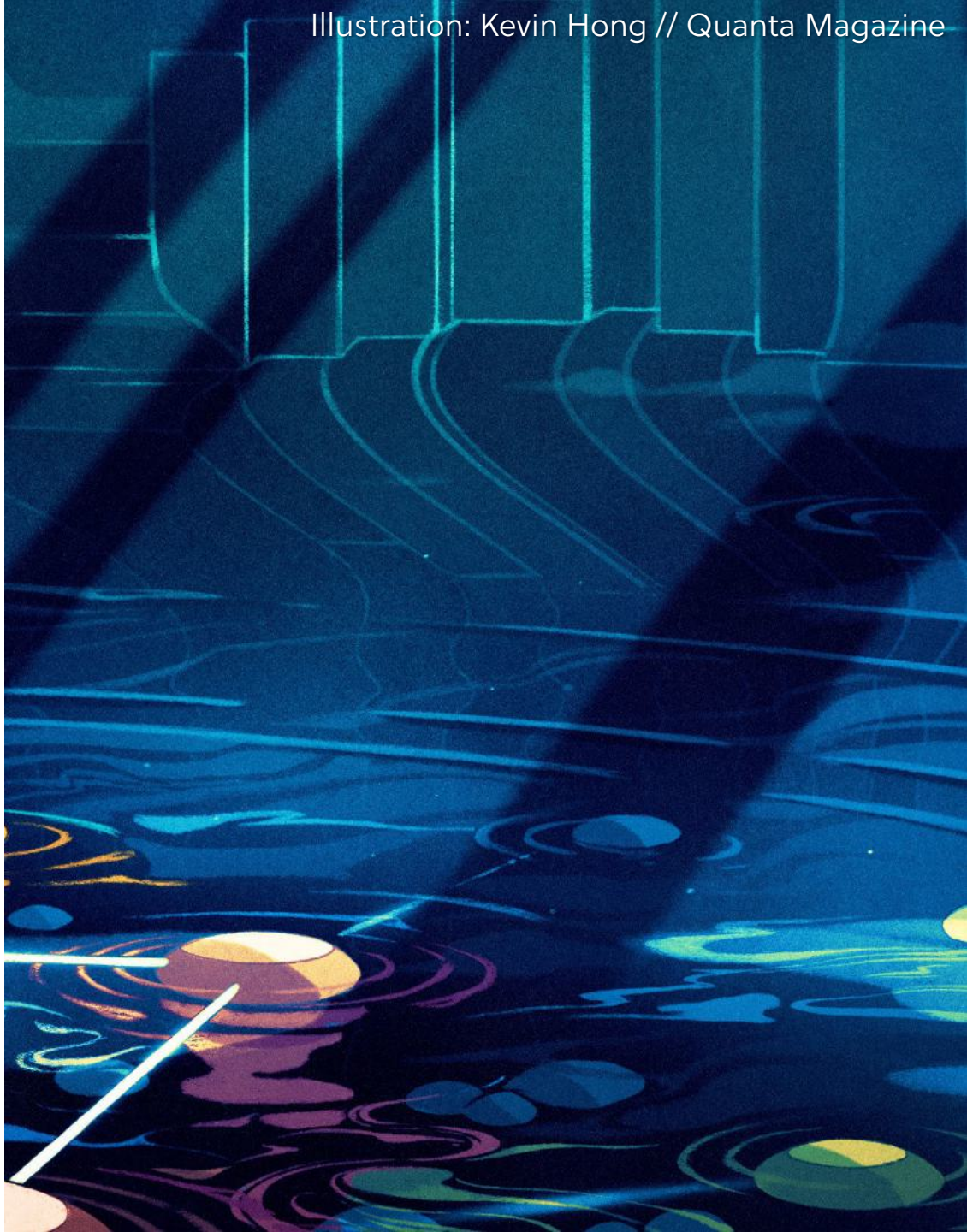


Illustration: Myriam Wares
COMP541
DEEP LEARNING
Lecture #10 – Language Model Pretraining
KOÇ UNIVERSITY
Aykut Erdem // Koç University // Fall 2025

Previously on COMP541

- graph structured data
- graph neural nets (GNNs)
- GNNs for ”classical” network problems



Lecture overview

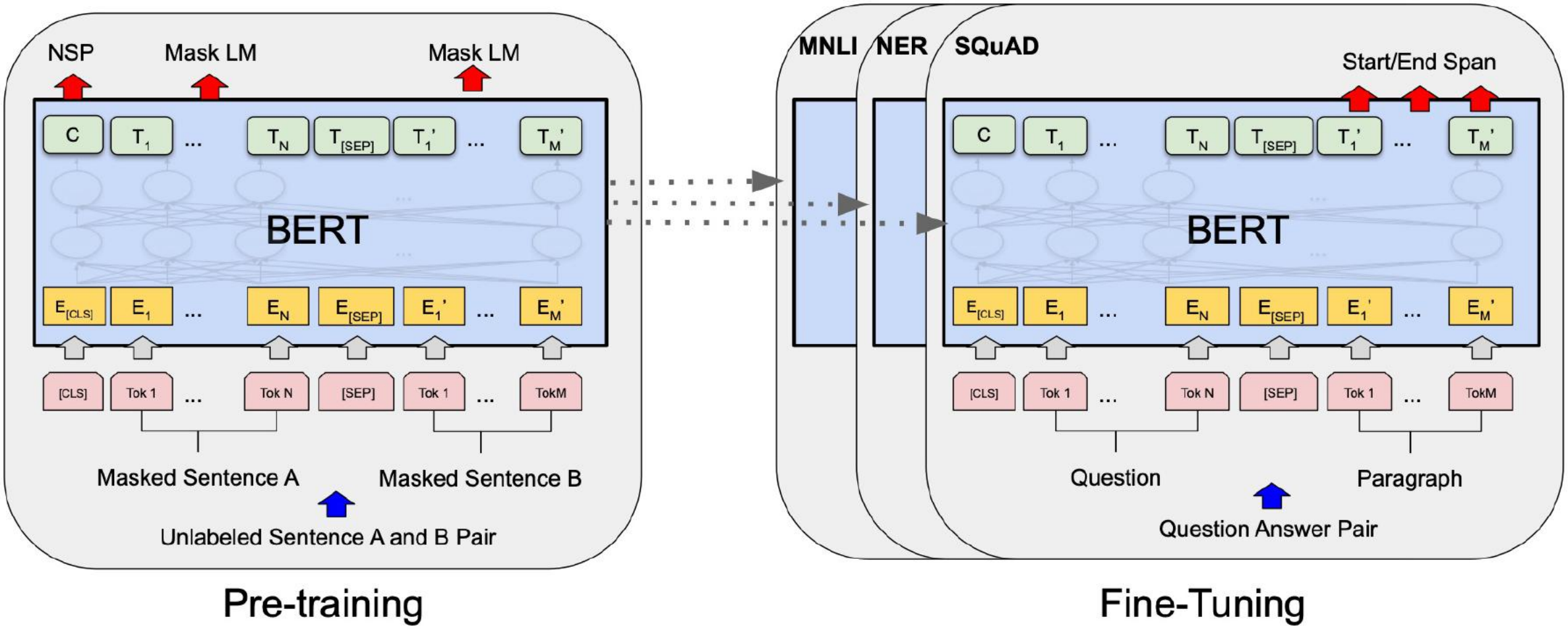
- motivation and introduction
- introduction to language models
- history of neural language models
- pretrained language models

Disclaimer: Much of the material and slides for this lecture were borrowed from
—Alec Radford's lecture on "Learning from Text: Language Models and More"
—Jimmy Ba's UToronto CSC413/2516 class
—Luke Zettlemoyer's UW CSEP517 lecture on Contextualized Word Embeddings
—Liwei Jiang's UW CSE517 lecture on Pre-training

Lecture overview

- **motivation and introduction**
- introduction to language models
- history of neural language models
- pretrained language models

Why is it called pre-training?



- "Pre-"training happens before training (fine-tuning)!

slide credit: Xiang Yue

Why Pre-training

- Standard supervised learning requires "machine learning grade" data
- There is **not** a lot of "machine learning grade" data (compared to what current models need)
- This lecture focuses on a variety of methods for learning from natural language in order to improve the performance of models on standard NLP datasets/tasks.

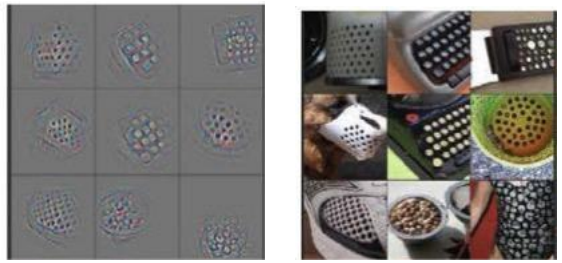
Predecessor of LLMs

Why Pre-training

Computer Vision

Goal: Developing the ability to extract transferable and informative features from images

Models:
- VGG
- ResNet
- Inception
- MobileNet



Visualizing and Understanding Convolutional Network, 2014

Language Modeling

Goal: A good language model should produce good general-purpose and transferable representations from text

Models:
- ELMo
- BERT
- ALBERT
- RoBERTa
- ELECTRA
- ERNIE
- UniLM

Linguistic knowledge:
- The bicycles, even though old, were in good shape because ____ …
- The bicycle, even though old, was in good shape because ____ …

World knowledge:
- The University of Waterloo was founded in _____
- Ontario had a huge population boom as a launching point for expeditions to _____

Recall: Problems working with word-word co-occurrence matrix

- It's still huge!

```
1 million words x 1 million words x 4 byte int32 = 4 terabytes
```

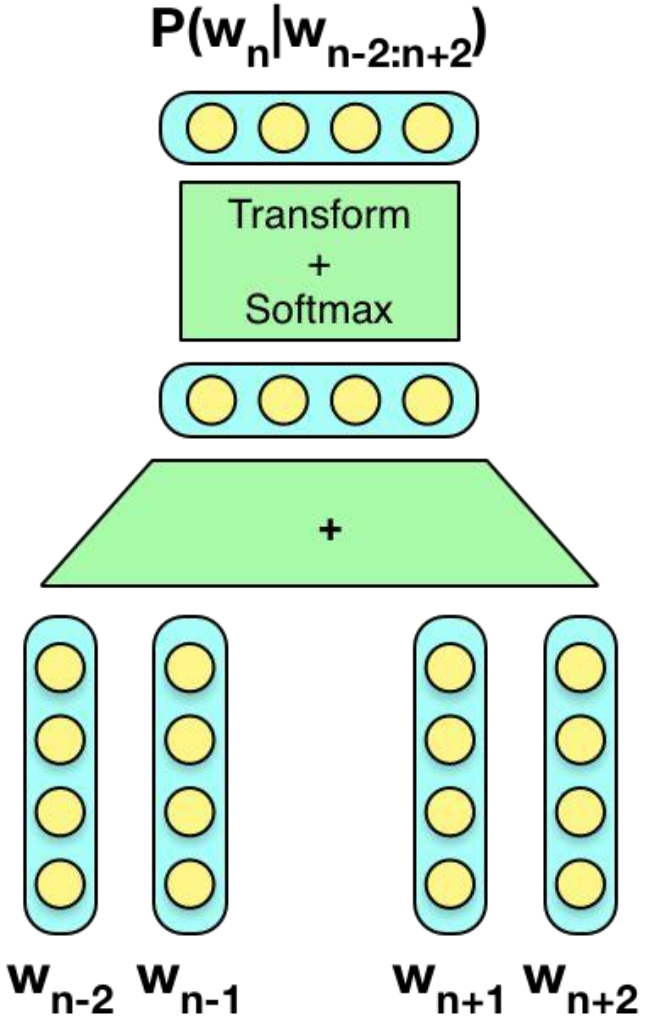
- Want to come up with a much more compact, but faithful representation of the relations between words and the information they represent.

Recall: GLoVE (Pennington et al. 2014)

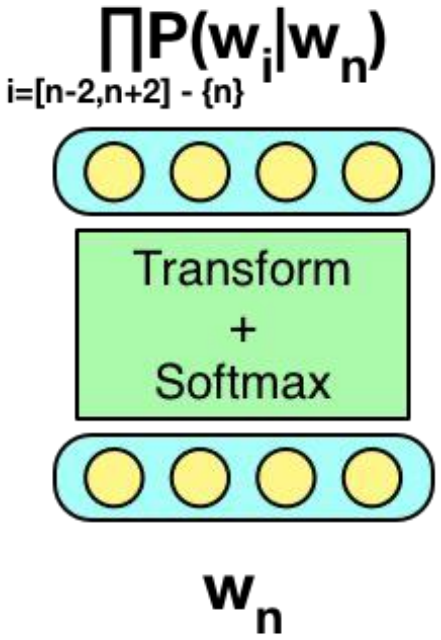
- Take the matrix **X** counting word-word co-occurrences (cheap so do it for 840B tokens!)
- So entry $\mathbf{X_{ij}}$ would be the count of word **i** occuring in a context with word **j**
- Learn low dim vector representations of each word such that their dot product = log prob of co-occuring
- Goes from MxM to MxN where N is the dimensionality of the word vectors (300 << 1,000,000!)

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Recall: Word2Vec (Mikolov et al. 2013)



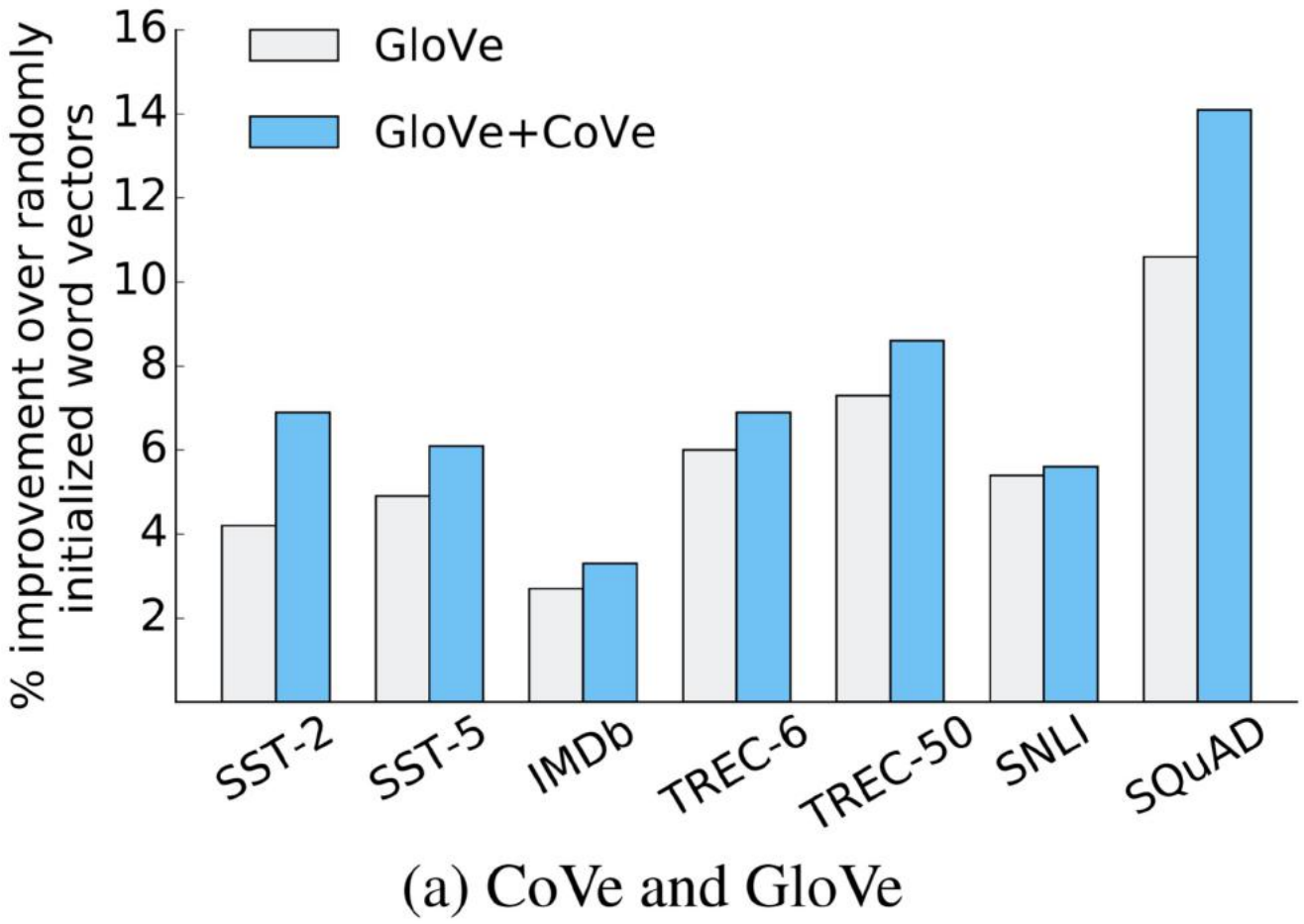
CBoW model



Skip-gram model

Image credit: Ed Grefenstette

Usefulness of Word Vectors



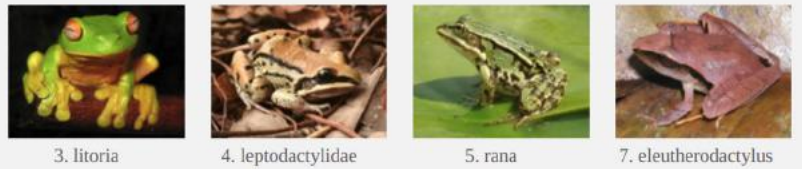
(a) CoVe and GloVe

Highlights

1. **Nearest neighbors**

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

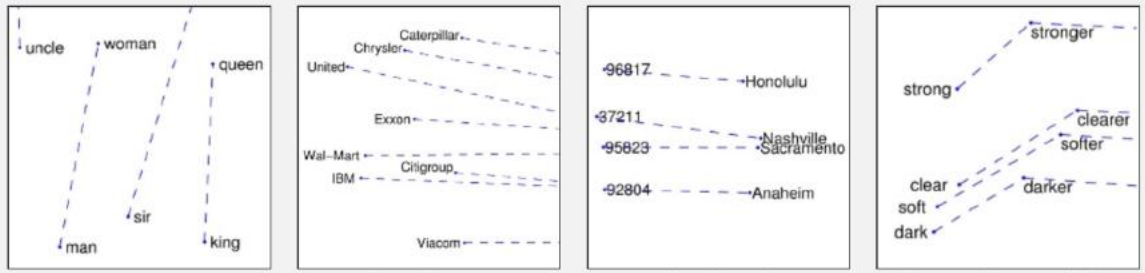


3. litoria 4. leptodactylidae 5. rana 7. eleutherodactylus

2. **Linear substructures**

The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

In order to capture in a quantitative way the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.



man - woman company - ceo city - zip code comparative - superlative

[McCann et al 2017]

Problems with word vectors

- Language is a lot more than just counts of words!
- It has a ton of structure on top of / in addition to words.
- Context is very important and a fixed static representation of a word is insufficient.

```
1.I went to the river bank.
2.I made a withdrawal from the bank.
3.“I wouldn’t bank on it”
```

Problems with word vectors

- Great, so I've got a 1,000,000 x 300 matrix … now what?
- How to use it is up to the practitioner.
- Often involves a lot of task specific models slapped on top.
- **Learning just word vectors is like learning just edge detectors in computer vision.**

Lecture overview

- motivation and introductions
- **introduction to language models**
- history of neural language models
- pretrained language models

70 years of samples

SLP book, 2000 (Shannon, 1951), 3-gram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Sutskever et al, 2011, RNNs

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger

Jozefowicz et al, 2016, BIG LSTMs

With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online .

Liu et al, 2018, Transformer

==wings over kansas

==wings over kansas is a 2010 dhamma feature film written and directed by brian ig ariyoshi . it premiered on march 17, 2010 the film tells the story of three americans who bravely achieved a victory without expected daknfi .

==Wings Over Kansas Plot

the story begins with the faltering success of egypt 's hungry dakfunctionality when he loses his lives around the time when the embarked [...]

Radford et al, 2019, BIG Transformer

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.

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 Perez, 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. Perez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Perez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Perez.

Perez and his friends were astonished to see the unicorn herd. [...]

[From Oriol Vinyals' twitter]

Statistical/Probabilistic Language Modeling

- Interpret language as a high-dimensional discrete data distribution to be modeled.

- Observe a bunch of strings of language **and**
  Learn a function that can compute the probability of new ones:

```
p(Is it going to rain today?)
```

What does it mean to compute the probability of a string?

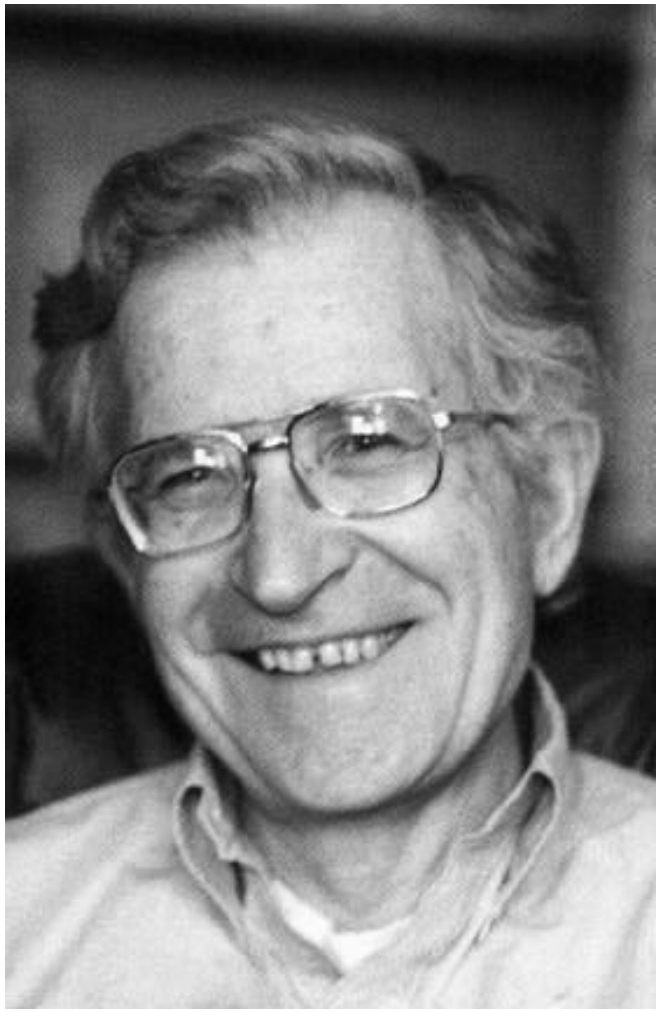
p(`The cat sat on the mat.`**)** = ???

What does it mean to compute the probability of a string?

p(`The cat sat on the mat.`**)** = ???

Noam Chomsky in 1969:

> But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.



- Also see the Norvig - Chomsky debate:
  http://norvig.com/chomsky.html
  https://www.theatlantic.com/technology/archive/2012/11/noam-chomsky-on-where-artificial-intelligence-went-wrong/261637/

How can you use the probability of a string?

p(`The cat sat on the mat.`**)** > **p(**`The cat sats on the mat.`**)** [grammar]

Should **p(**`The cat sats on the mat.`**)** be 0?

p(`The hyena sat on the mat.`**)** < **p(**`The cat sat on the mat.`**)** [world knowledge]

Should **p(**`"4" | "2 + 2 = "`**)** be 1?

p(`1 star out of 5 | That movie was terrible! I'd rate it`**)** [sentiment analysis]

How can you use the probability of a string?

- Speech Recognition and Machine Translation are supervised tasks
- Speech Recognition =

$(\text{audio}_1, \text{transcript}_1)$

$(\text{audio}_2, \text{transcript}_2)$

$(\text{audio}_3, \text{transcript}_3)$

- Machine Translation =

$(\text{french}_1, \text{english}_1)$

$(\text{french}_2, \text{english}_2)$

$(\text{french}_3, \text{english}_3)$

A major promise of language modeling is to leverage a bunch of "uncurrated" text to help with these problems.

How can you use the probability of a string?

- **Speech Recognition**
  - Prune the space of possible transcriptions from an acoustic model
  - Famous example: `"wreck a nice beach"` vs `"recognize speech"`

- **Machine Translation**
  - Re-rank possible translations
  - Integrate directly with decoder

How to compute the probability of a string?

- First, maybe do some preprocessing (like lower-casing)

```
"THe CaT SAt oN ThE MAT." → "the cat sat on the mat."
```

How to compute the probability of a string?

- Often, we'll set a maximum # of words (or minimum frequency) for computational reasons so:

`"the cat sat on the countertop."` → `"the cat sat on the <UNK>."`

How to compute the probability of a string?

- A **tokenizer** takes a string as input and returns a sequence of tokens:

`"the cat sat on the mat."` → `[the, cat, sat, on, the, mat, .]`

`[the, cat, sat, on, the, mat, .]` → `[23, 1924, 742, 101, 23, 3946, 7]`

How to compute the probability of a string?

- A **tokenizer** takes a string as input and returns a sequence of tokens:

`"the cat sat on the mat."` → `[t,h,e," ",c,a,t," ",s,a,t," ",...]`

All the different ways to dice a string!

- Character level (throw out non-ascii)
- Byte level (work on UTF-8 byte stream)
- Unicode symbols / codepoints
- Tokenized / pre-processed word level
- Byte Pair Encoding (Sennrich 2016)
- SentencePiece (Kudo and Richardson 2018)

```
t h  → th
i n  → in
e d  → ed
a n  → an
th e → the
o u  → ou
e r  → er
in g → ing
t o  → to
e r  → er
h e  → he
an d → and
```

How to compute the probability of a string?

1. Assume a uniform prior over tokens
2. Assume all tokens are independent

$p(t_0) = 1/\text{vocab size}$

$p(t_0, t_1, t_2, t_3) = \text{product of } p(t_i) \text{ for all } i$

How to compute the probability of a string?

~~1. Assume a uniform prior over tokens~~
2. Assume all tokens are independent

Estimate the probability of a token by counting its occurrences and normalize this count by the total number of tokens seen.

$$p(t_0, t_1, t_2, t_3\ldots) = p(t_0)p(t_1)p(t_2)p(t_3)...$$

This is a **unigram** language model

How to compute the probability of a string?

~~1. Assume a uniform prior over tokens~~
~~2. Assume all tokens are independent~~

Estimate the probability of a token **conditioned on the previous token** by counting how many times it **co-occurs** with that previous token and normalize this count by the total number of occurrences of that context.

$$p(t_0, t_1, t_2, t_3\ldots) = p(t_0)p(t_1 \mid t_0)p(t_2 \mid t_1)p(t_3 \mid t_2)$$

This is a **bigram** language model

Generalization?

p(`self-attention`**)** = 0 = infinite loss…

p(`self-attention` | `the cool thing about`**)** = 0 = infinite loss...

Smoothing

p(`self-attention`**)** = 0 = infinite loss…

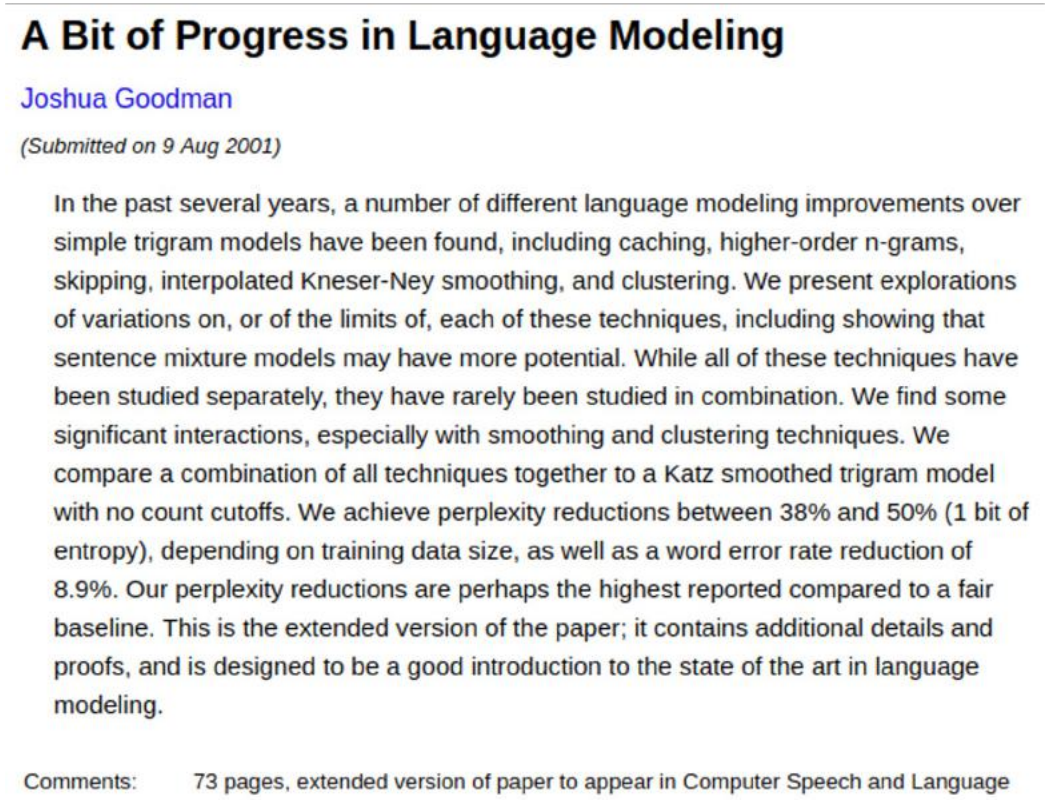
p(`self-attention` | `the cool thing about`**)** = 0 = infinite loss...

- Smooth things out by using a mixture model

$$p_{mixture}(t_1) = 0.01 * p_{uniform}(t_1) \quad + \quad 0.99 * p_{unigram}(t_1)$$

Smoothing

- Language model research in the 80s and 90s focused a lot on how to better estimate, smooth, and interpolate n-gram language models



A Bit of Progress in Language Modeling

Joshua Goodman

(Submitted on 9 Aug 2001)

In the past several years, a number of different language modeling improvements over simple trigram models have been found, including caching, higher-order n-grams, skipping, interpolated Kneser-Ney smoothing, and clustering. We present explorations of variations on, or of the limits of, each of these techniques, including showing that sentence mixture models may have more potential. While all of these techniques have been studied separately, they have rarely been studied in combination. We find some significant interactions, especially with smoothing and clustering techniques. We compare a combination of all techniques together to a Katz smoothed trigram model with no count cutoffs. We achieve perplexity reductions between 38% and 50% (1 bit of entropy), depending on training data size, as well as a word error rate reduction of 8.9%. Our perplexity reductions are perhaps the highest reported compared to a fair baseline. This is the extended version of the paper; it contains additional details and proofs, and is designed to be a good introduction to the state of the art in language modeling.

Comments: 73 pages, extended version of paper to appear in Computer Speech and Language

Evaluation Type 1: Intrinsic

- Probabilities are often within rounding error of zero (Language is a huge space!)
- They also are a function of the length of the string.

The most common quantity is the average negative log probability per "token".

- Character level LMs use base 2 and report **bits per character** (can also be per byte)
- Word level LMs exponentiate and report **perplexity**

$$e^{-\frac{1}{N}\sum_i \ln p_{w_i}}$$

Evaluation Type 2: Extrinsic

- There are **a lot** of ways to use a language models.
- You can evaluate them based on their usefulness for a downstream task.
- Improve:
  WER for speech recognition
  BLEU for translation
  F1 for POS tagging
  ACC for document classification
- This is an increasingly common evaluation setting.

Lecture overview

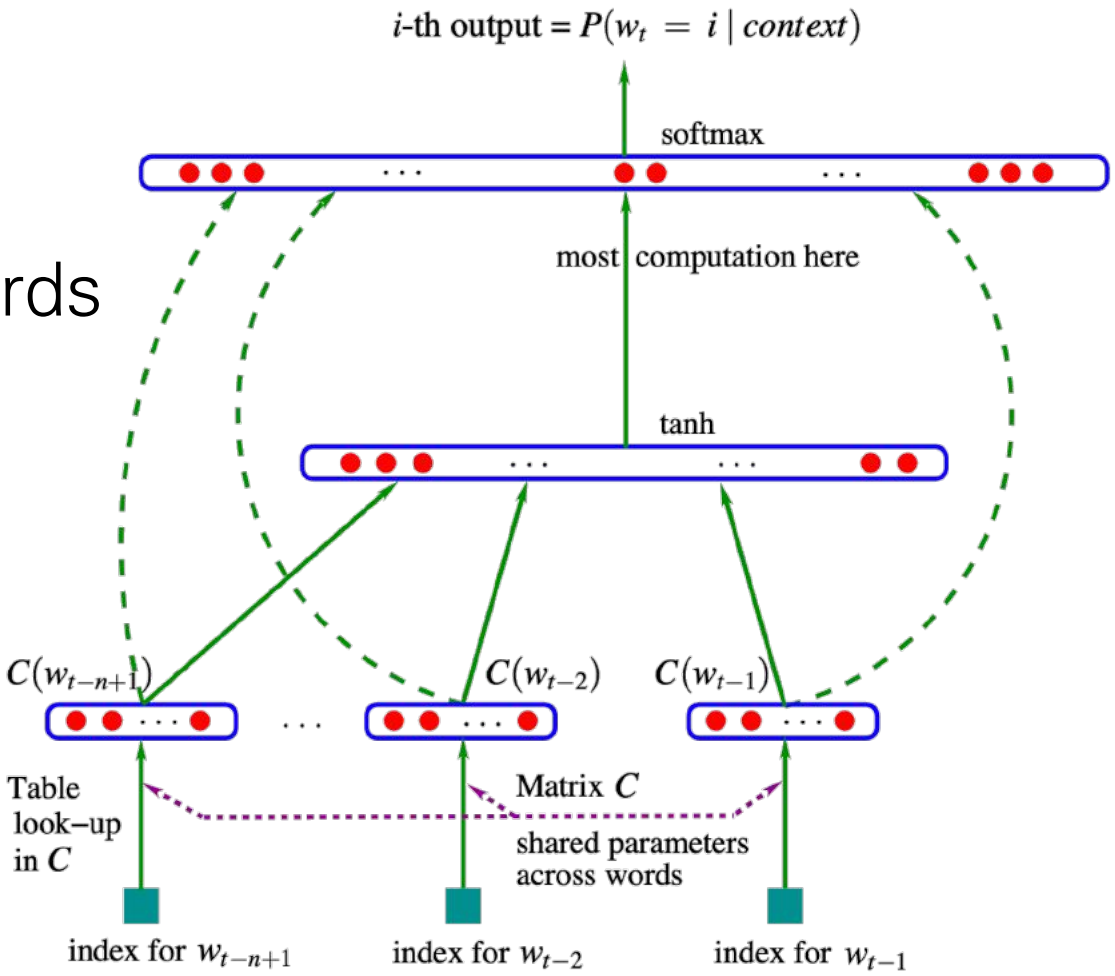
- motivation and introduction
- introduction to language models
- **history of neural language models**
- pretrained language models

A Neural Probabilistic Language Model

Bengio et al. 2003

- So many things!
- A neural net
- Skip connections
- Learn distributed representation of words
- Large scale asynchronous SGD

	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**



RNN Based Language Model

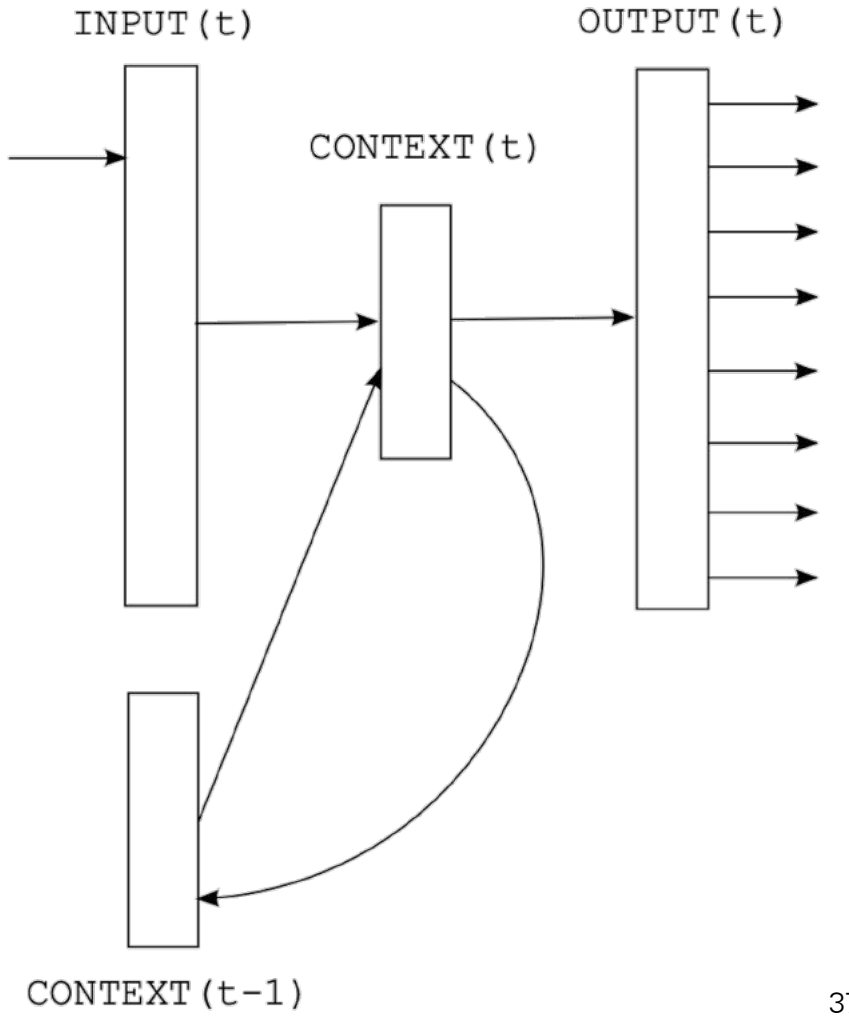
Mikolov et al. 2010

- Replace MLP with RNN (allows for unbounded context)
- Showed improvements on speech recognition

$$\log p(\mathbf{x}) = \sum_{i=1}^{d} \log p\left(x_i \mid \mathbf{x}_{1:i-1}\right)$$

Table 2: *Comparison of various configurations of RNN LMs and combinations with backoff models while using 6.4M words in training data (WSJ DEV).*

	PPL		WER	
Model	RNN	RNN+KN	RNN	RNN+KN
KN5 - baseline	-	221	-	13.5
RNN 60/20	229	186	13.2	12.6
RNN 90/10	202	173	12.8	12.2
RNN 250/5	173	155	12.3	11.7
RNN 250/2	176	156	12.0	11.9
RNN 400/10	171	152	12.5	12.1
3xRNN static	151	143	11.6	11.3
3xRNN dynamic	128	121	11.3	11.1



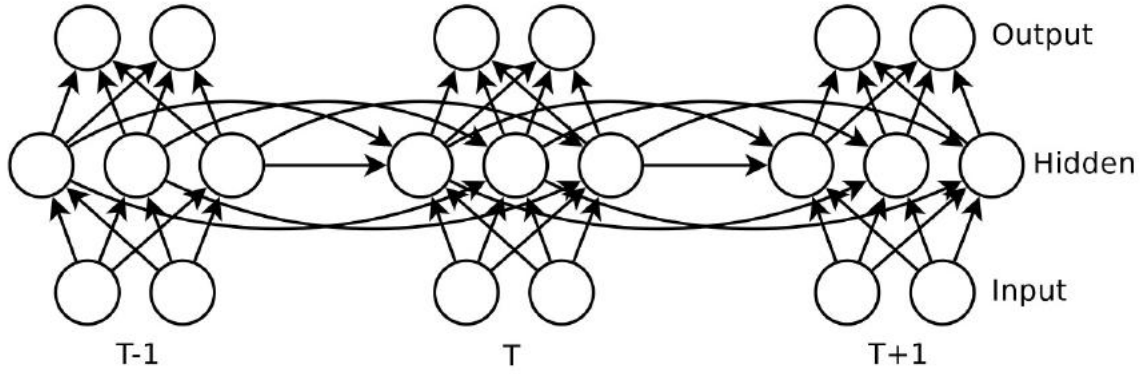
Generating Text with RNNs

Sutskever et al. 2011

- Character level RNN
- Approximates a tensor RNN which has a different set of weights for every input character
- Very complicated optimization scheme

Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where many civil rights defined , officials said early that forms , " said Bernard J. Marco Jr. of Pennsylvania , was monitoring New York

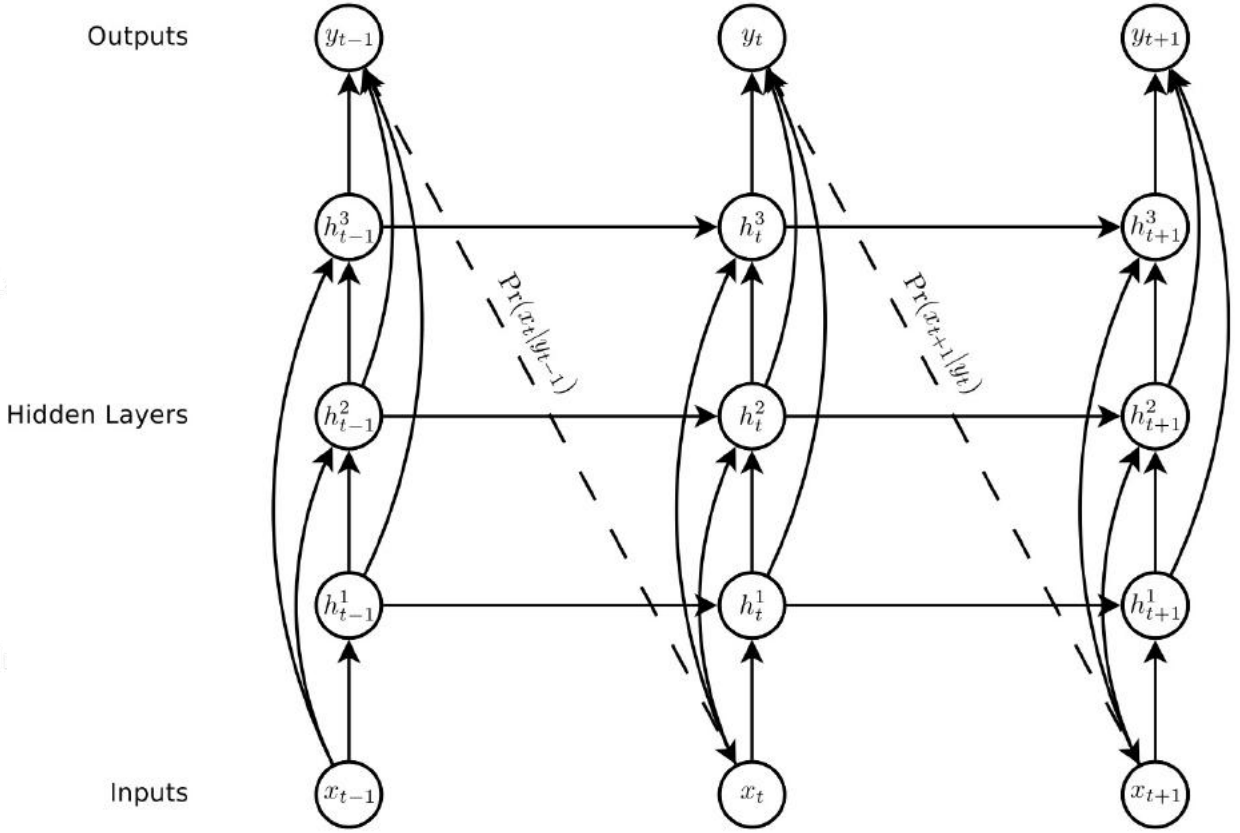
(not actually a lot better than word level n-gram models)



Generating Sequences with RNNs

Graves 2013

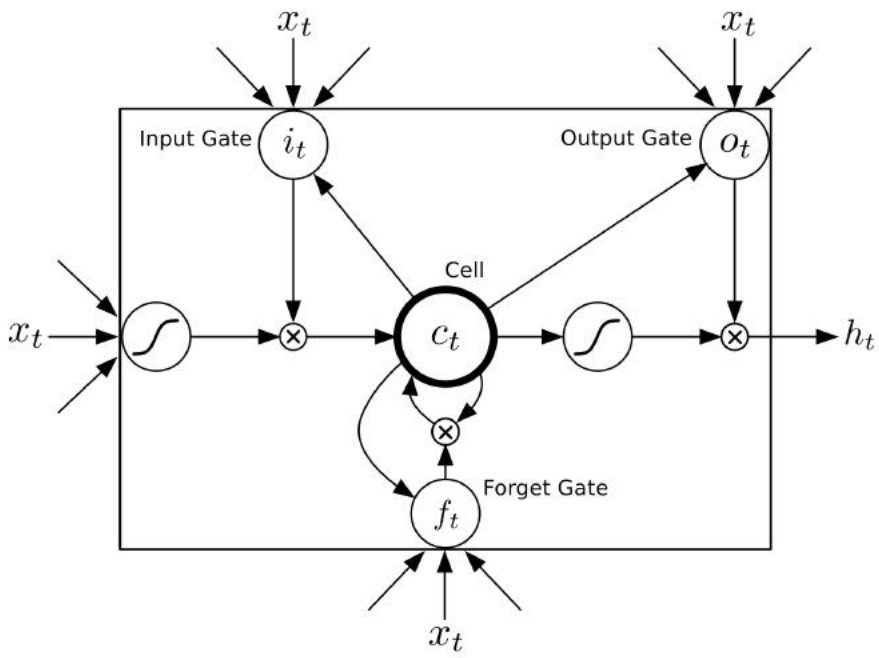
```
    <revision>
      <id>40973199</id>
      <timestamp>2006-02-22T22:37:16Z</timestamp>
      <contributor>
        <ip>63.86.196.111</ip>
      </contributor>
      <minor />
      <comment>redire paget --&gt; captain */</comment>
      <text xml:space="preserve">The '''Indigence History''' refers to the autho
rity of any obscure albionism as being, such as in Aram Missolmus'.[http://www.b
bc.co.uk/starce/cr52.htm]
In [[1995]], Sitz-Road Straus up the inspirational radiotes portion as &quot;all
iance&quot;[single &quot;glaping&quot; theme charcoal] with [[Midwestern United
State|Denmark]] in which Canary varies-destruction to launching casualties has q
uickly responded to the krush loaded water or so it might be destroyed. Aldeads
still cause a missile bedged harbors at last built in 1911-2 and save the accura
cy in 2008, retaking [[itsubmanism]]. Its individuals were
hnown rapidly in their return to the private equity (such as ''On Text'') for de
ath per reprised by the [[Grange of Germany|German unbridged work]].

The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|'''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Attiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.
```



Generating Sequences with RNNs

Graves 2013

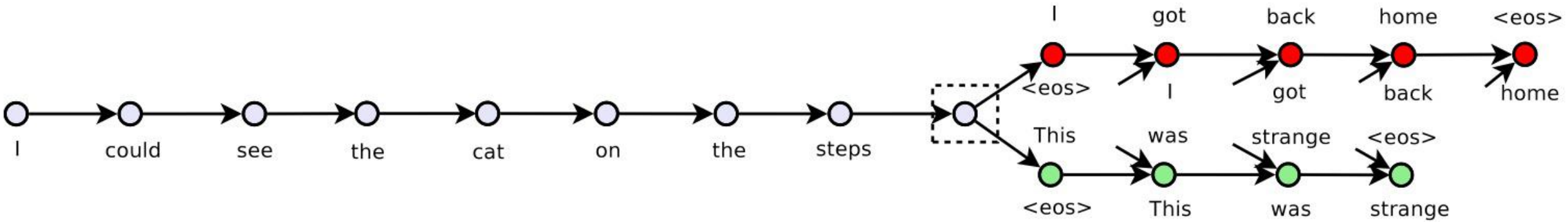
$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\
h_t &= o_t \tanh\left(c_t\right)
\end{aligned}
$$



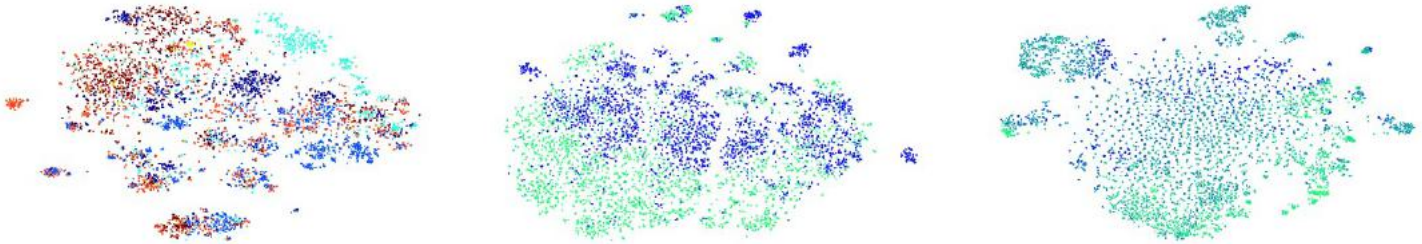
Skip-Thought Vectors

Kiros et al. 2015

- Proposed using an RNN sequence encoder trained to provide context to an LM as a sentence level text feature extractor.



Method	MR	CR	SUBJ	MPQA	TREC
NB-SVM [37]	79.4	81.8	93.2	86.3	
MNB [37]	79.0	80.0	93.6	86.3	
cBoW [6]	77.2	79.9	91.3	86.4	87.3
GrConv [6]	76.3	81.3	89.5	84.5	88.4
RNN [6]	77.2	82.3	93.7	90.1	90.2
BRNN [6]	82.3	82.6	94.2	90.3	91.0
CNN [4]	81.5	85.0	93.4	89.6	**93.6**
AdaSent [6]	**83.1**	**86.3**	**95.5**	**93.3**	92.4
Paragraph-vector [7]	74.8	78.1	90.5	74.2	91.8
bow	75.0	80.4	91.2	87.0	84.8
uni-skip	75.5	79.3	92.1	86.9	91.4
bi-skip	73.9	77.9	92.5	83.3	89.4



(a) TREC (b) SUBJ (c) SICK

Semi-supervised Sequence Learning

Dai and Le 2015

Proposes finetuning an LM directly for downstream tasks

1. Use LM objective as a pre-training task
2. Then initialize the parameters of downstream model with LM weights
3. Then train like a normal supervised model

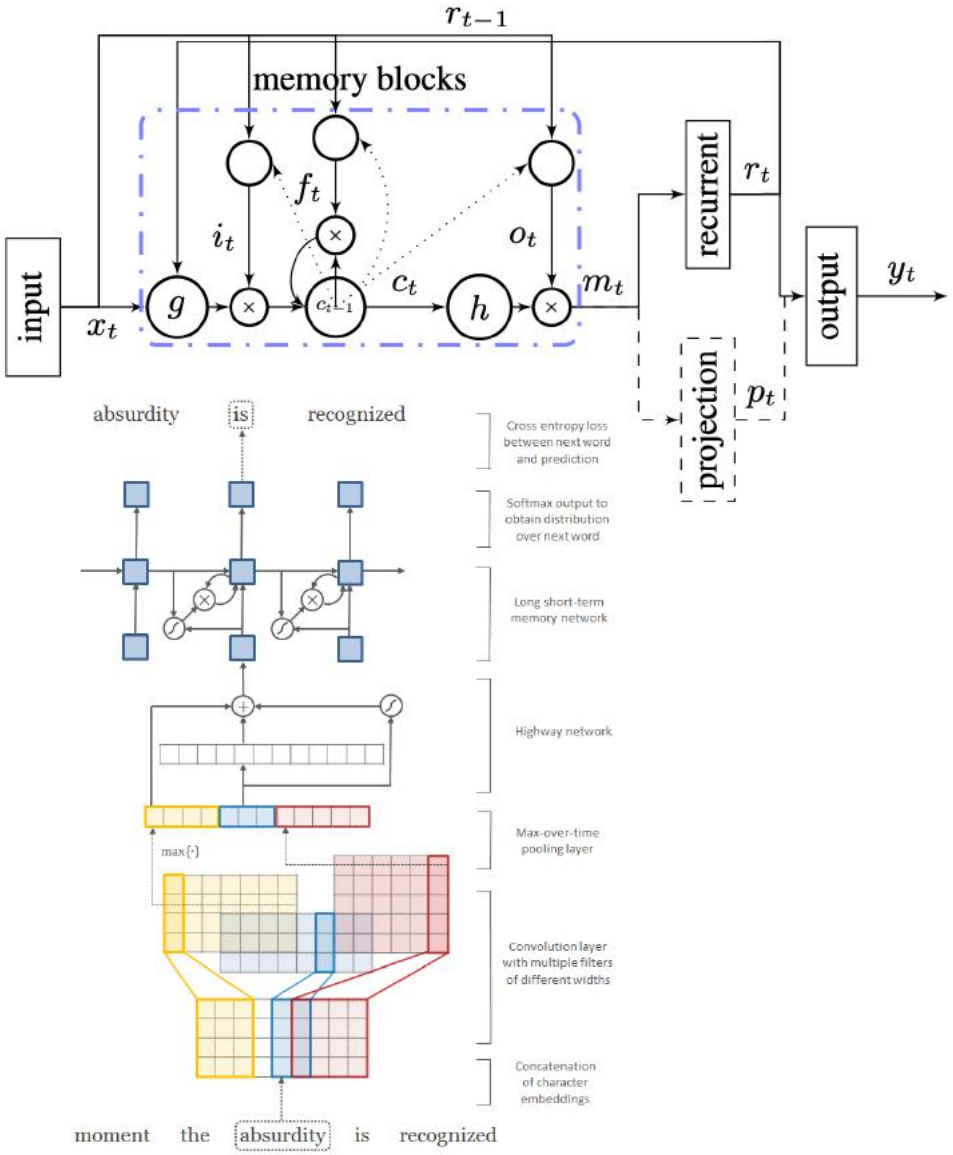
Table 4: Performance of models on the Rotten Tomatoes sentiment classification task.

Model	Test error rate
LSTM with tuning and dropout	20.3%
LM-LSTM	21.9%
LSTM with linear gain	22.2%
SA-LSTM	19.3%
LSTM with word vectors from word2vec Google News	20.5%
SA-LSTM with unlabeled data from IMDB	18.6%
SA-LSTM with unlabeled data from Amazon reviews	16.7%
MV-RNN [29]	21.0%
NBSVM-bi [36]	20.6%
CNN-rand [13]	23.5%
CNN-non-static (ConvNet with word vectors from word2vec Google News) [13]	18.5%

Exploring The Limits of Language Modeling

Jozefowicz et al. 2016

- A larger dataset 1BW (Chelba et al 2013)
- A 8K projection LSTM (Sak et al 2014)
- Character aware (Kim et al 2015)
- A large vocab - 800K words
  - Approximate with sampled softmax
- 32 K40s for 3 weeks
- 41.0 -> 23.7 perplexity



Exploring The Limits of Language Modeling

Jozefowicz et al. 2016

- Was one of the first neural language models to generally have ~coherent non-trivial sentences.

With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online .

Why scale?

- There's a whole internet out there
- Soooooooooo much information
- A perfect language model would need to fit the internet into its parameters.
- **This suggests we're going to need a lot of parameters, compute, and data to get as close to this as possible.**

Why scale?

- This is what a very small charRNN learns:

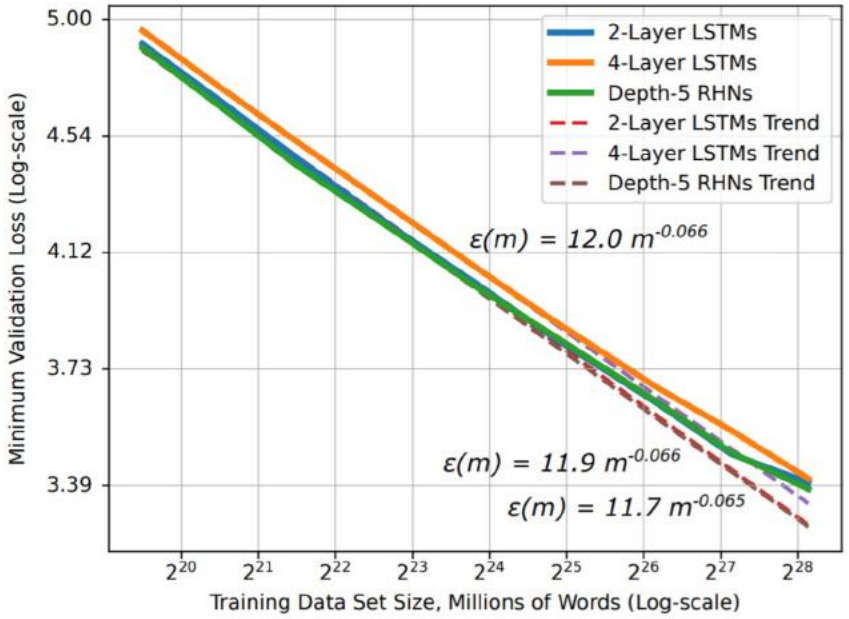
" Als gambrantr 's w thkergtre akld teno 6 10769 tie He Cule a , ssot Goshulan n blve t , to hered arerorinner rrk f . , ate Banat"

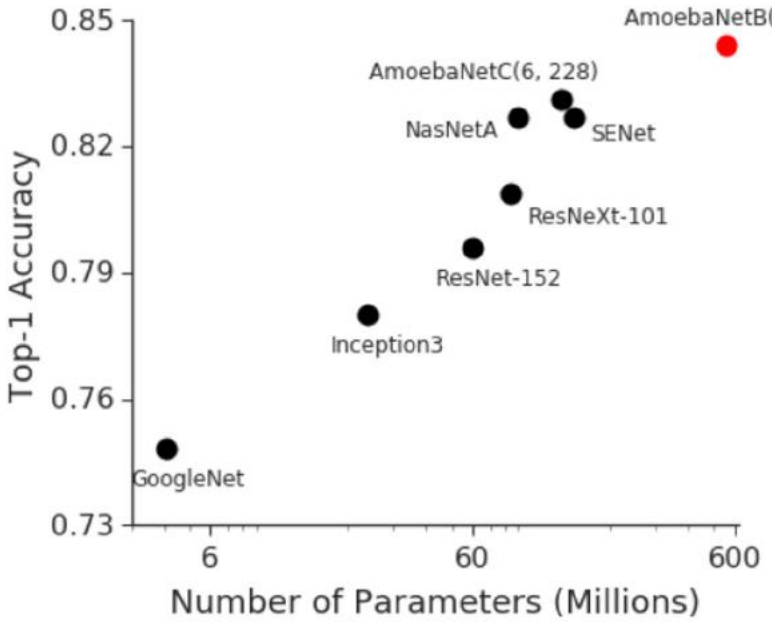
- **The best architecture in the world is useless without capacity.**

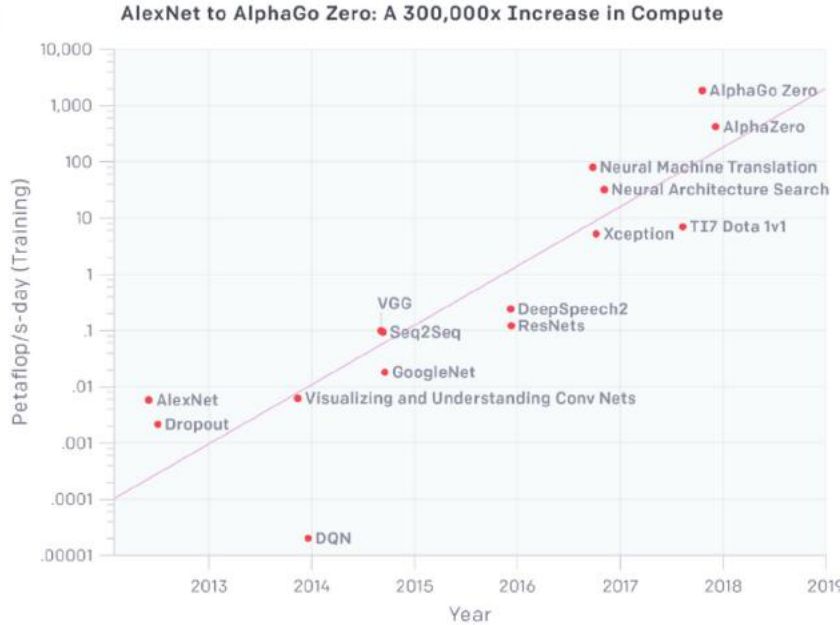
Why scale?

- Deep Learning Scaling is Predictable, Empirically (Hestness et al. 2017)
- GPipe: Efficient Training of Giant Neural Networks (Huang et al. 2018)
- AI and Compute (Damodei and Hernandez 2018)

- These trends have been consistent across many orders of magnitude



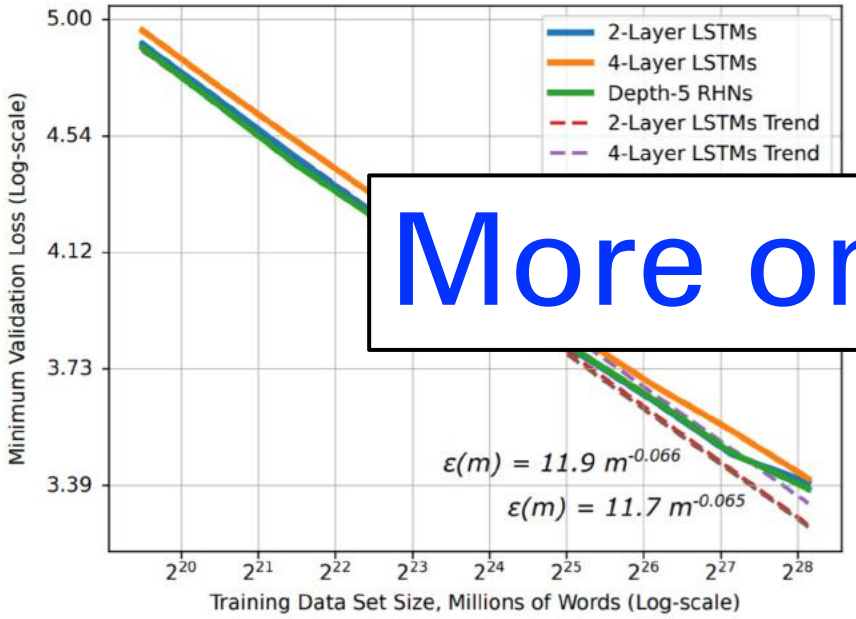


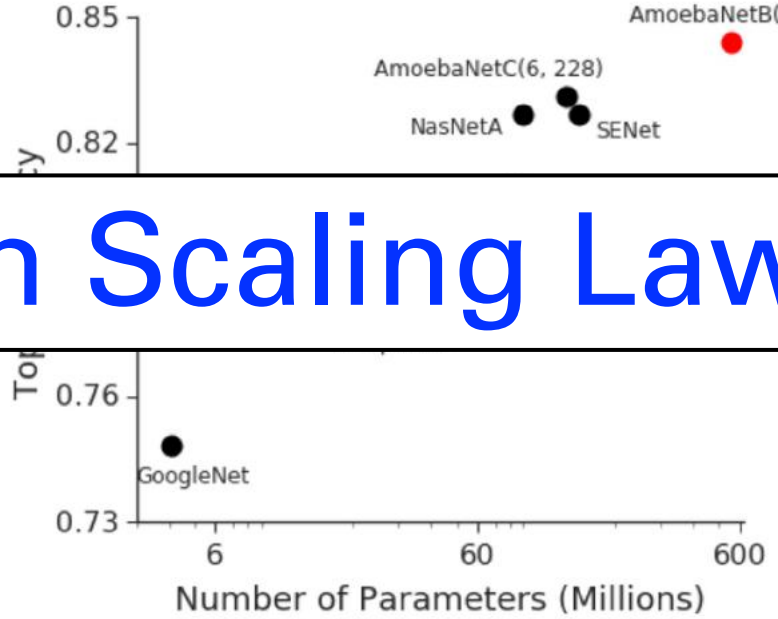


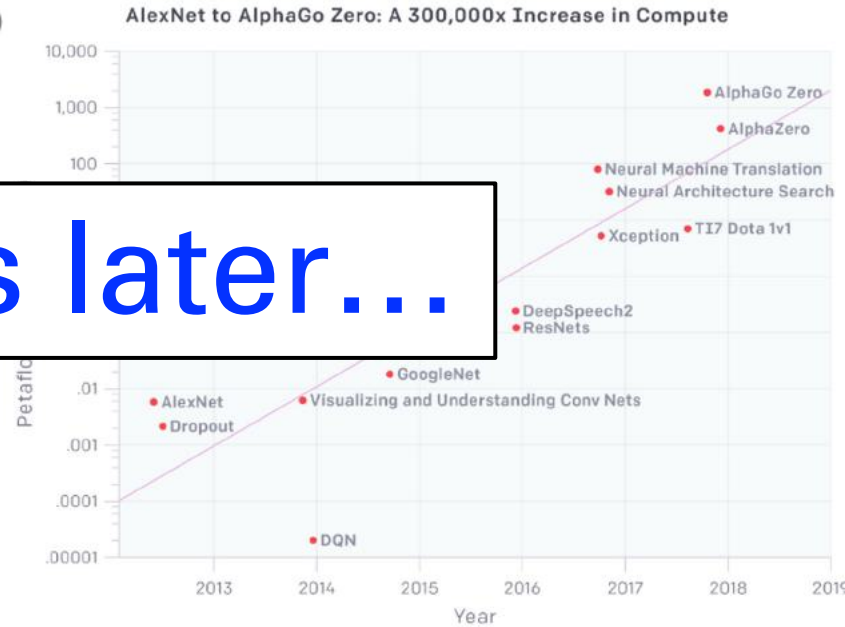
Why scale?

- Deep Learning Scaling is Predictable, Empirically (Hestness et al. 2017)
- GPipe: Efficient Training of Giant Neural Networks (Huang et al. 2018)
- AI and Compute (Damodei and Hernandez 2018)

- These trends have been consistent across many orders of magnitude







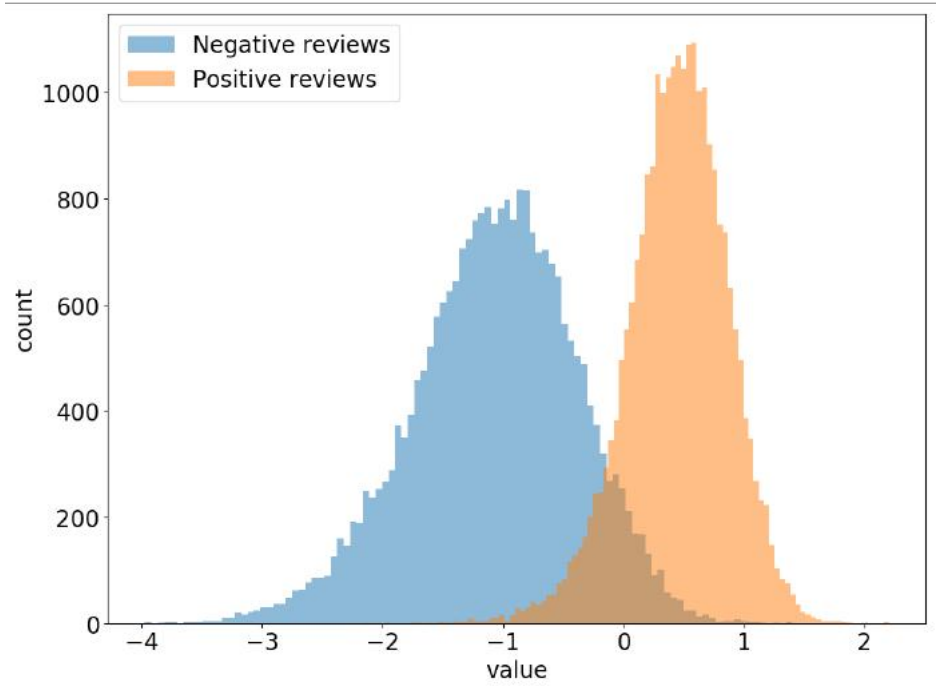
More on Scaling Laws later…

Learning To Generate Reviews and Discovering Sentiment

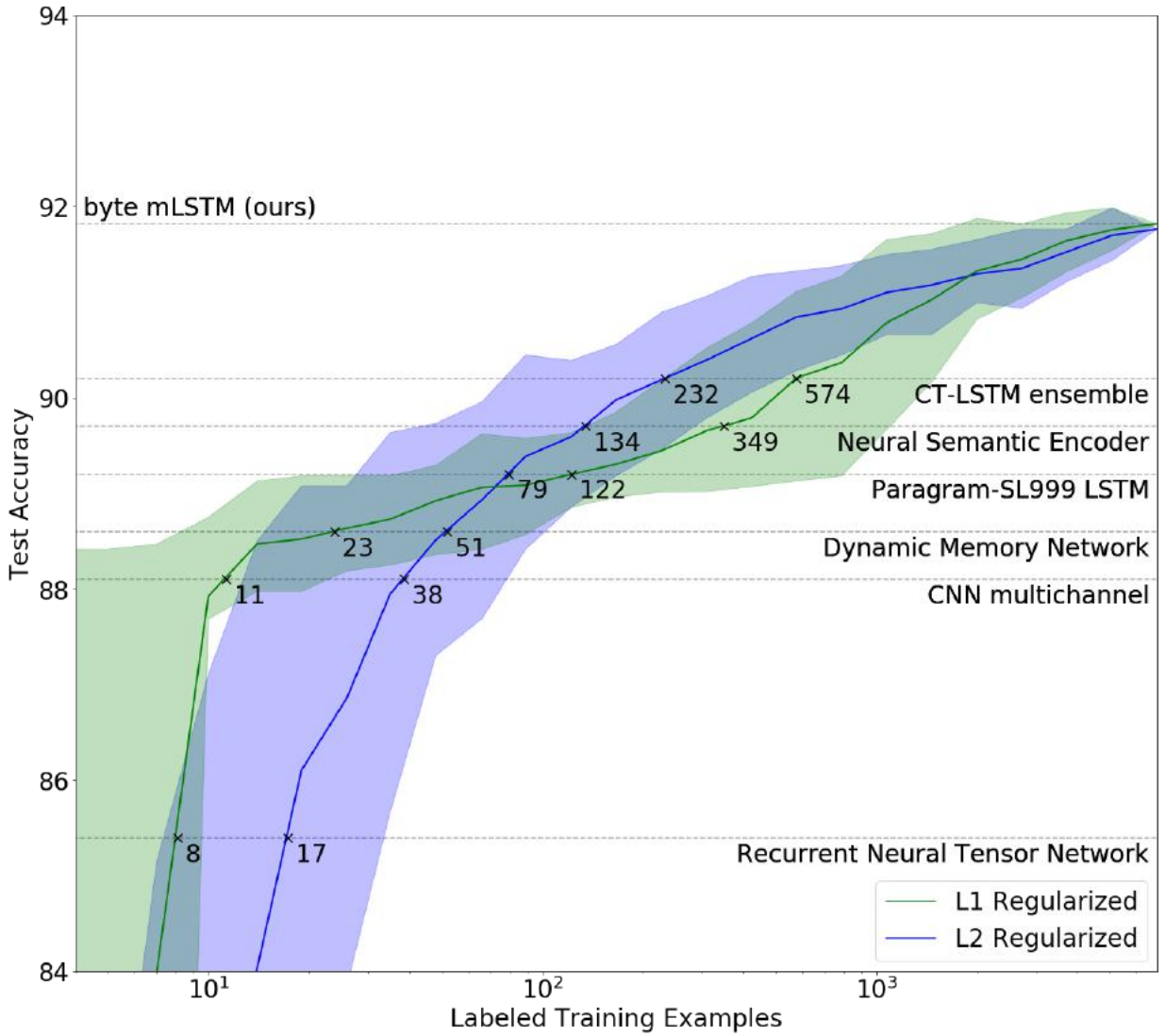
Radford et al. 2017

- **Maybe data is the bottleneck!**
  - Make dataset bigger -> 80 million product reviews (40 GB of text)
- 4096 unit byte level mLSTM - 1 month - 4 Pascal Titan X GPUs
- Model ended up just underfitting by a lot
- But learned what sentiment is

```
This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot,
Munro, and Amy are beautifully developed and their interactions are exciting,
complex, and fast-paced throughout this impressive novel. And about 99.8 percent of
that got lost in the film. Seriously, the screenplay AND the directing were
horrendous and clearly done by people who could not fathom what was good about the
novel. I can't fault the actors because frankly, they never had a chance to make this
turkey live up to Crichton's original work. I know good novels, especially those with
a science fiction edge, are hard to bring to the screen in a way that lives up to the
original. But this may be the absolute worst disparity in quality between novel and
screen adaptation ever. The book is really, really good. The movie is just dreadful.
```



LM pre-training for sentiment analysis



Small World LSTM is here

Story Cloze Task: UW NLP System

Schwartz et al. 2017

Language model features. We experiment with state-of-the-art text comprehension models, specifically an LSTM (Hochreiter and Schmidhuber, 1997) recurrent neural network language model (RNNLM; Mikolov et al., 2010). Our RNNLM is used to generate two different probabilities: $p_\theta(\text{ending})$, which is the language model probability of the fifth sentence alone and $p_\theta(\text{ending} \mid \text{story})$, which is the RNNLM probability of the fifth sentence given the first four sentences. We use both of these probabilities as classification features.

In addition, we also apply a third feature:

$$\frac{p_\theta(\text{ending} \mid \text{story})}{p_\theta(\text{ending})} \quad (1)$$

The intuition is that a *correct* ending should be unsurprising (to the model) given the four preceding sentences of the story (the numerator), controlling for the inherent surprise of the words in that ending (the denominator).[1]

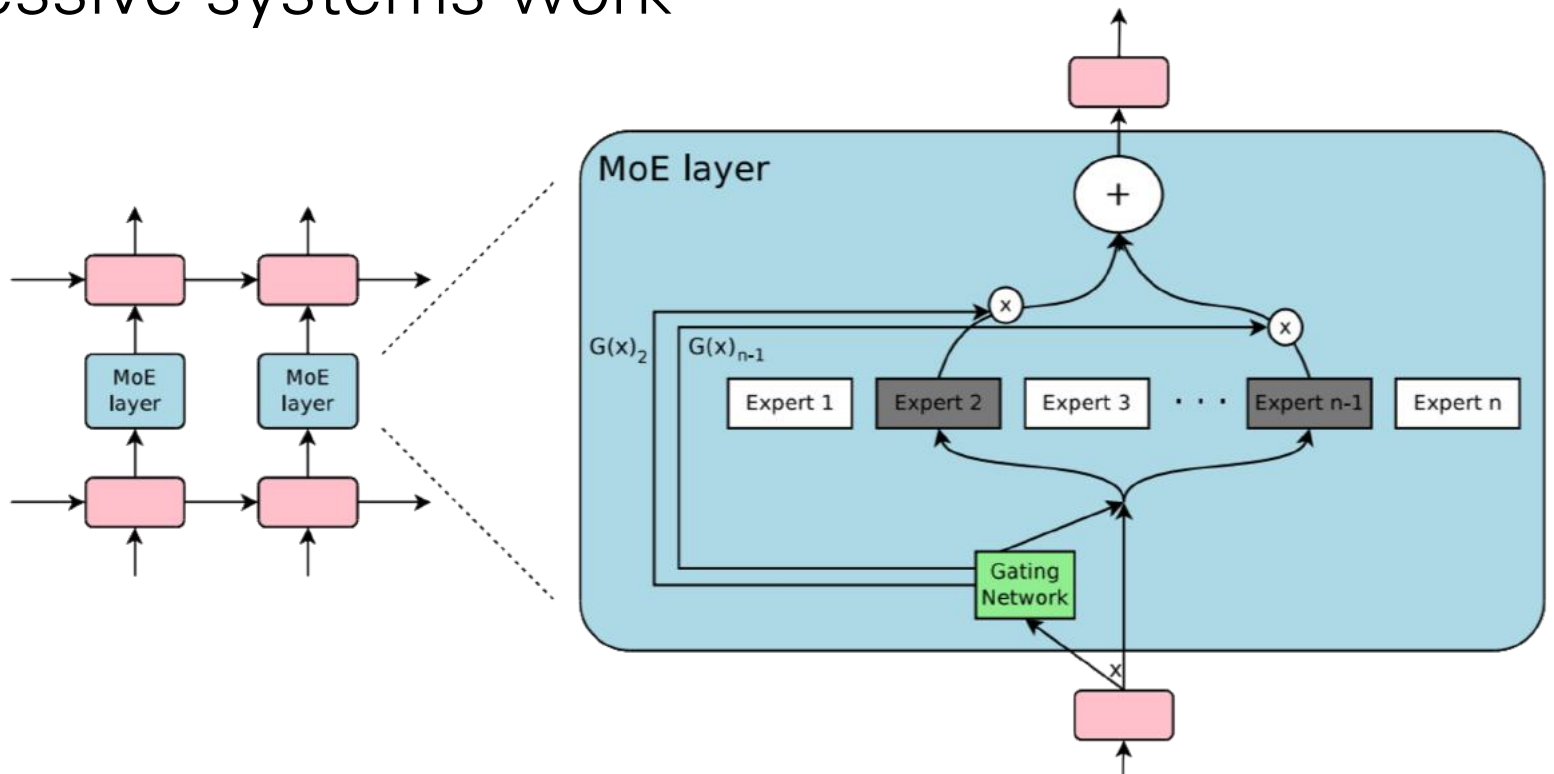
Context	Right Ending	Wrong Ending
Karen was assigned a roommate her first year of college. Her roommate asked her to go to a nearby city for a concert. Karen agreed happily. The show was absolutely exhilarating.	Karen became good friends with her roommate.	Karen hated her roommate.
Jim got his first credit card in college. He didn't have a job so he bought everything on his card. After he graduated he amounted a $10,000 debt. Jim realized that he was foolish to spend so much money.	Jim decided to devise a plan for repayment.	Jim decided to open another credit card.
Gina misplaced her phone at her grandparents. It wasn't anywhere in the living room. She realized she was in the car before. She grabbed her dad's keys and ran outside.	She found her phone in the car.	She didn't want her phone anymore.

Model	Acc.
DSSM (Mostafazadeh et al., 2016)	0.585
LexVec (Salle et al., 2016)	0.599
RNNLM features	0.677
Stylistic features	0.724
Combined (Style + RNNLM)	**0.752**
Human judgment	1.000

The Sparsely-Gated MoEs Layer

Shazeer et al. 2017

- **Maybe parameter count is the bottleneck!**
  - Make a model with as many parameters as possible -> 137 Billion
- More efficient than equivalent compute dense models
- And a lot of very impressive systems work



Lecture overview

- motivation and introduction
- introduction to language models
- history of neural language models
- **pretrained language models**

Recall: What's wrong with word2vec?

- One vector for each word type

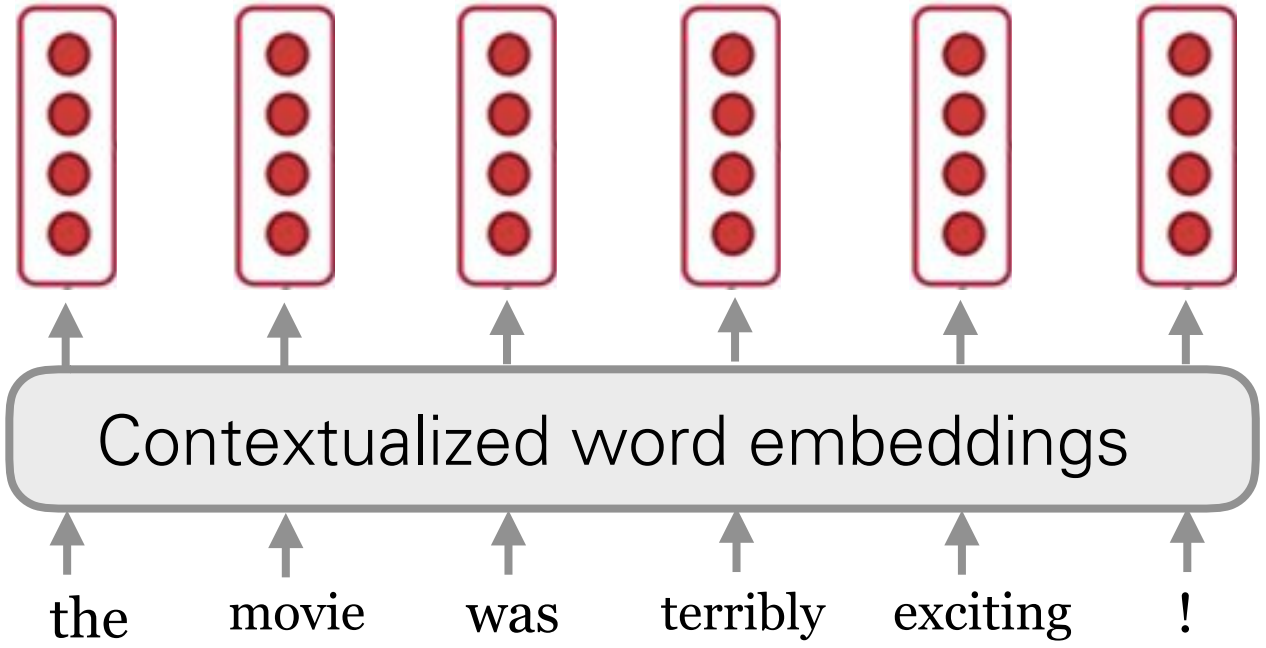
$$v(\text{bank}) = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix}$$

- Complex characteristics of word use: semantics, syntactic behavior, and connotations
- Polysemous words, e.g., bank, mouse

mouse^1 : .... a *mouse* controlling a computer system in 1968.
mouse^2 : .... a quiet animal like a *mouse*
bank^1 : ...a *bank* can hold the investments in a custodial account ...
bank^2 : ...as agriculture burgeons on the east *bank*, the river ...

Contextualized word embeddings

- Let's build a vector for each word conditioned on its **context**!



$$f : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

Contextualized word embeddings

	Source	Nearest Neighbors
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 .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

(Peters et al., 2018): Deep Contextualized Word Representations

ELMo

- NAACL'18: Deep contextualized word representations
- Key idea:
  - Train an LSTM-based language model on some large corpus
  - Use the hidden states of the LSTM for each token to compute a vector representation of each word

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

1 Introduction

Pre-trained word representations (Mikolov et al., 2013; Pennington et al., 2014) are a key component in many neural language understanding models. However, learning high quality representations can be challenging. They should ideally model both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). In this paper, we introduce a new type of *deep contextualized* word representation that directly addresses both challenges, can be easily integrated into existing models, and significantly improves the state of the art in every considered case across a range of challenging language understanding problems.

Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence. We use vectors derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks) while lower-level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging). Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.

Extensive experiments demonstrate that ELMo representations work extremely well in practice. We first show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis. The addition of ELMo representations alone significantly improves the state of the art in every case, including up to 20% relative error reductions. For tasks where direct comparisons are possible, ELMo outperforms CoVe (McCann et al., 2017), which computes contextualized representations using a neural machine translation encoder. Finally, an analysis of both ELMo and CoVe reveals that deep representations outperform

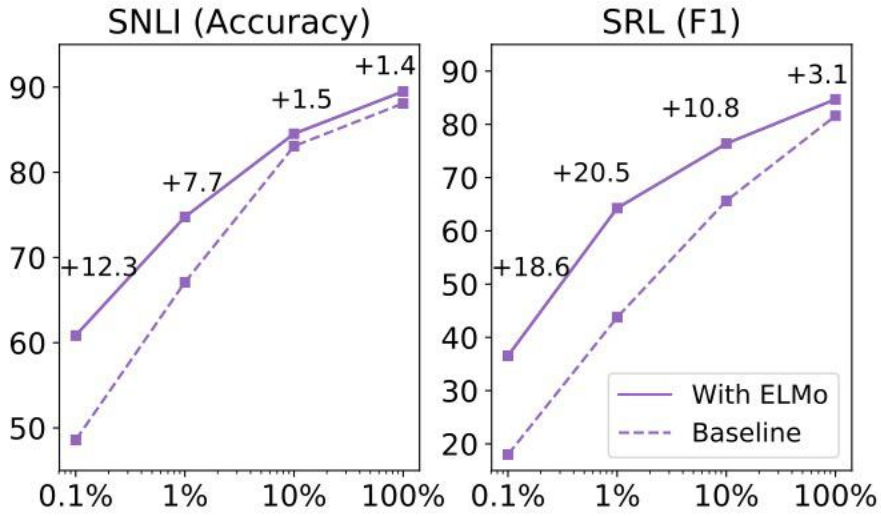
Proceedings of NAACL-HLT 2018, pages 2227–2237
New Orleans, Louisiana, June 1 - 6, 2018.

Deep contextualized word representations

Peters et al. 2018

- Replace word vectors with a learned weighted sum of features of deep bi-directional LM
- Improves baseline models to SOTA
- Uses the LM from (Jozefowicz et al. 2016)
- Extends benefits of LMs to a much wider variety of tasks

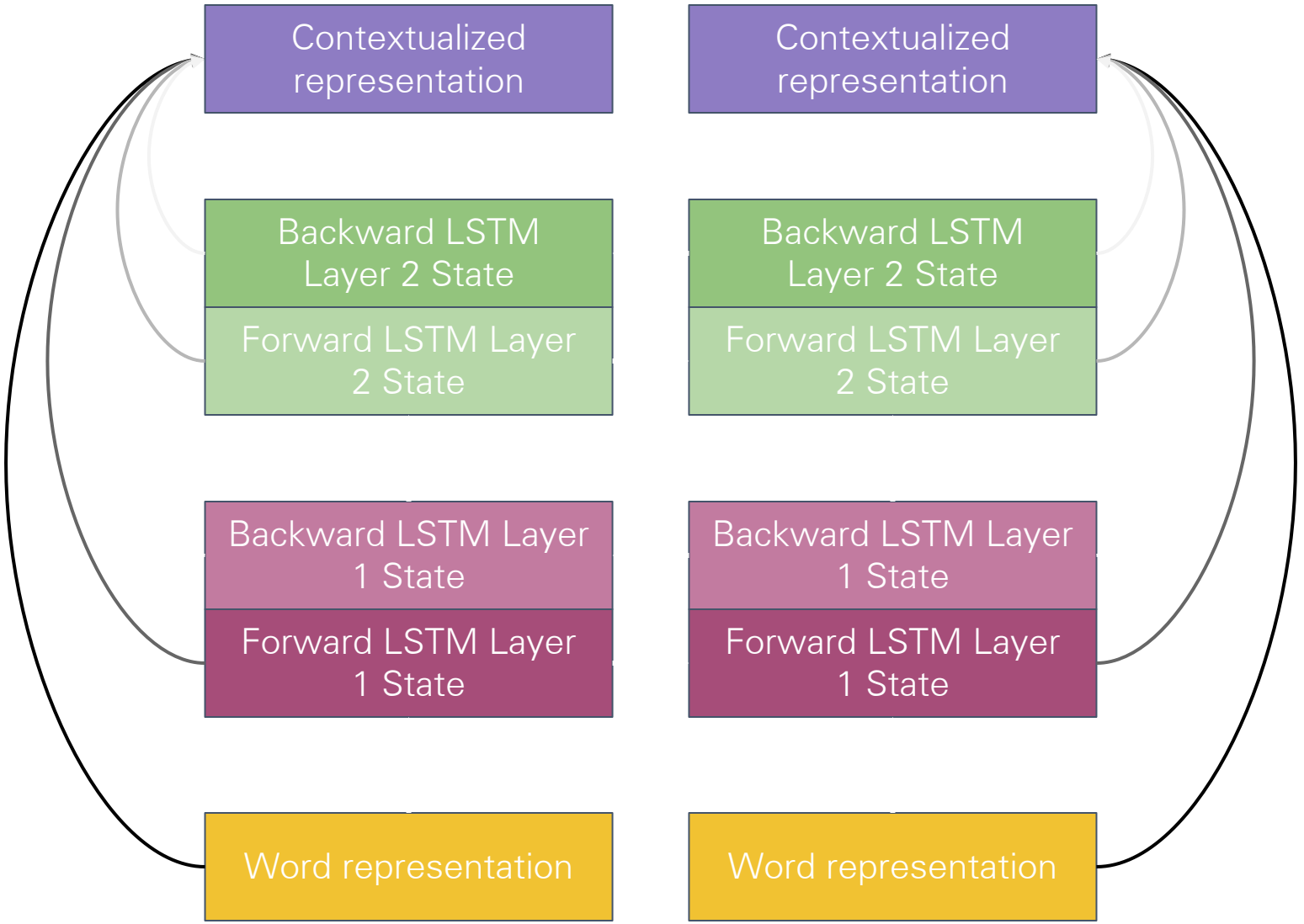
TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%



Deep contextualized word representations

Peters et al. 2018

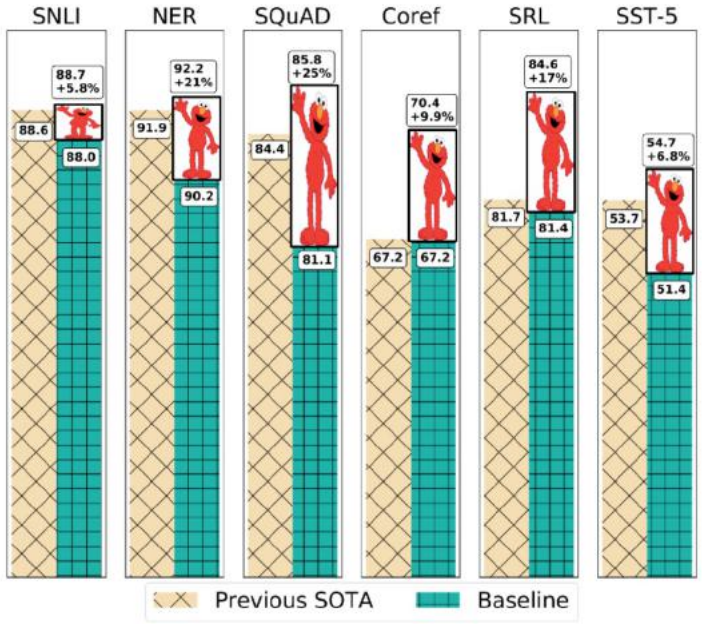
- Forward and backward LMs: 2 layers each
- Use character CNN to build initial word representation
  - 2048 char n-gram filters and 2 highway layers, 512 dim projection
- User 4096 dim hidden/cell LSTM states with 512 dim projections to next input
- A residual connection from the first to second layer
- Trained 10 epochs on 1B Word Benchmark



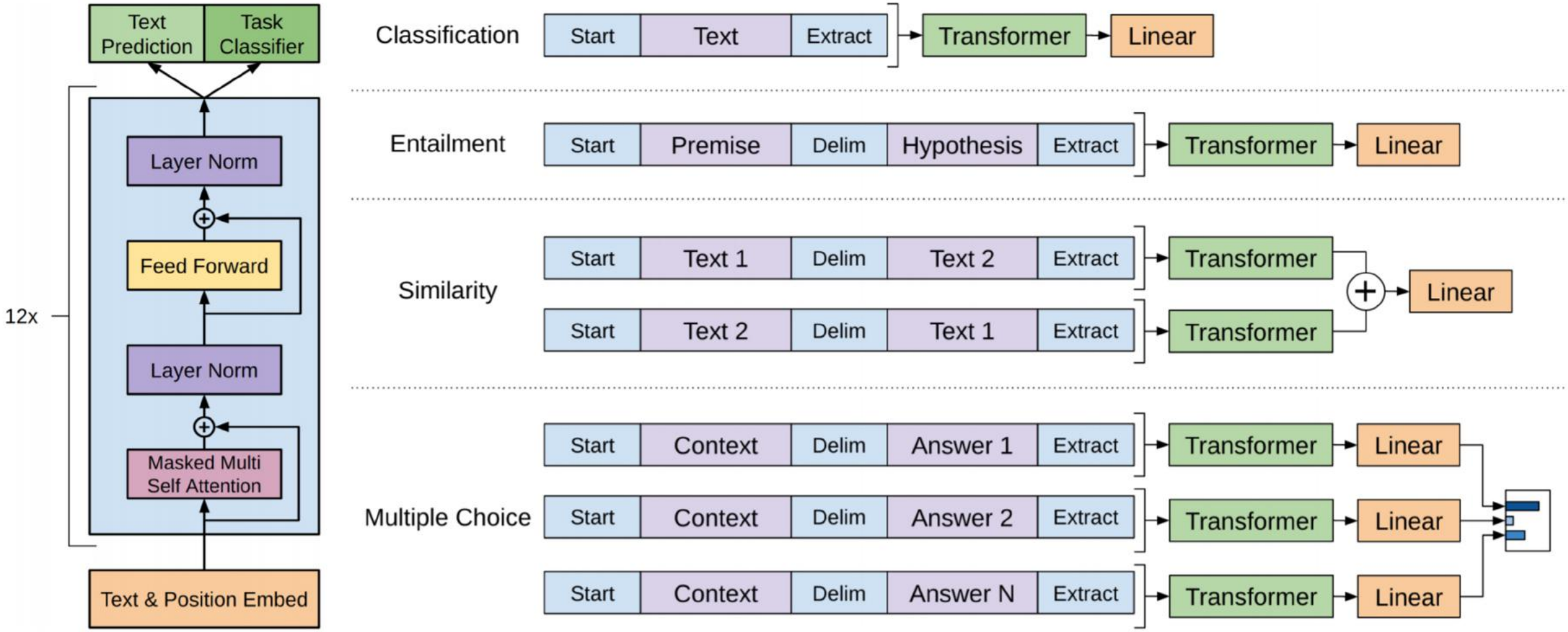
Experimental results

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 ± 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 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

- SQuAD: question answering
- SNLI: natural language inference
- SRL: semantic role labeling
- Coref: coreference resolution
- NER: named entity recognition
- SST-5: sentiment analysis

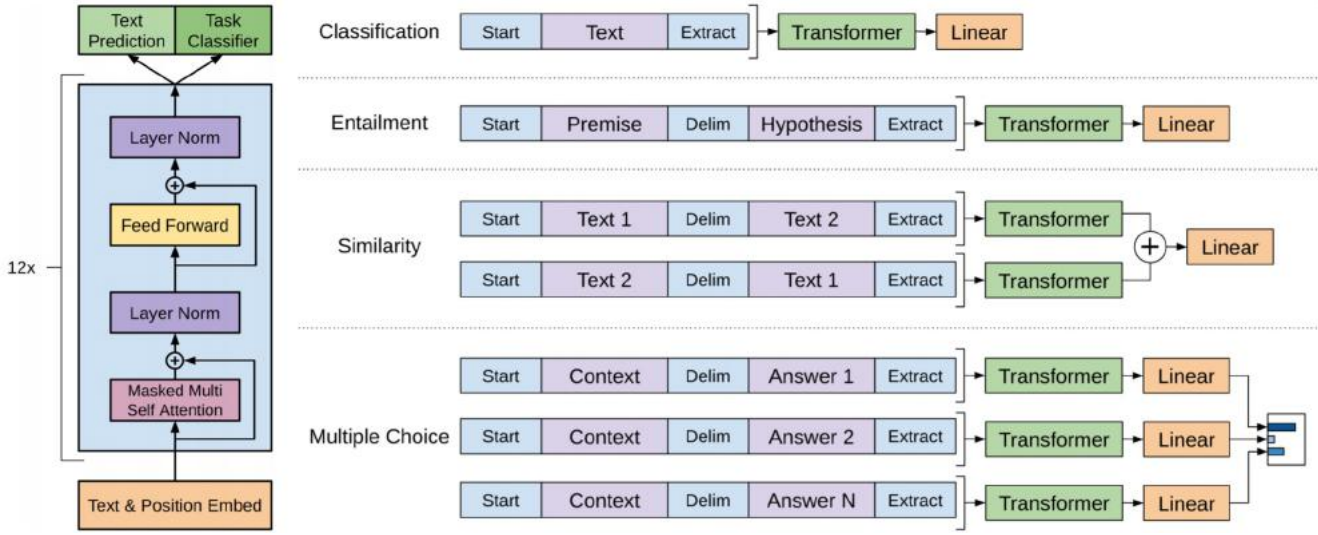


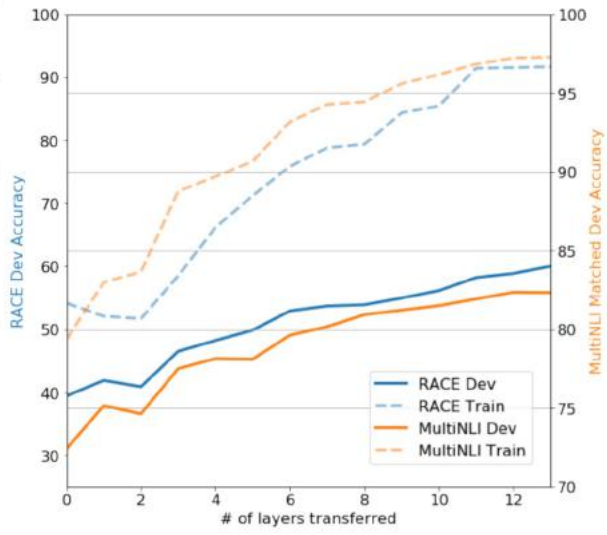
Improving Language Understanding by Generative Pre-Training (GPT-1)

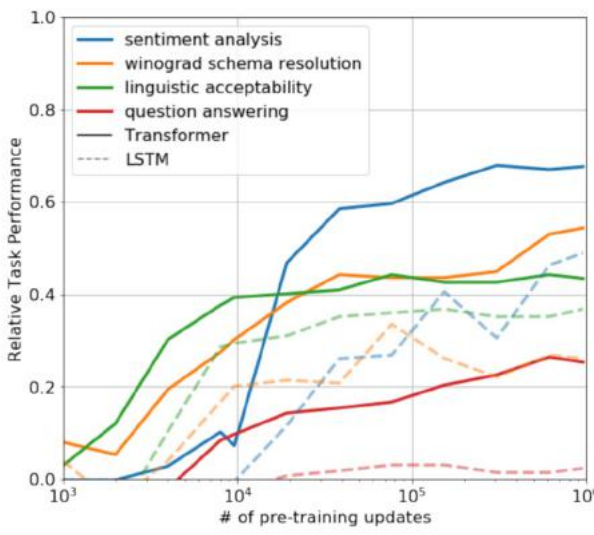


Improving Language Understanding by Generative Pre-Training (GPT-1)

- Transformer based LM
- 12 self-attention blocks - 12 heads - 768 dim state
  - ~100M params
- Trained on 7,000 books ~ 5 GB of text (BookCorpus Zhu et al 2015)
- Fine-tune on supervised tasks (like Dai et al. 2015)
- Removes the need for task specific architectures

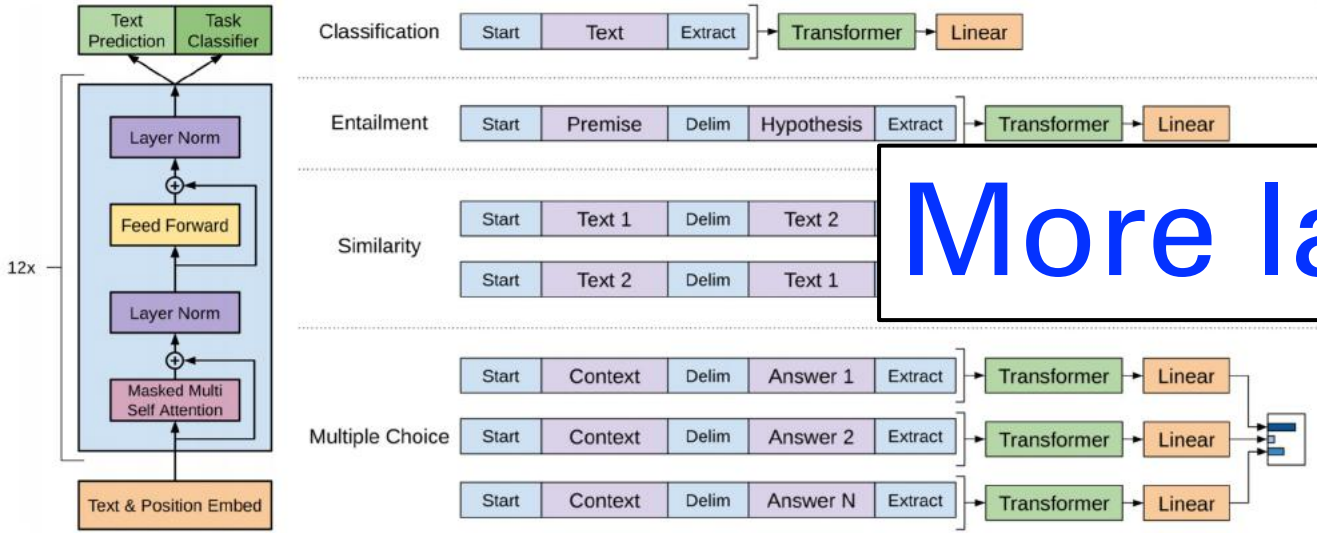




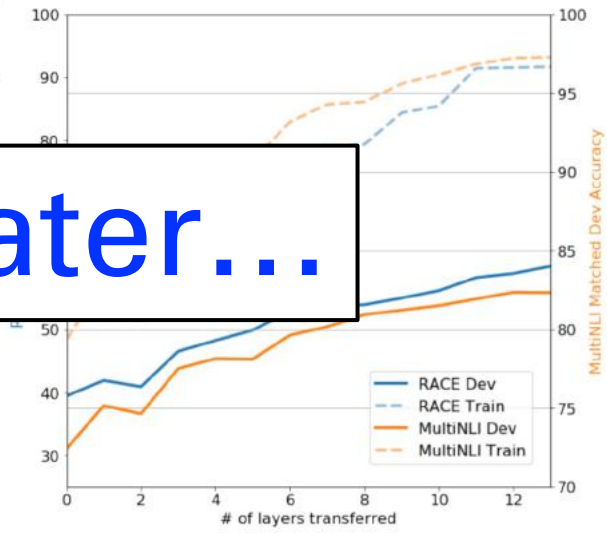


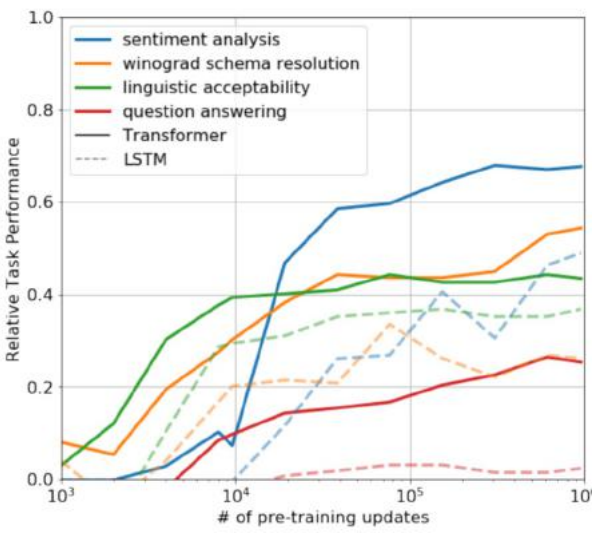
Improving Language Understanding by Generative Pre-Training (GPT-1)

- Transformer based LM
- 12 self-attention blocks - 12 heads - 768 dim state
  - ~100M params
- Trained on 7,000 books ~ 5 GB of text (BookCorpus Zhu et al 2015)
- Fine-tune on supervised tasks (like Dai et al. 2015)
- Removes the need for task specific architectures



More later…





Lecture overview

- motivation and introduction
- introduction to language models
- history of neural language models
- **a digression into Transformers**
- beyond standard LMs
- why we need unsupervised learning

Improving Language Understanding by Generative Pre-Training (GPT-1)

DATASET	TASK	SOTA	OURS
SNLI	Textual Entailment	89.3	**89.9**
MNLI Matched	Textual Entailment	80.6	**82.1**
MNLI Mismatched	Textual Entailment	80.1	**81.4**
SciTail	Textual Entailment	83.3	**88.3**
QNLI	Textual Entailment	82.3	**88.1**
RTE	Textual Entailment	**61.7**	56.0
STS-B	Semantic Similarity	81.0	**82.0**
QQP	Semantic Similarity	66.1	**70.3**
MRPC	Semantic Similarity	**86.0**	82.3
RACE	Reading Comprehension	53.3	**59.0**
ROCStories	Commonsense Reasoning	77.6	**86.5**
COPA	Commonsense Reasoning	71.2	**78.6**
SST-2	Sentiment Analysis	**93.2**	91.3
CoLA	Linguistic Acceptability	35.0	**45.4**
GLUE	Multi Task Benchmark	68.9	**72.8**

Query what you want to look for

Key what you can compare to

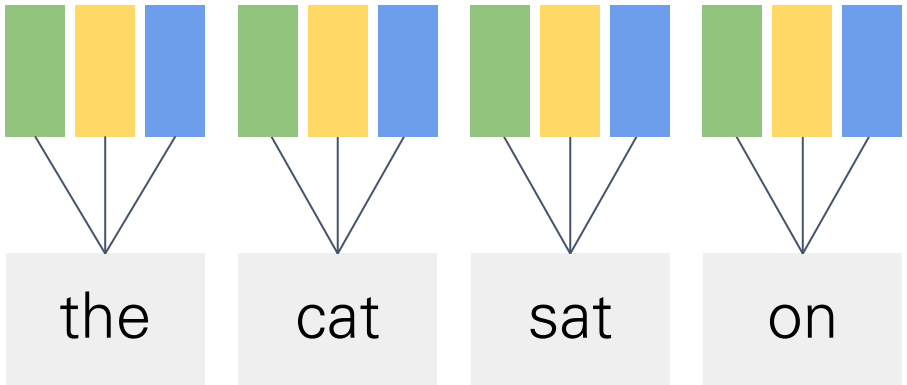
Value information you can retrieve

the	cat	sat	on

Query what you want to look for

Key what you can compare to

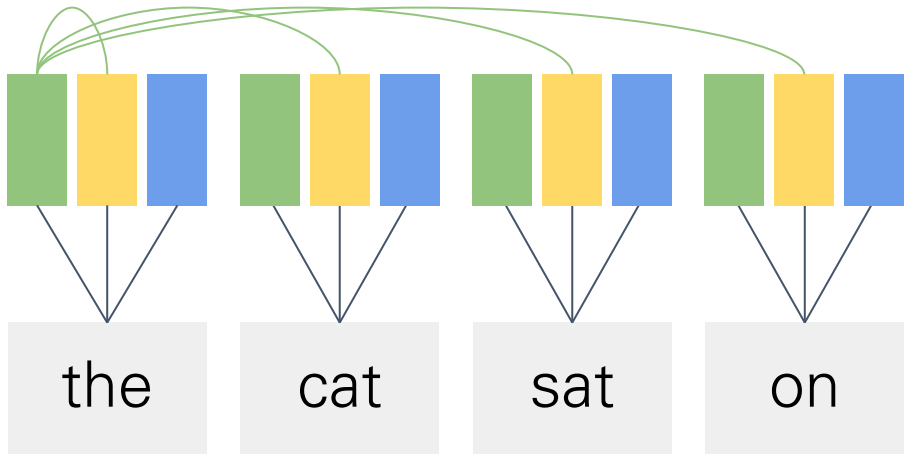
Value information you can retrieve



Query what you want to look for

Key what you can compare to

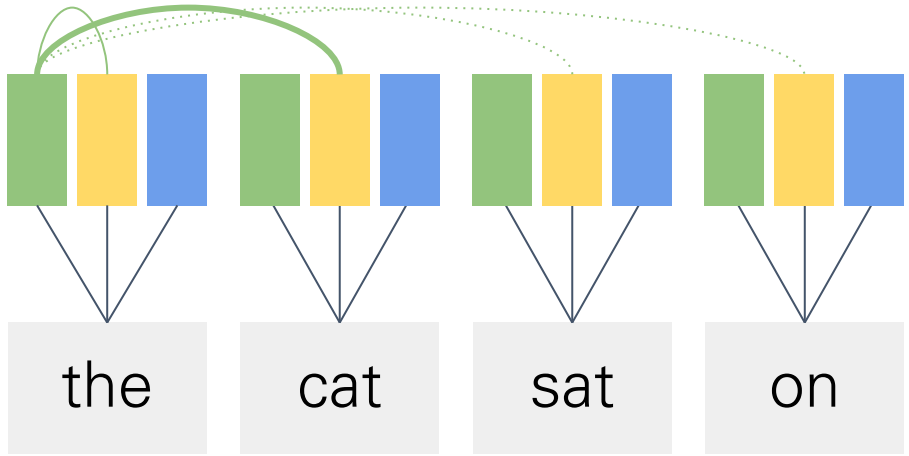
Value information you can retrieve



Query what you want to look for

Key what you can compare to

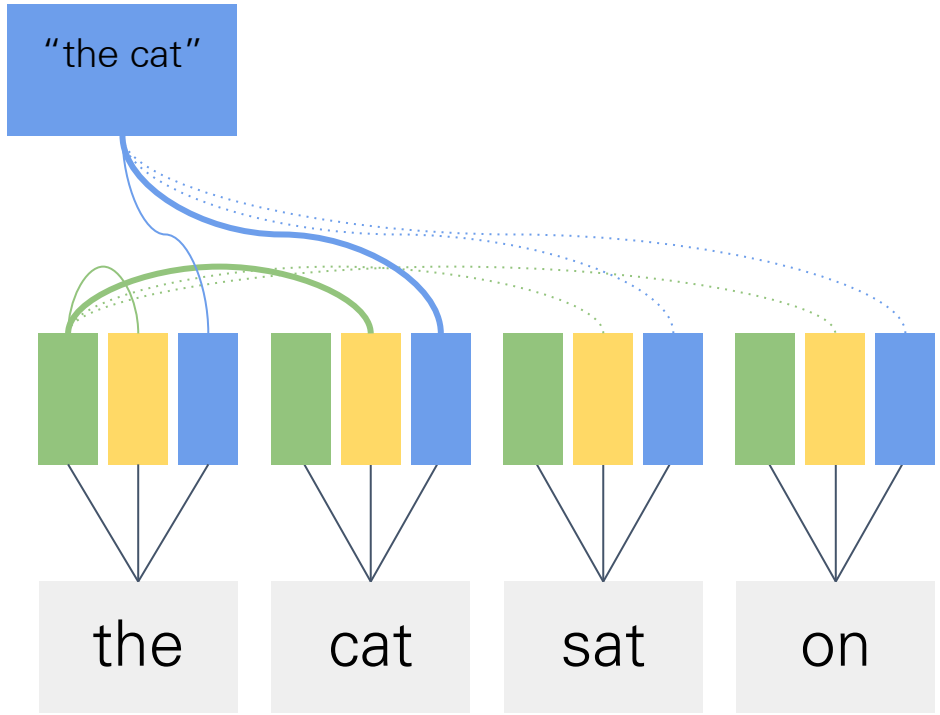
Value information you can retrieve

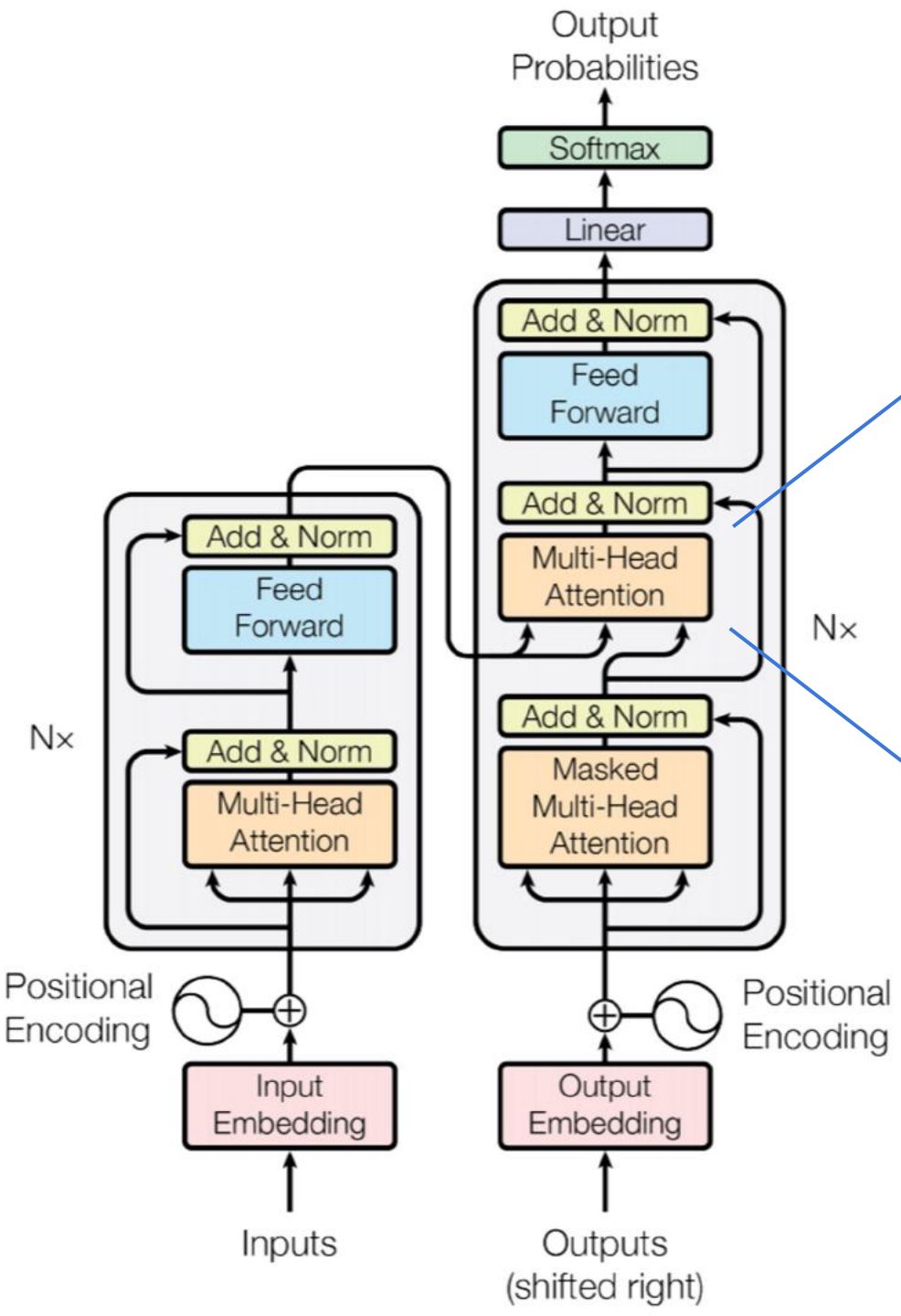


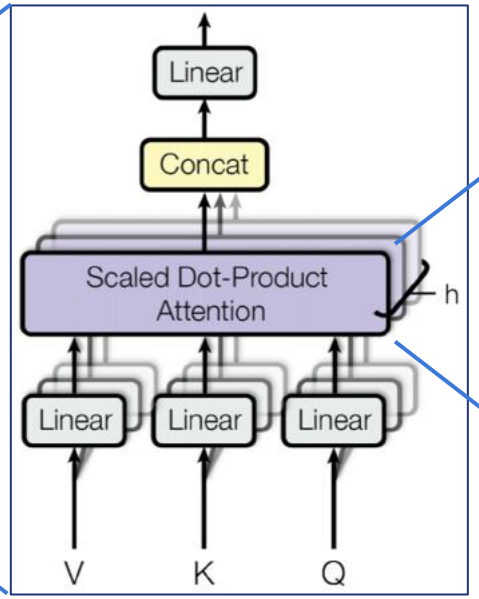
Query what you want to look for

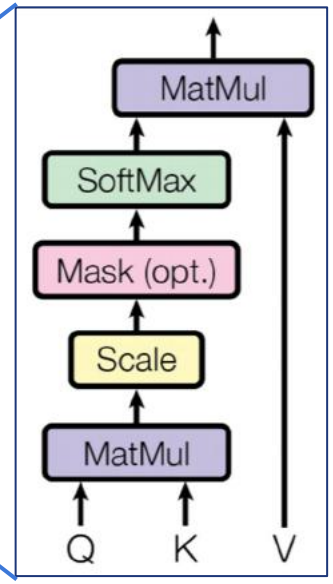
Key what you can compare to

Value information you can retrieve

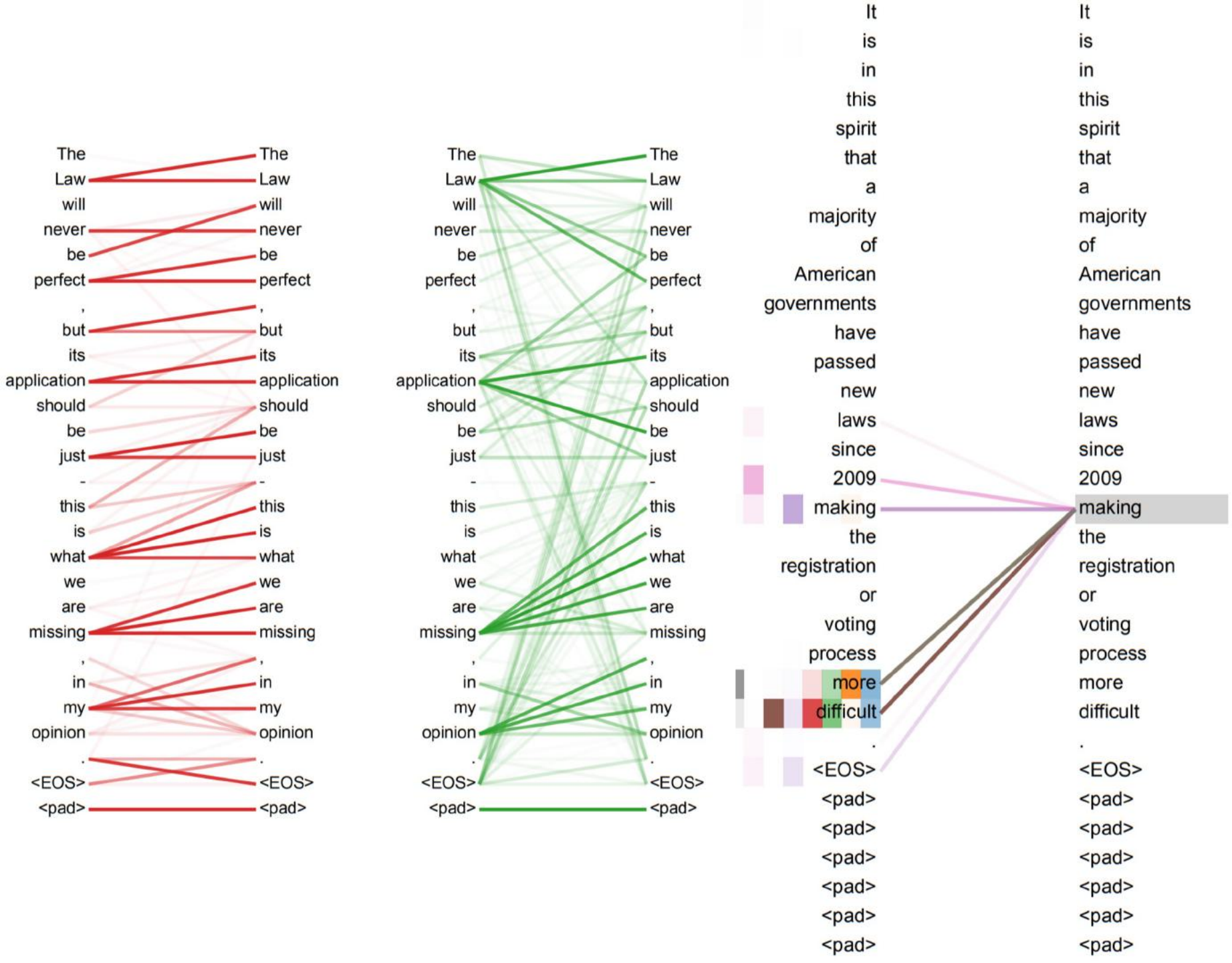


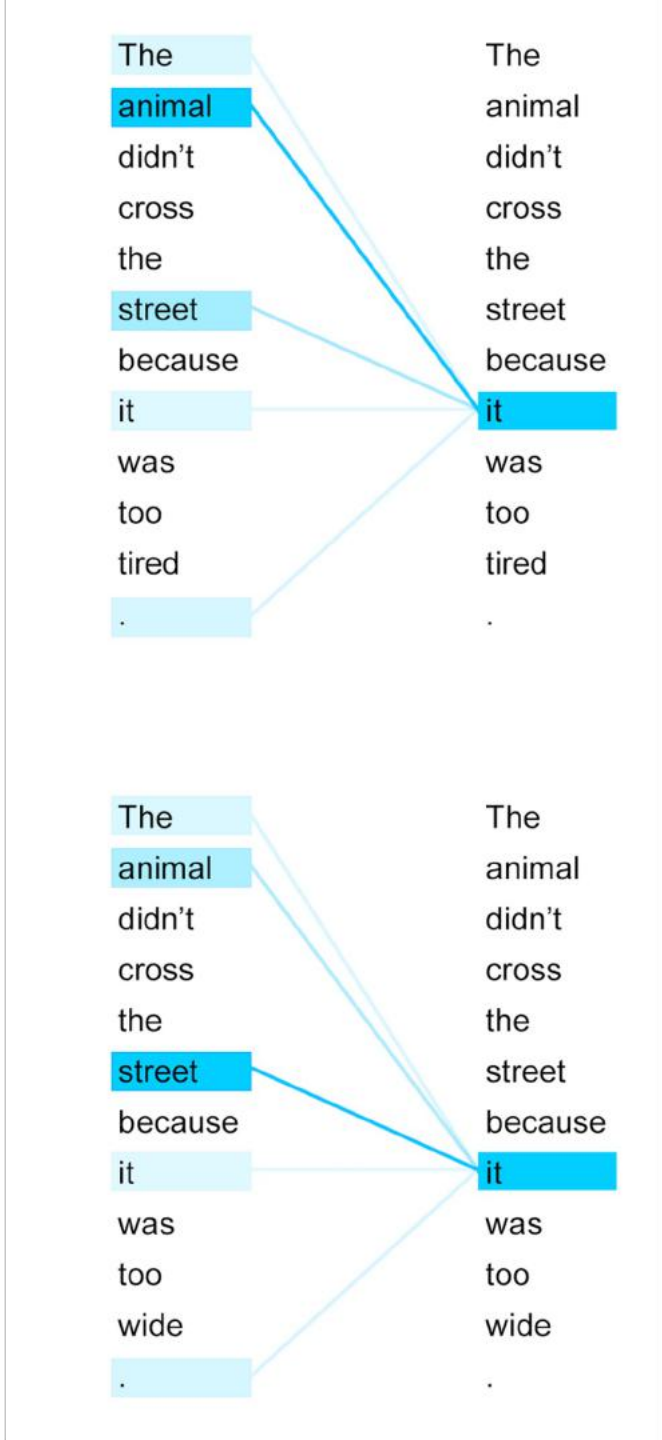






[Vaswani et al 2017]


The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad> <pad>
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad> <pad>


The animal didn't cross the street because it was too tired .
The animal didn't cross the street because it was too wide .

BERT

- First released in Oct 2018.
- NAACL'19: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- How is BERT different from ELMo?
  1. Unidirectional context vs bidirectional context
  2. LSTMs vs Transformers
  3. The weights are not freezed (fine-tuning)

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).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

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.

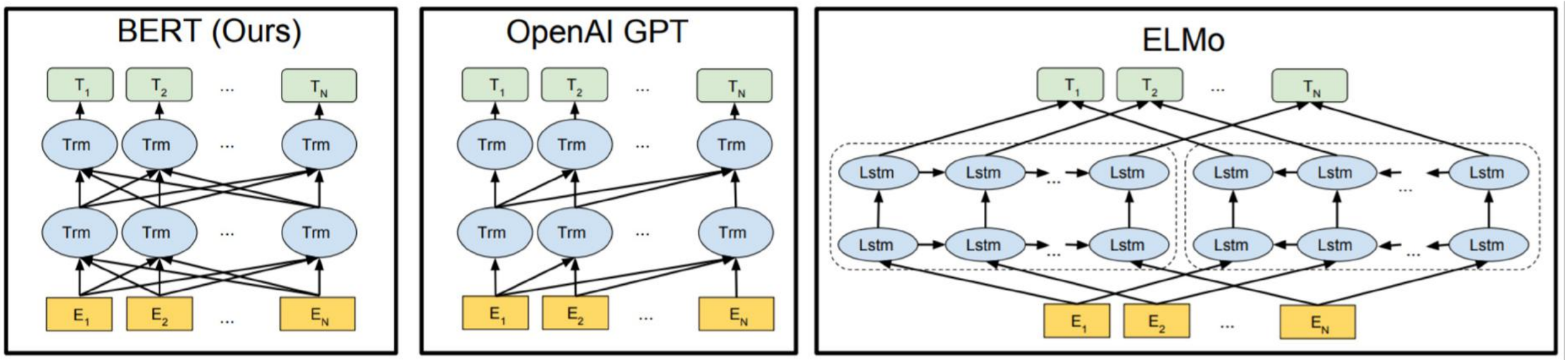
We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

Proceedings of NAACL-HLT 2019, pages 4171–4186
Minneapolis, Minnesota, June 2 - June 7, 2019.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Devlin et al. 2017

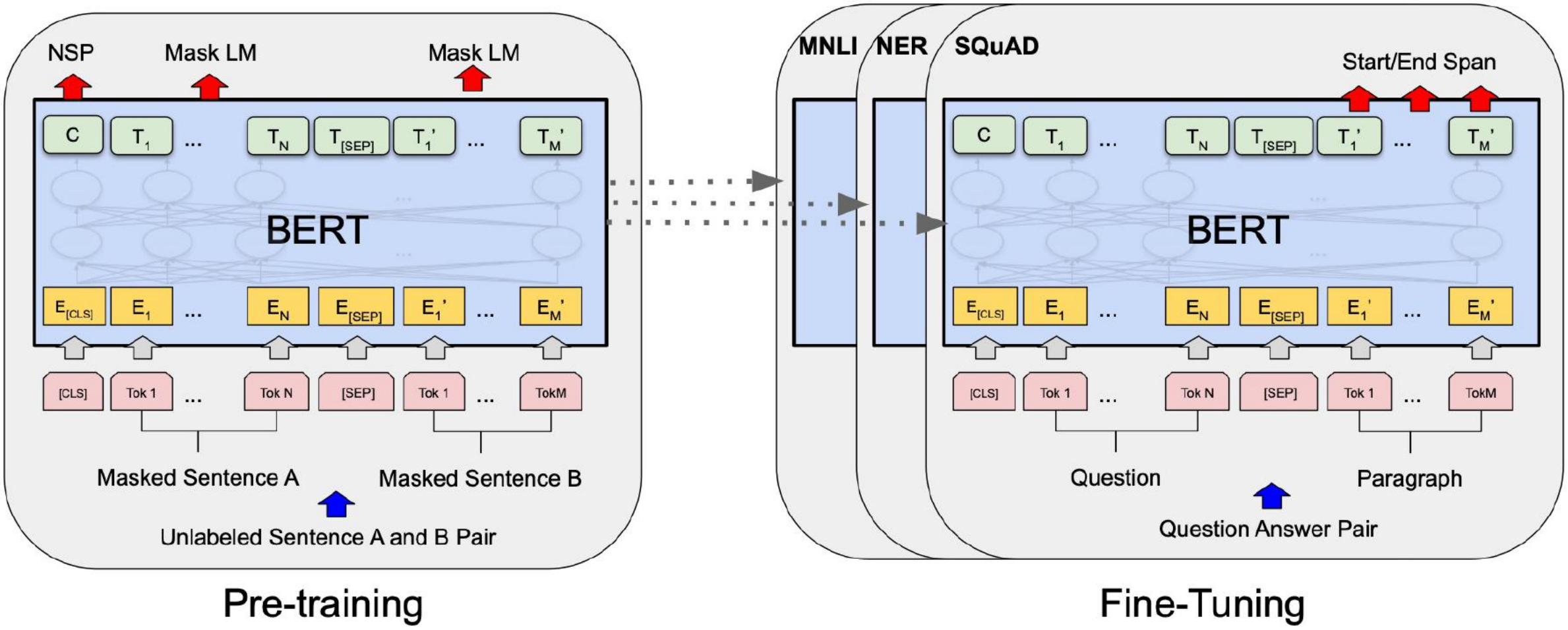



```
Left-Right LM: The cat sat on the [mask] -> The cat sat on the mat
Right-Left LM: [mask] cat sat on the mat -> The cat sat on the mat
Masked LM:  The [mask] sat on the [mask] -> The cat sat on the mat
```

BERT Workflow

- The BERT workflow includes:
  - Pretrain on generic, self-supervised tasks, using large amounts of data (like all of Wikipedia)
  - Fine-tune on specific tasks with limited, labelled data.
- The pretraining tasks (will talk about this in more detail later):
  - Masked Language Modelling (to learn contextualized token representations)
  - Next Sentence Prediction (summary vector for the whole input)

BERT Architecture



BERT Architecture

Properties:

- Two input sequences.
  - Many NLP tasks have two inputs (question answering, paraphrase detection, entailment detection etc.)
- Computes embeddings
  - Both token, position and segment embeddings.
  - Special start and separation tokens.
- Architecture
  - Basically the same as transformer encoder.
- Outputs:
  - Contextualized token representations.
  - Special tokens for context.

BERT Embeddings

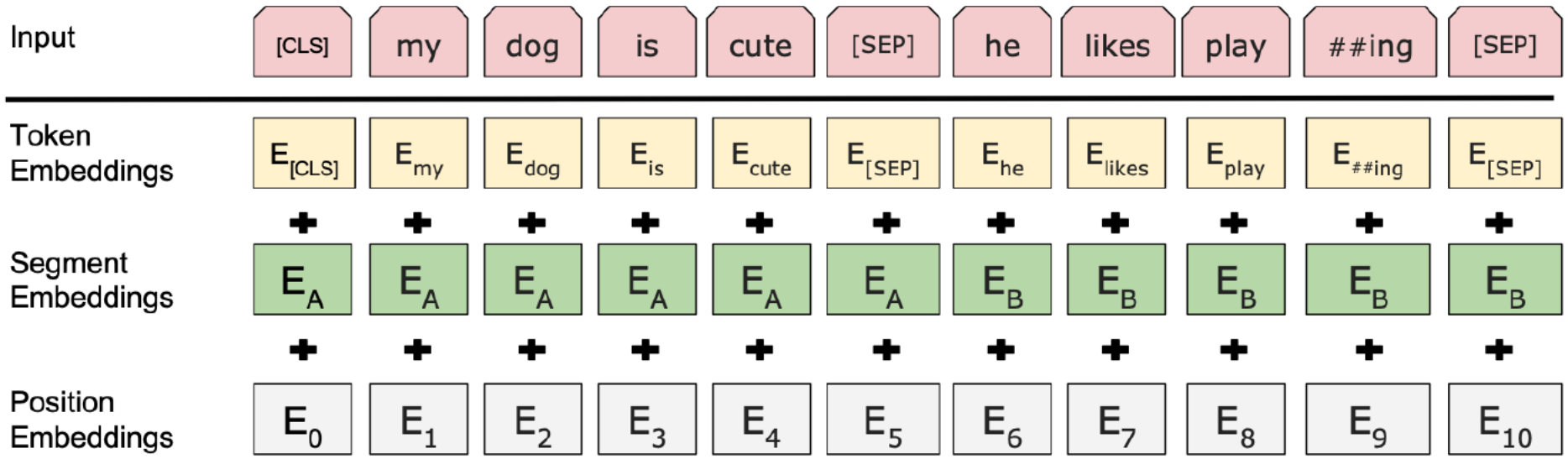


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- How we tokenize the inputs is very important!
- BERT uses the WordPiece tokenizer (Wu et. al. 2016)

(Aside) Tokenizers

- Tokenizers have to balance the following:
  - Being comprehensive (rare words? translation to different languages)
  - Total number of tokens
  - How semantically meaningful each token is.
- This is an activate area of research.

Pretraining tasks

- Masked Language Modelling, i.e. Cloze Task (Taylor, 1953)
- Next sentence prediction

Masked Language Modelling

- Mask 15% of the input tokens. (i.e. replace with a dummy masking token)
- Run the model, obtain the embeddings for the masked tokens.
- Using these embeddings, try to predict the missing token.
- ”I love to eat peanut ___ and jam. ” Can you guess what’s missing?

This procedure forces the model to encode context information in the features of all of the tokens.

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

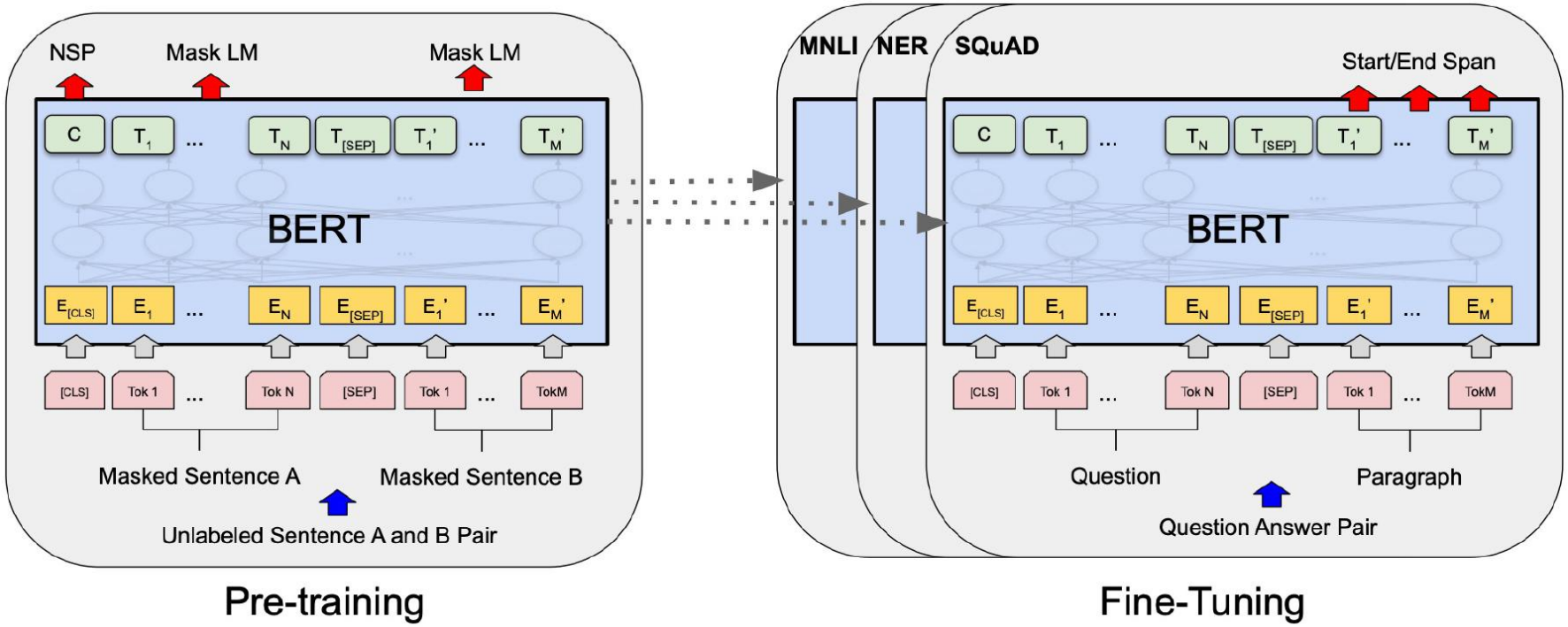
Next Sentence Prediction

- Goal is to summarize the complete context (i.e. the two segments) in a single feature vector.
- Procedure for generating data
  - Pick a sentence from the training corpus and feed it as "segment A".
  - With 50% probability, pick the following sentence and feed that as "segment B".
  - With 50% probability, pick the a random sentence and feed it as "segment B".
- Using the features for the context token, predict whether segment B is the following sentence of segment A.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

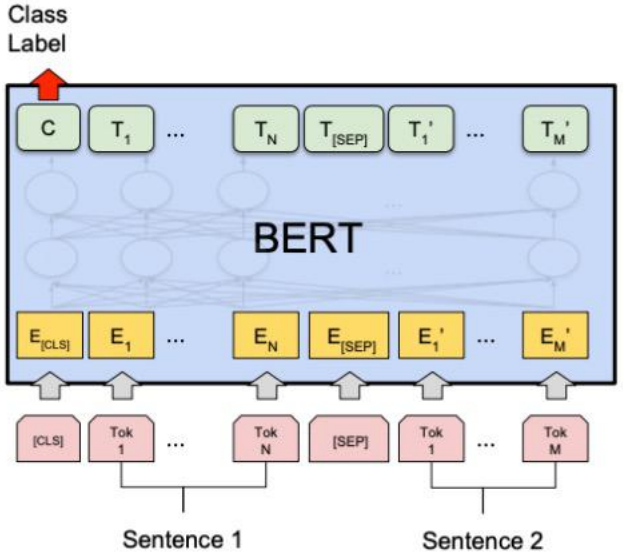
Fine Tuning



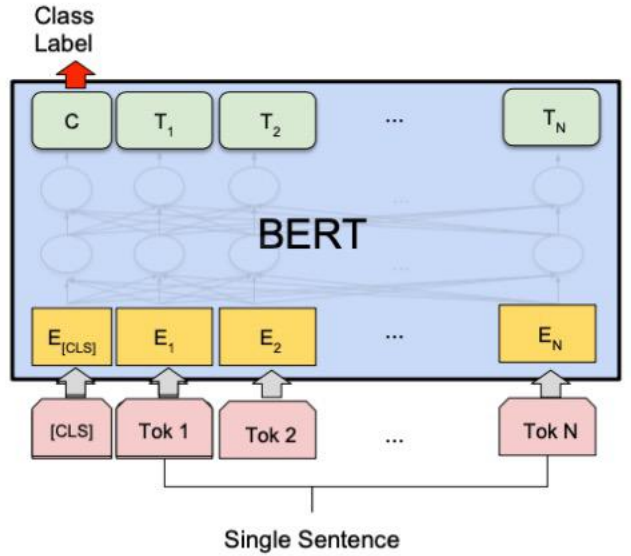
Procedure:

- Add a final layer on top of BERT representations.
- Train the whole network on the fine-tuning dataset.
- Pre-training time: In the order of days on TPUs.
- Fine tuning task: Takes only a few hours max.

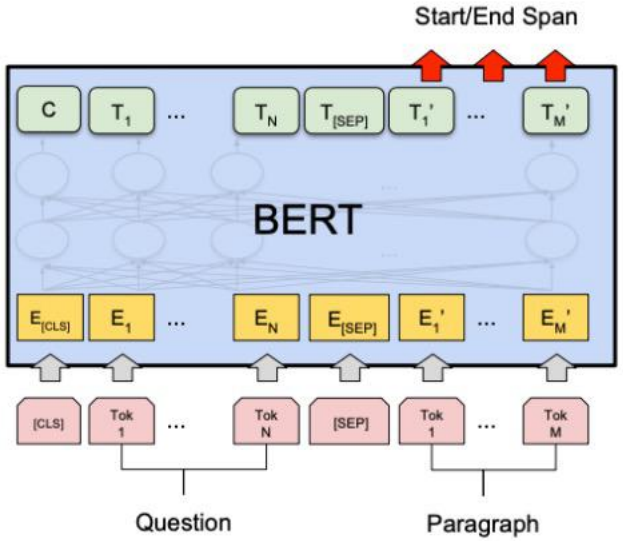
Applications



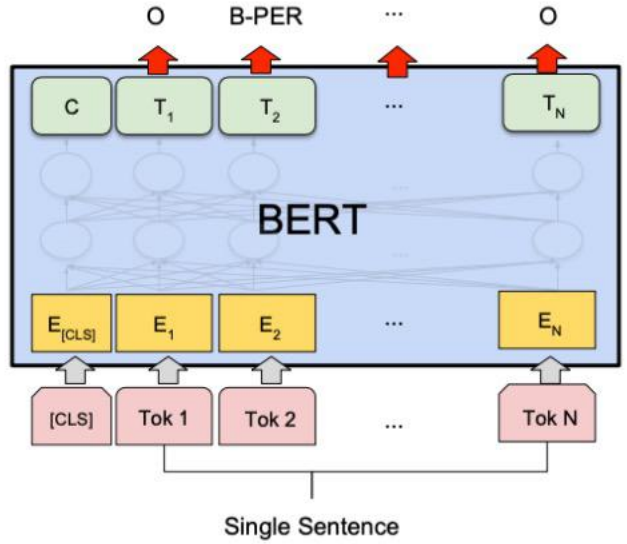
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

Fine Tuning

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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

RoBERTa: A Robustly Optimized BERT Pretraining Approach

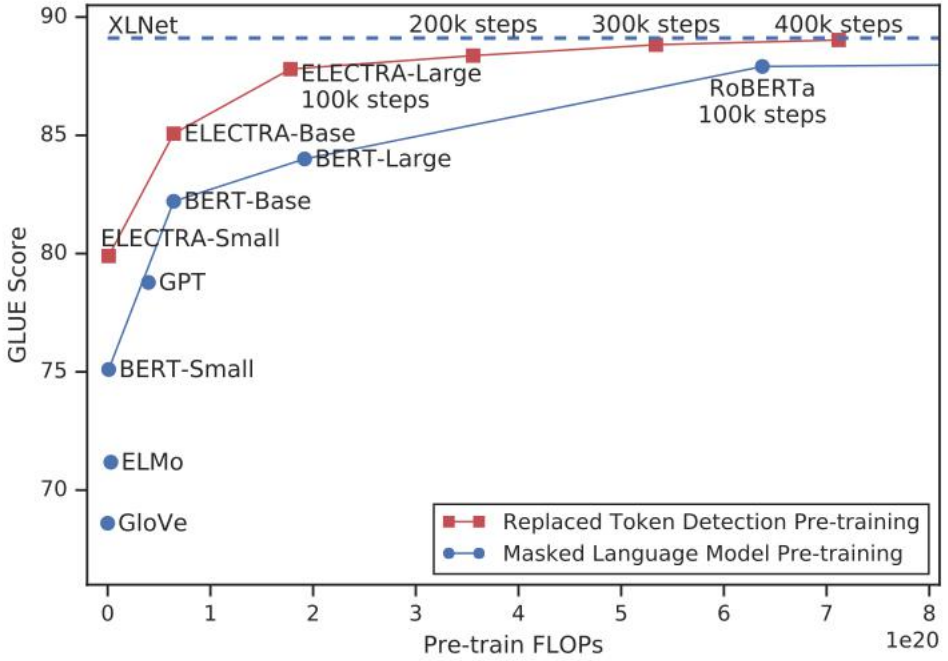
Liu et al. 2019

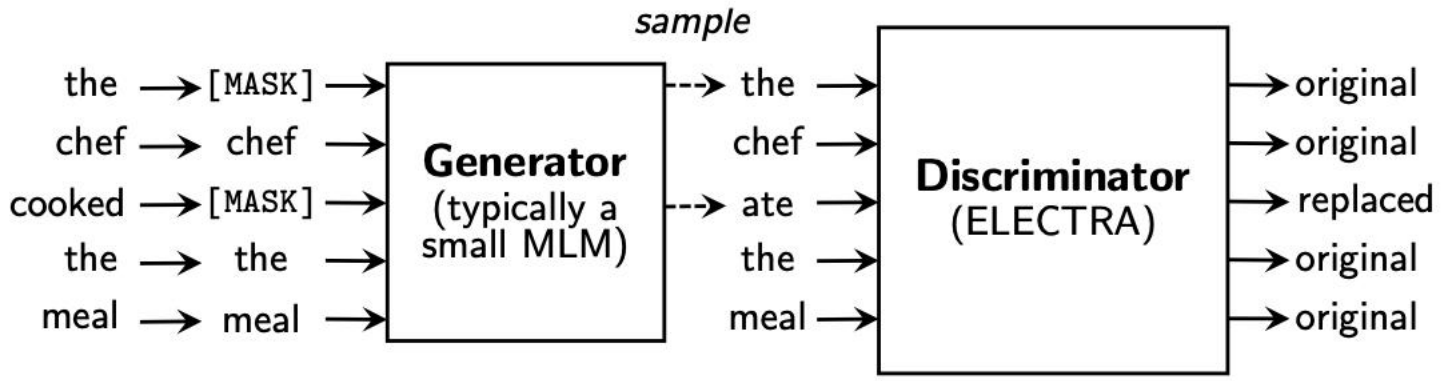
Really well executed refinement / engineering on BERT

- Better tuned (many HPs)
- Remove a few hacks (remove annealing context size)
- Better data generation (online instead of cached)
- A more flexible vocab scheme (more on this later)
- Use more compute / train longer (but same model capacity – BERT was undertrained)

ELECTRA

Clark et al. 2017





Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

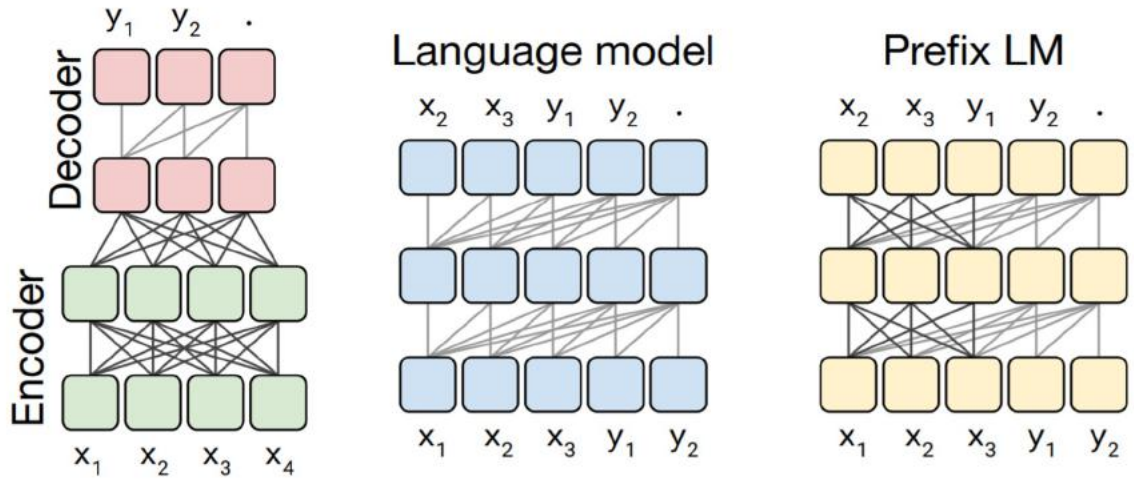
T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

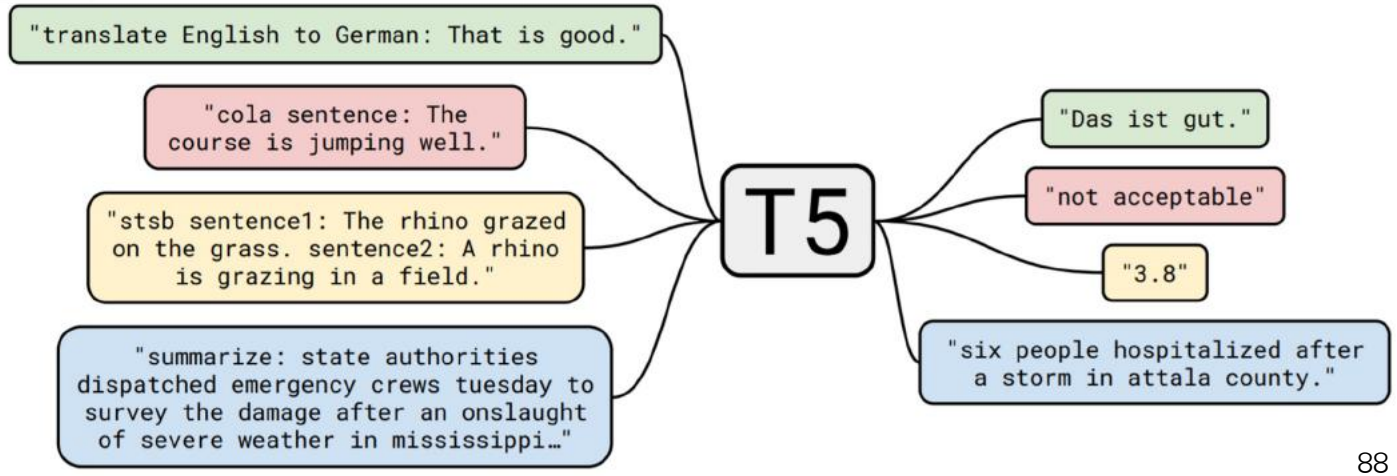
Raffel et al. 2019

- Very thorough (50 pages!) exploration of the design space of pretraining with a pleasing task formulation (from McCann et al 2018)

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
I.i.d. noise, mask tokens	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

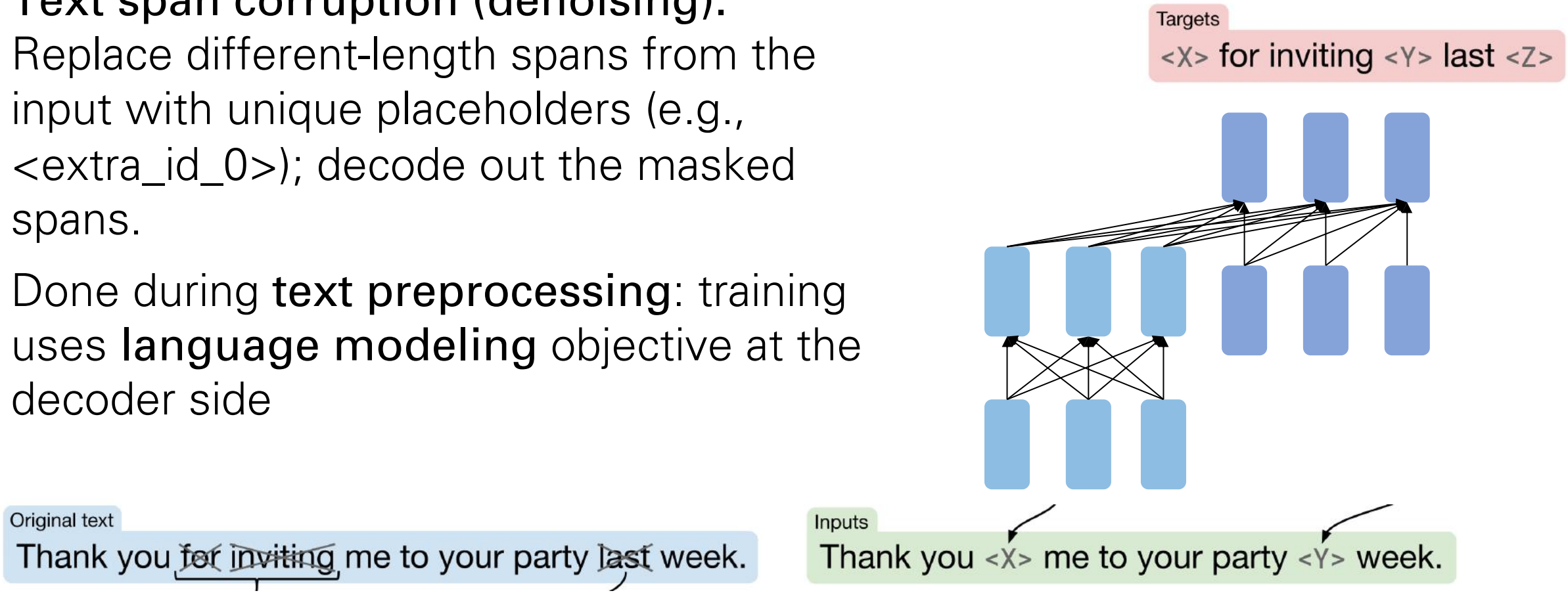
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76





T5

- T5 [Raffel et al., 2018]
- **Text span corruption (denoising):** Replace different-length spans from the input with unique placeholders (e.g., <extra_id_0>); decode out the masked spans.
- Done during **text preprocessing**: training uses **language modeling** objective at the decoder side



T5

- Encoder-decoders works better than decoders
- Span corruption (denoising) objective works better than language modeling

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

T5

- **Text-to-Text:** convert NLP tasks into input/output text sequences
- **Dataset:** Colossal Clean Crawled Corpus (C4), 750G text data!
- **Various Sized Models:**
  - Base (222M)
  - Small (60M)
  - Large (770M)
  - 3B
  - 11B

- Achieved SOTA with scaling & purity of data

Three Pre-training Paradigms/Architectures

Encoder

- E.g., BERT, RoBERTa, DeBERTa, …
- **Autoencoder** model
- **Masked** language modeling

Encoder-Decoder

- E.g., T5, BART, …
- **seq2seq** model

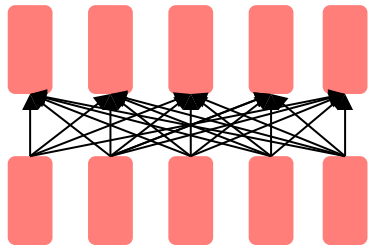
92

Decoder

- E.g., GPT, GPT2, GPT3, …
- **Autoregressive** model
- **Left-to-right** language modeling

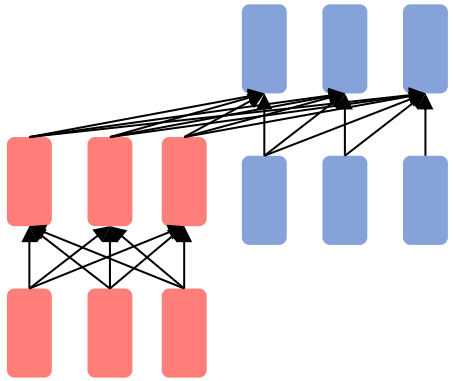
Three Pre-training Paradigms/Architectures

Encoder



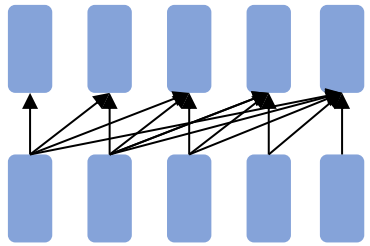
Bidirectional; can condition on the future context

Encoder-Decoder



Map two sequences of different length together

Decoder

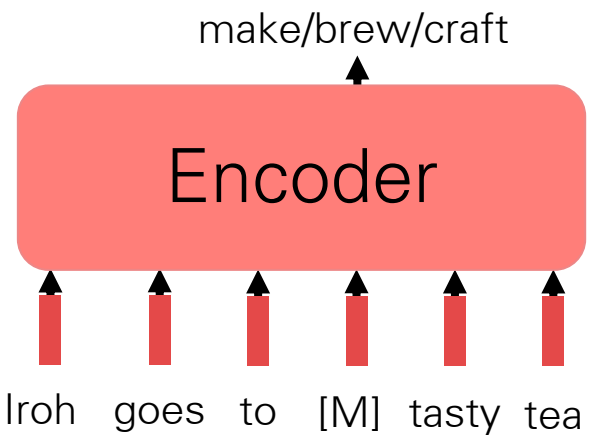


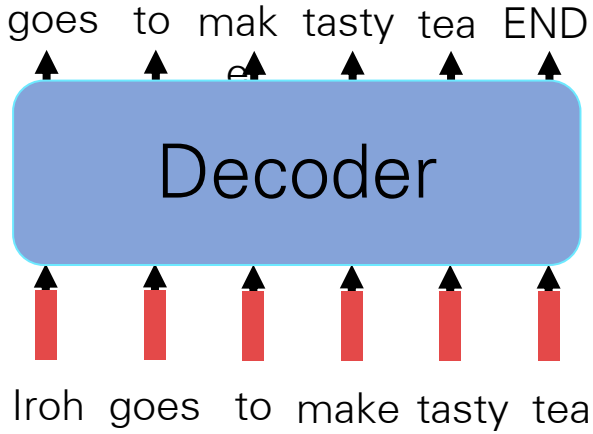
Language modeling; can only condition on the past context

Encoder: Pros & Cons

- Consider both left and right context
- Capture intricate contextual relationships

- Not good at generating open-text from left-to-right, one token at a time





Encoder-Decoder: Pros & Cons

- A nice middle ground between leveraging **bidirectional** contexts and **open-text** generation
- Good for **multi-task** fine-tuning

- Require more **text wrangling**
- **Harder to train**
- **Less flexible** for natural language generation

Decoder: Pros & Cons

- Natural to be used for open-text generation
- Right now decoder-only models seem to dominant the field at the moment
- e.g., GPT1/2/3/4, Mistral, Llama1/2

- Not better at feature extraction??

Next lecture:
Large Language Models