Lecture 13 Contextual Word Representations and Pretraining

Lecture Plan

1. Reflections on word representations
2. Pre-ELMo and ELMO
3. ULMfit and onward
4. Transformer architectures (20 mins)
5. BERT

The remaining lectures

- Transformers
- BERT
- Question answering
- Text generation and summarization
- “New research, latest updates in the field”
- “Successful applications of NLP in industry today”
- “More neural architectures that are used to solve NLP problem”
- “More linguistics stuff and NLU!”

1. Representations for a word

现在我们可以获得一个单词的表示

- 我们开始时学过的单词向量 Word2vec, GloVe, fastText

Pre-trained word vectors: The early years
Collobert, Weston, et al. 2011 results

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	**96.37**	**81.47**
Unsupervised pre-training followed by supervised NN**	**97.20**	**88.87**
+ hand-crafted features***	97.29	89.59

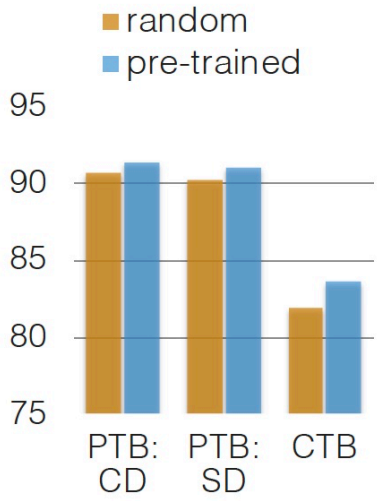
* Representative systems: POS: (Toutanova et al. 2003), NER: (Ando & Zhang 2005)

** 130,000-word embedding trained on Wikipedia and Reuters with 11 word window, 100 unit hidden layer – **for 7 weeks!** – then supervised task training

*** Features are character suffixes for POS and a gazetteer for NER

Pre-trained word vectors: Current sense (2014–)

- We can just start with random word vectors and train them on our task of interest
- But in most cases, use of pre-trained word vectors helps, because we can train them for **more words** on **much more data**



- Chen and Manning (2014) Dependency parsing
- Random: uniform(-0.01, 0.01)
- Pre-trained:
  - PTB (C & W): +0.7%
  - CTB (word2vec): +1.7%

Tips for unknown words with word vectors

- 简单且常见的解决方案

- 训练时：词汇表 Vocab 为 $\{ \text{words occurring, say, } \geq 5 \text{ times} \} \cup \{< \cup NK >\}$
- 将所有罕见的词（数据集中出现次数小于 5）都映射为 $< \text{UNK} >$，为其训练一个词向量
- 运行时：使用 $< \text{UNK} >$ 代替词汇表之外的词 OOV
- 问题
  - 没有办法区分不同UNK话说,无论是身份还是意义
- 解决方案
  1. 使用字符级模型学习期词向量
     - 特别是在 QA 中，match on word identity 是很重要的,即使词向量词汇表以外的单词
  2. Try these tips (from Dhingra, Liu, Salakhutdinov, Cohen 2017)
     - 如果测试时的 $< \text{UNK} >$ 单词不在你的词汇表中，但是出现在你使用的无监督词嵌入中，测试时直接使用这个向量
     - 此外，你可以将其视为新的单词，并为其分配一个随机向量，将它们添加到你的词汇表。
- 帮助很大 或者 也许能帮点忙
- 你可以试试另一件事
  - 将它们分解为词类（如未知号码，大写等等），每种都对应一个 $< \text{UNK-class} >$

Representations for a word

存在两个大问题

- 对于一个 **word type** 总是是用相同的表示，不考虑这个 **word token** 出现的上下文
  - 比如 star 这个单词，有天文学上的含义以及娱乐圈中的含义
  - 我们可以进行非常细粒度的词义消歧
- 我们对一个词只有一种表示，但是单词有不同的方面，包括语义，句法行为，以及表达 / 含义
  - 表达：同样的意思可以是用多个单词表示，他们的词义是一样的

Did we all along have a solution to this problem?

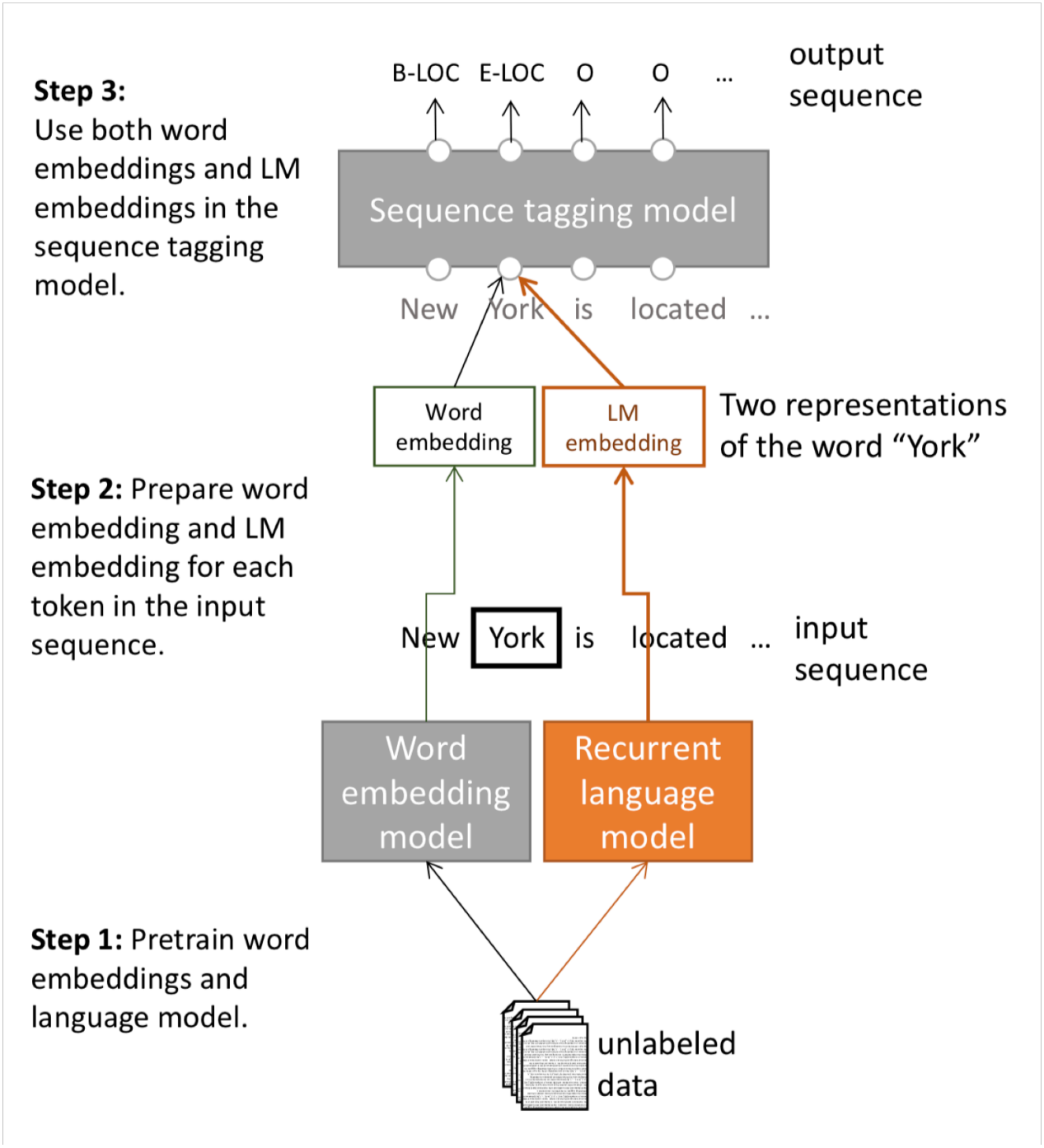
- 在模型中，我们通过LSTM层(也许只在语料库上训练)
- 那些LSTM层被训练来预测下一个单词
- 但这些语言模型在每一个位置生成特定于上下文的词表示

2. Peters et al. (2017): TagLM – “Pre-ELMo”

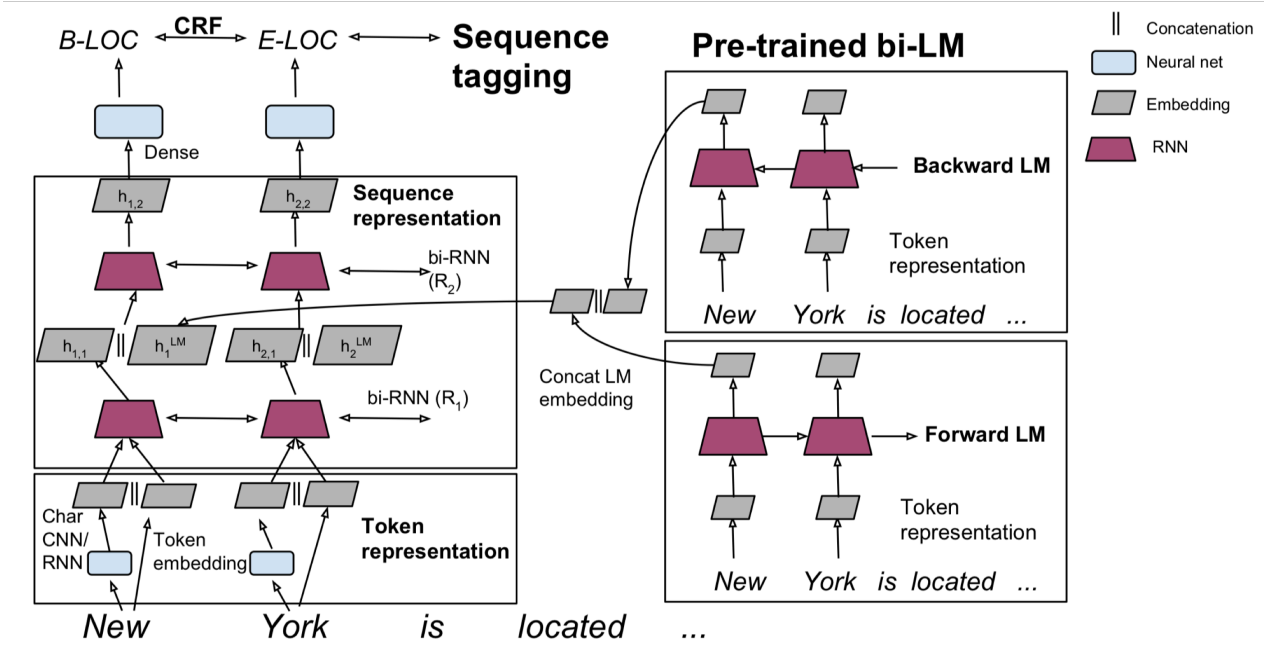
https://arxiv.org/pdf/1705.00108.pdf

- 想法：想要获得单词在上下文的意思，但标准的 RNN 学习任务只在 task-labeled 的小数据上（如 NER ）
- 为什么不通过半监督学习的方式在大型无标签数据集上训练 NLM，而不只是词向量

Tag LM



- 与上文无关的单词嵌入 + RNN model 得到的 hidden states 作为特征输入

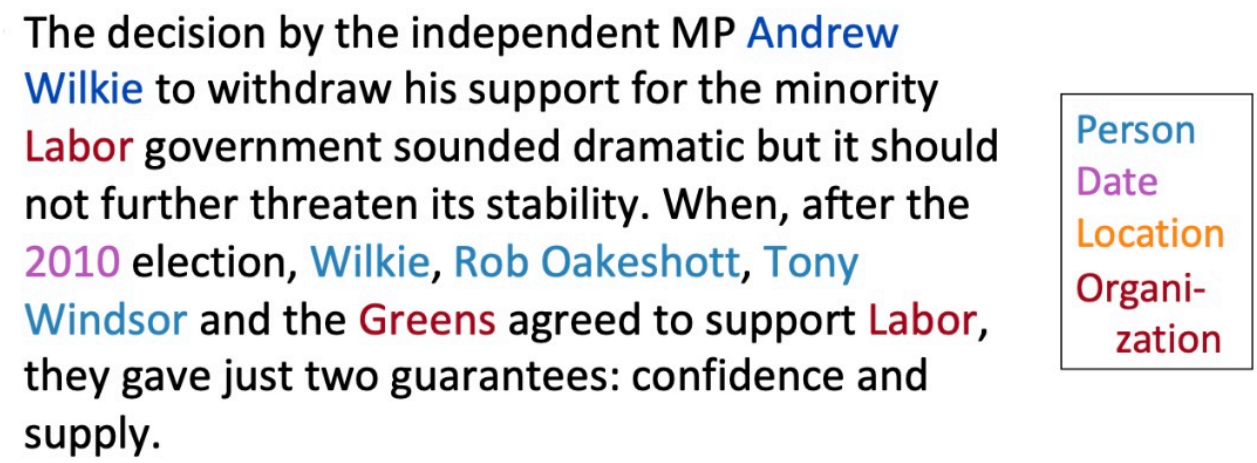


$$\mathbf{h}_{k,l} = \left[\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}\right] \tag{1}$$

- Char CNN / RNN + Token Embedding 作为 bi-LSTM 的输入
- 得到的 hidden states 与 Pre-trained bi-LM（冻结的） 的 hidden states 连接起来输入到第二层的 bi-LSTM 中

Named Entity Recognition (NER)

- 一个非常重要的NLP子任务：查找和分类文本中的实体，例如



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

65

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

语言模型在"Billion word benchmark"的8亿个训练单词上训练

语言模型观察结果

- 在监督数据集上训练的LM并不会受益
- 双向LM仅有助于 forward 过程，提升约0.2
- 具有巨大的 LM 设计（困惑度 30） 比较小的模型（困惑度 48）提升约0.3

> 困惑度

任务特定的BiLSTM观察结果

- 仅使用LM嵌入来预测并不是很好：88.17 F1
  - 远低于仅在标记数据上使用 BiLSTM 标记器

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

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

- 也有使用训练好的序列模型为其他NLP模型提供上下文的想法
- 想法：机器翻译是为了保存意思，所以这也许是个好目标?
- 使用seq2seq + attention NMT system中的Encoder，即 2层 bi-LSTM ，作为上下文提供者
- 所得到的 CoVe 向量在各种任务上都优于 GloVe 向量
- 但是，结果并不像其他幻灯片中描述的更简单的NLM培训那么好，所以似乎被放弃了
  - 也许NMT只是比语言建模更难?
  - 或许有一天这个想法会回来?

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

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

- **word token vectors** or **contextual word vectors** 的爆发版本
- 使用长上下文而不是上下文窗口学习 word token 向量(这里，整个句子可能更长)
- 学习深度Bi-NLM，并在预测中使用它的所有层
- 训练一个双向LM
- 目标是 performant 但LM不要太大
  - 使用2个biLSTM层
  - (仅)使用字符CNN构建初始单词表示
  - 2048 个 char n-gram filters 和 2 个 highway layers，512 维的 projection
  - 4096 dim hidden/cell LSTM状态，使用 512 dim的对下一个输入的投影
  - 使用残差连接
  - 绑定 token 的输入和输出的参数(softmax)，并将这些参数绑定到正向和反向LMs之间
- ELMo学习biLM表示的特定任务组合
- 这是一个创新，TagLM 中仅仅使用堆叠LSTM的顶层，ELMo 认为BiLSTM所有层都是有用的

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

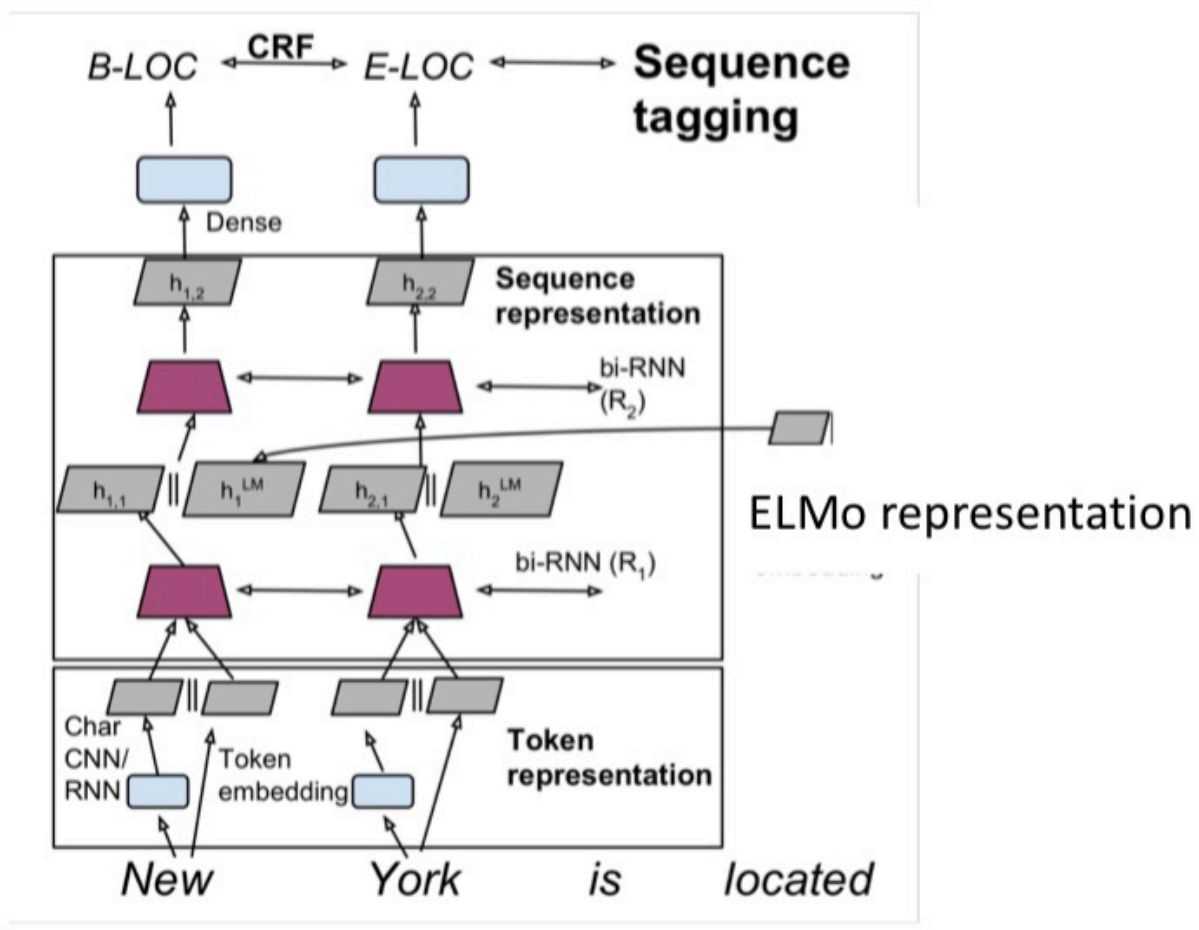
$$
\mathbf{ELMo}_k^{task} = E\left(R_k; \Theta^{task}\right) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

- γ^{task} 衡量ELMo对任务的总体有用性，是为特定任务学习的全局比例因子
- $\mathbf{s}^{task}$ 是 softmax 归一化的混合模型权重，是 BiLSTM 的加权平均值的权重，对不同的任务是不同的，因为不同的任务对不同层的 BiLSTM 的

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

- 首先运行 biLM 获取每个单词的表示
- 然后让(无论什么)最终任务模型使用它们
- 冻结ELMo的权重，用于监督模型
- 将ELMo权重连接到特定于任务的模型中
  - 细节取决于任务
    - 像 TagLM 一样连接到中间层是典型的
    - 可以在生产输出时提供更多的表示，例如在问答系统中

ELMo used in a sequence tagger



$$\mathbf{h}_{k,l} = \left[\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}\right] \tag{2}$$

ELMo results: Great for all 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%

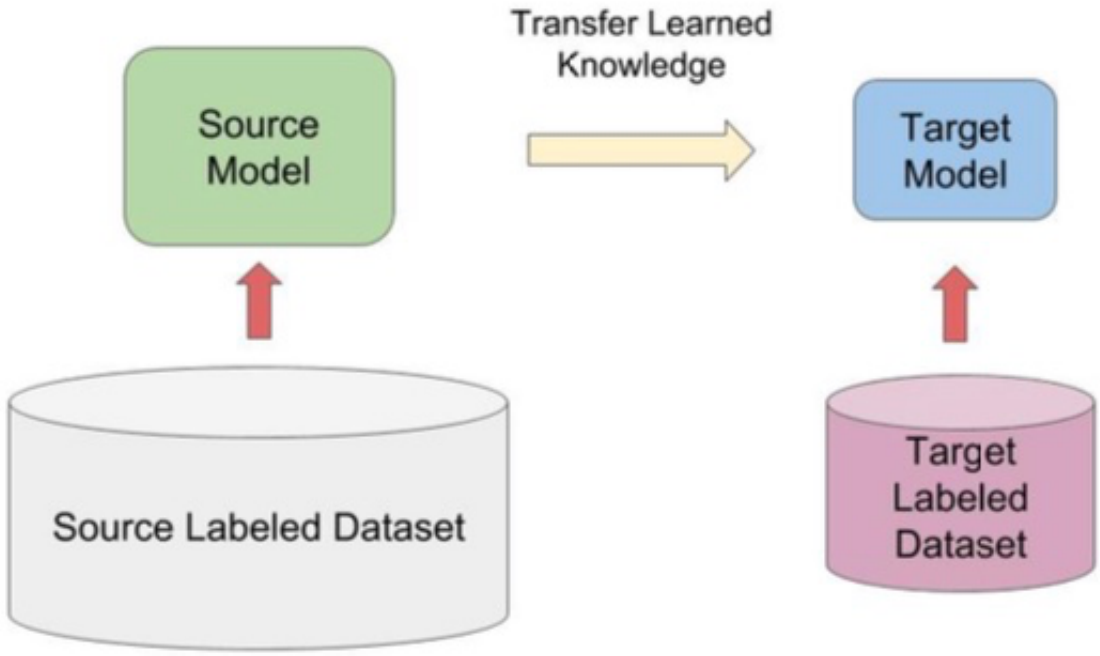
ELMo: Weighting of layers

- 这两个biLSTM NLM层有不同的用途/含义
  - 低层更适合低级语法，例如
    - 词性标注(part-of-speech tagging)、句法依赖(syntacticdependency)、NER
  - 高层更适合更高级别的语义
    - 情绪、Semantic role labeling 语义角色标记 、question answering、SNLI
- 这似乎很有趣，但它是如何通过两层以上的网络来实现的看起来更有趣

Also around: ULMfit

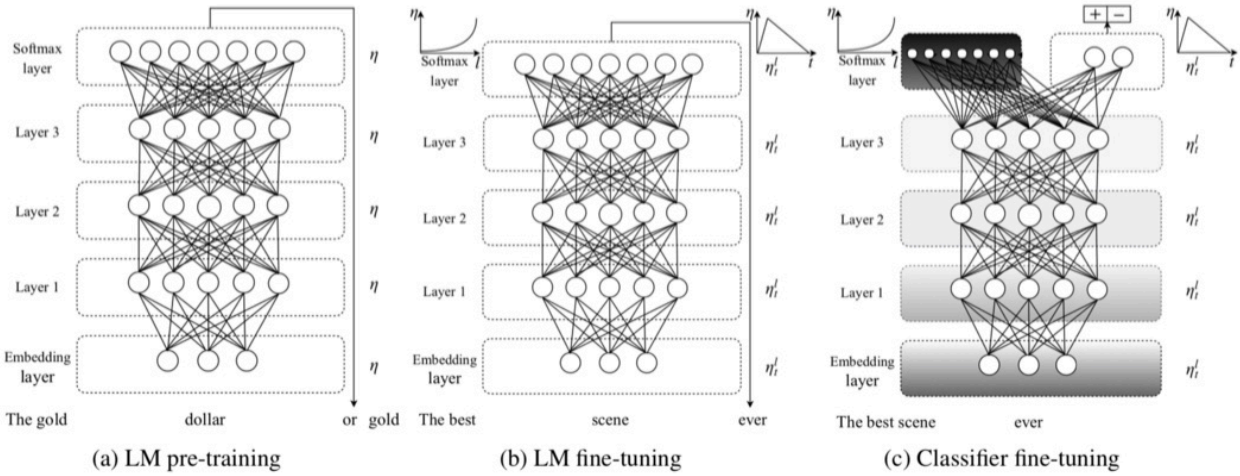
Howard and Ruder (2018) Universal Language Model Fine-tuning for Text Classification. https://arxiv.org/pdf/1801.06146.pdf

- 转移NLM知识的一般思路是一样的
- 这里应用于文本分类



3. ULMfit

- 在大型通用领域的无监督语料库上使用 biLM 训练
- 在目标任务数据上调整 LM
- 对特定任务将分类器进行微调



ULMfit emphases

- 使用合理大小的“1 GPU”语言模型，并不是真的很大
- 在LM调优中要注意很多
  - 不同的每层学习速度
  - 倾斜三角形学习率(STLR)计划

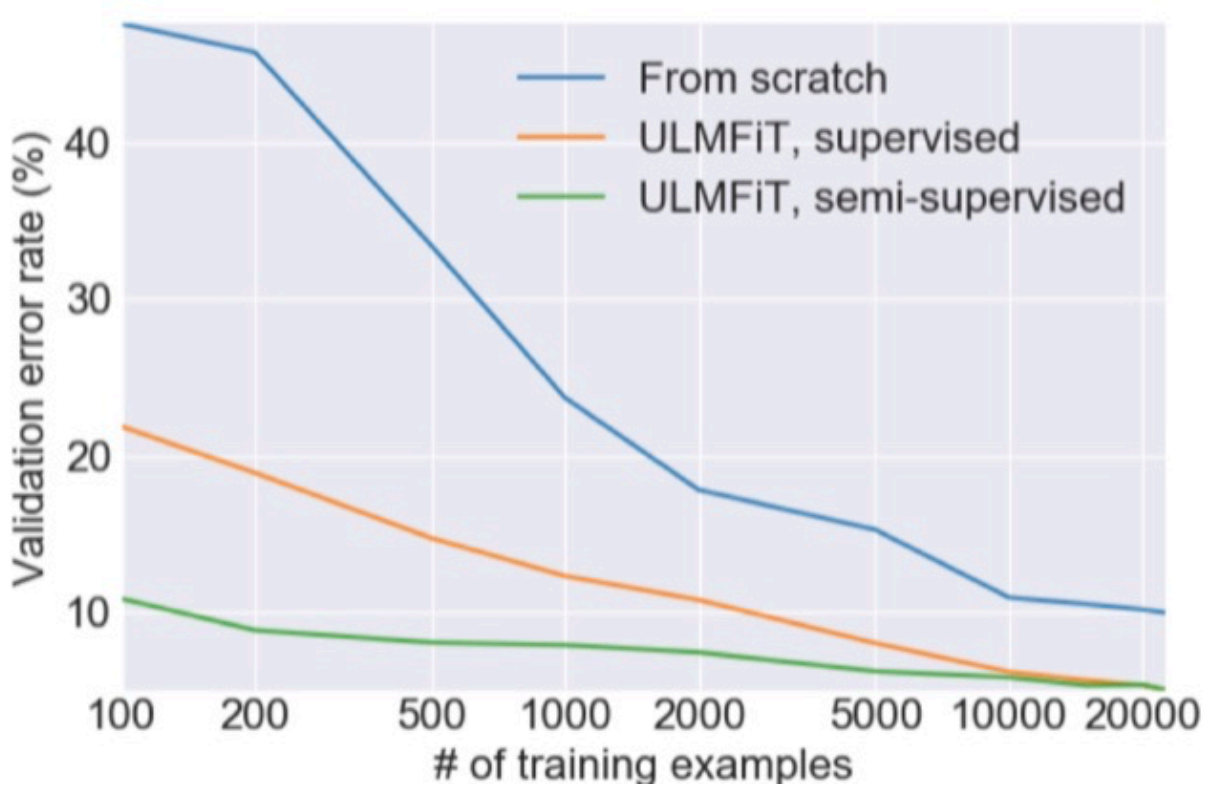
    - 学习分类器时逐步分层解冻和STLR
    - 使用$[h_T, \text{maxpool}(\mathbf{h}), \text{meanpool}(\mathbf{h})]$进行分类
- 使用大型的预训练语言模型是一种提高性能的非常有效的方法

ULMfit performance

- 文本分类器错误率

	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



- 如果使用监督数据进行训练文本分类器，需要大量的数据才能学习好

Let's scale it up!

ULMfit	GPT	BERT	GPT-2
Jan 2018	June 2018	Oct 2018	Feb 2019
Training:	Training	Training	Training
1 GPU day	240 GPU days	256 TPU days ~320–560 GPU days	~2048 TPU v3 days according to a reddit thread

fast.ai OpenAI Google AI OpenAI

GPT-2 language model (cherry-picked) output

- 文本生成的样例

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

35

Transformer models

All of these models are Transformer architecture models … so maybe we had better learn about Transformers?

ULMfit	GPT	BERT	GPT-2
Jan 2018	June 2018	Oct 2018	Feb 2019
Training: 1 GPU day	Training 240 GPU days	Training 256 TPU days ~320–560 GPU days	Training ~2048 TPU v3 days according to a reddit thread

fast.ai

OpenAI

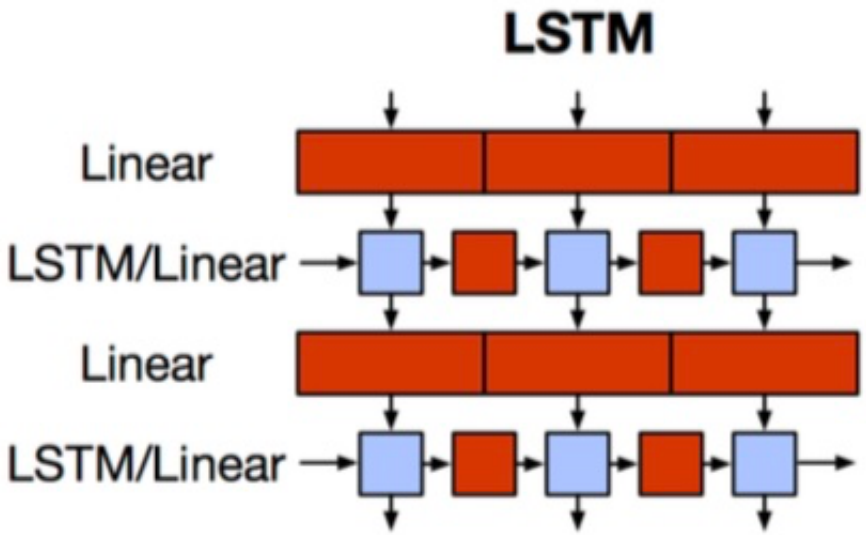
Google AI

OpenAI

36

- Transformer 不仅狠强大，而且允许扩展到更大的尺寸

4. The Motivation for Transformers



- 我们想要并行化，但是RNNs本质上是顺序的
- 尽管有GRUs和LSTMs, RNNs仍然需要注意机制来处理长期依赖关系——否则状态之间的 path length **路径长度** 会随着序列增长
- 但如果注意力让我们进入任何一个状态……也许我们可以只用注意力而不需要RNN?

Transformer Overview

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

- Non-recurrent sequence-to-sequence encoder-decoder model
- 任务：平行语料库的机器翻译
- 预测每个翻译单词
- 最终成本/误差函数是 softmax 分类器基础上的标准交叉熵误差

Transformer Basics

- 自学
  - 主要推荐资源
    - http://nlp.seas.harvard.edu/2018/04/03/attention.html
    - The Annotated Transformer by Sasha Rush
  - An Jupyter Notebook using PyTorch that explains everything!
- 现在：我们定义 Transformer 网络的基本构建块：第一，新的注意力层

Dot-Product Attention (Extending our previous def.)

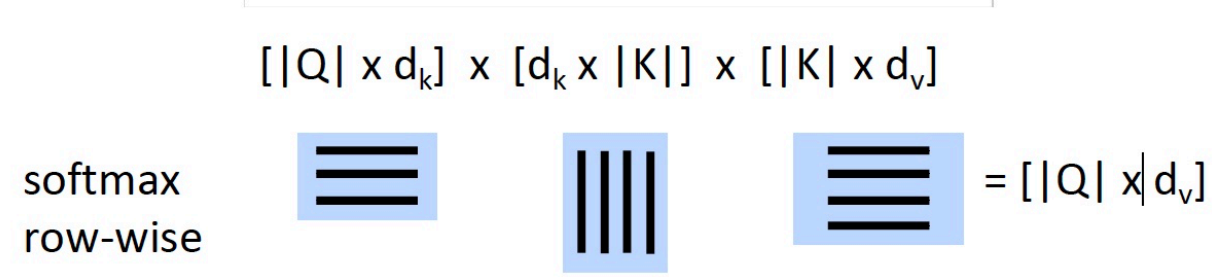
- 输入：对于一个输出而言的查询 q 和一组键-值对 k-v
- Query, keys, values, and output 都是向量
- 输出值的加权和
- 权重的每个值是由查询和相关键的内积计算结果
- Query 和 keys 有相同维数 d_k ， value 的维数为 d_v

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i \tag{3}$$

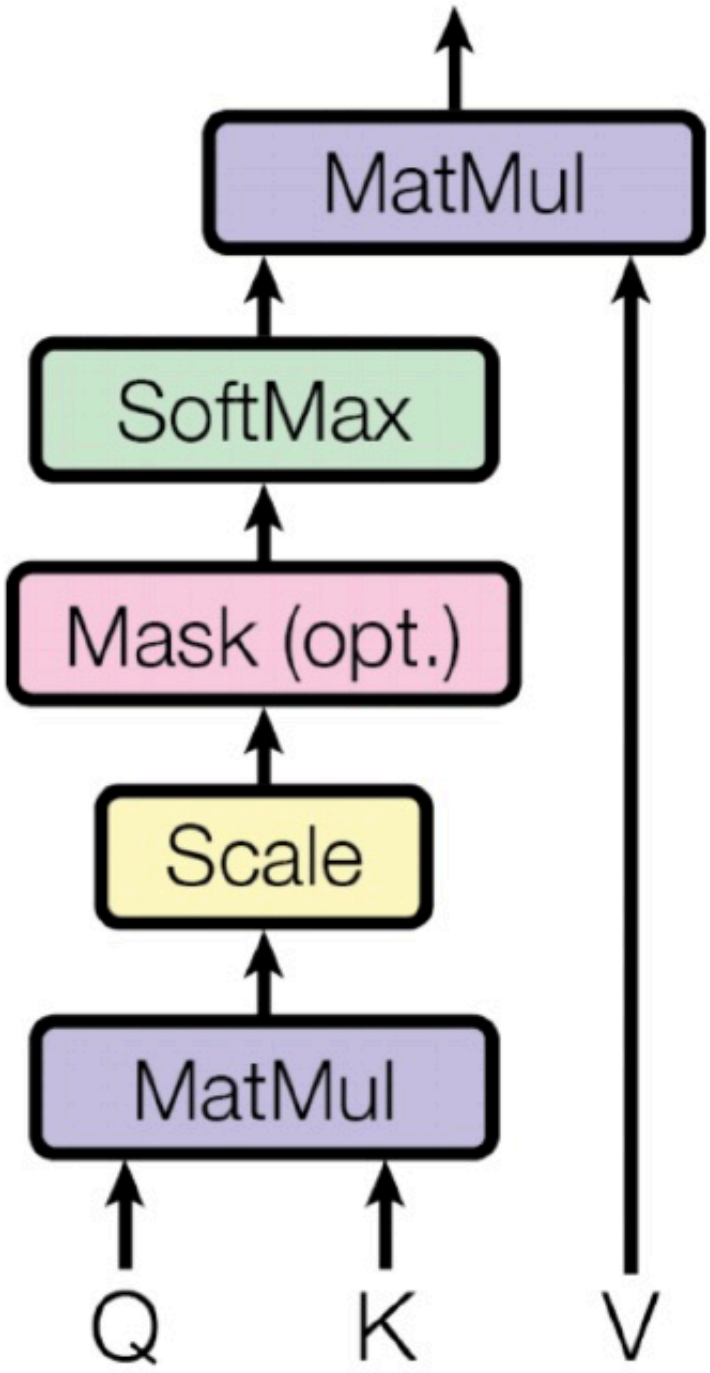
Dot-Product Attention – Matrix notation

- 当我们有多个查询 q 时，我们将它们叠加在一个矩阵 Q 中

$$A(Q, K, V) = \text{softmax}\left(QK^T\right)V \tag{4}$$



Scaled Dot-Product Attention



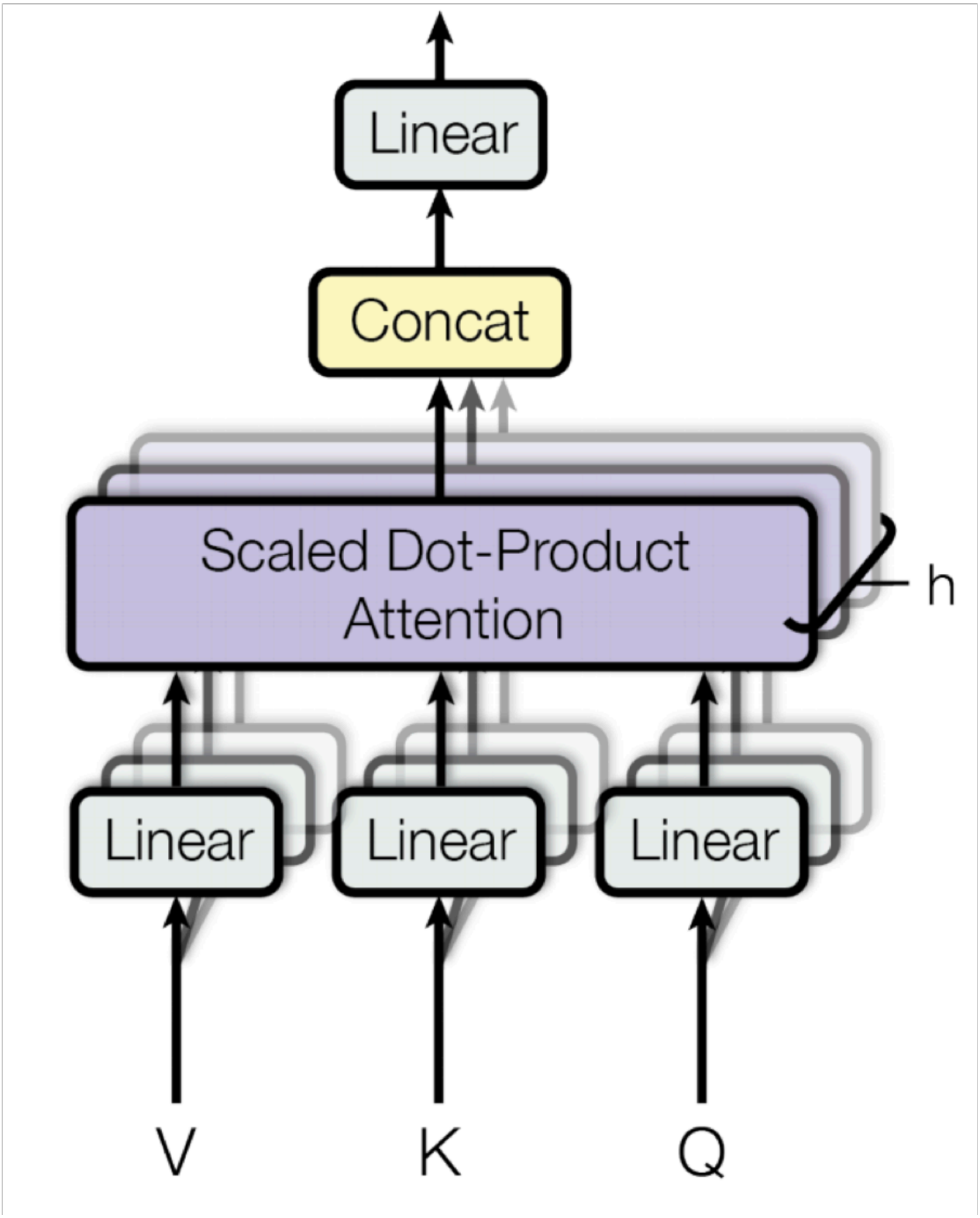
- 问题：d_k 变大时，$q^T k$ 的方差增大 $\rightarrow$ 一些 softmax 中的值的方差将会变大 $\rightarrow$ softmax 得到的是峰值 $\rightarrow$ 因此梯度变小了
- 解决方案：通过query/key向量的长度进行缩放

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

Self-attention in the encoder

- 输入单词向量是queries, keys and values
- 换句话说：这个词向量自己选择彼此
- 词向量堆栈= Q = K = V
- 我们会通过解码器明白为什么我们在定义中将他们分开

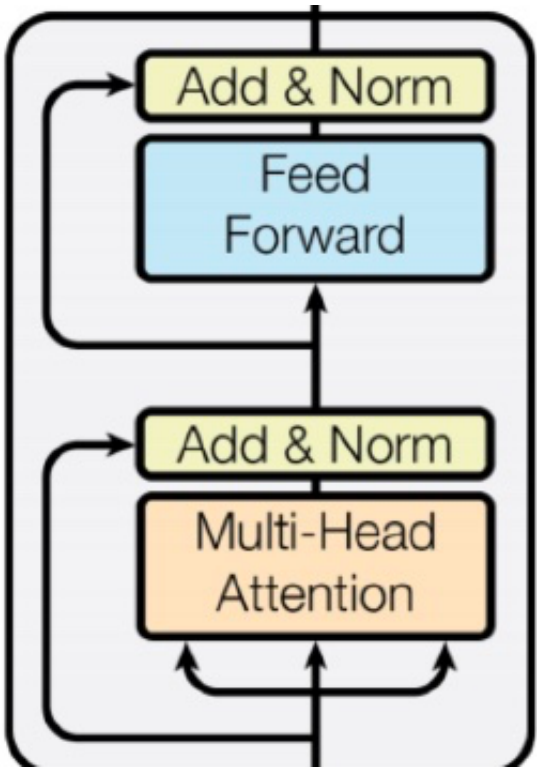
Multi-head attention



- 简单self-attention的问题
- 单词只有一种相互交互的方式
- 解决方案：多头注意力
- 首先通过矩阵 W 将 Q, K, V 映射到 h = 8 的许多低维空间
- 然后应用注意力，然后连接输出，通过线性层

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h) \\ \text{where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{6}$$

Complete transformer block



- 每个 block 都有两个“子层”
  1. 多头 attention
  2. 两层的前馈神经网络，使用 ReLU

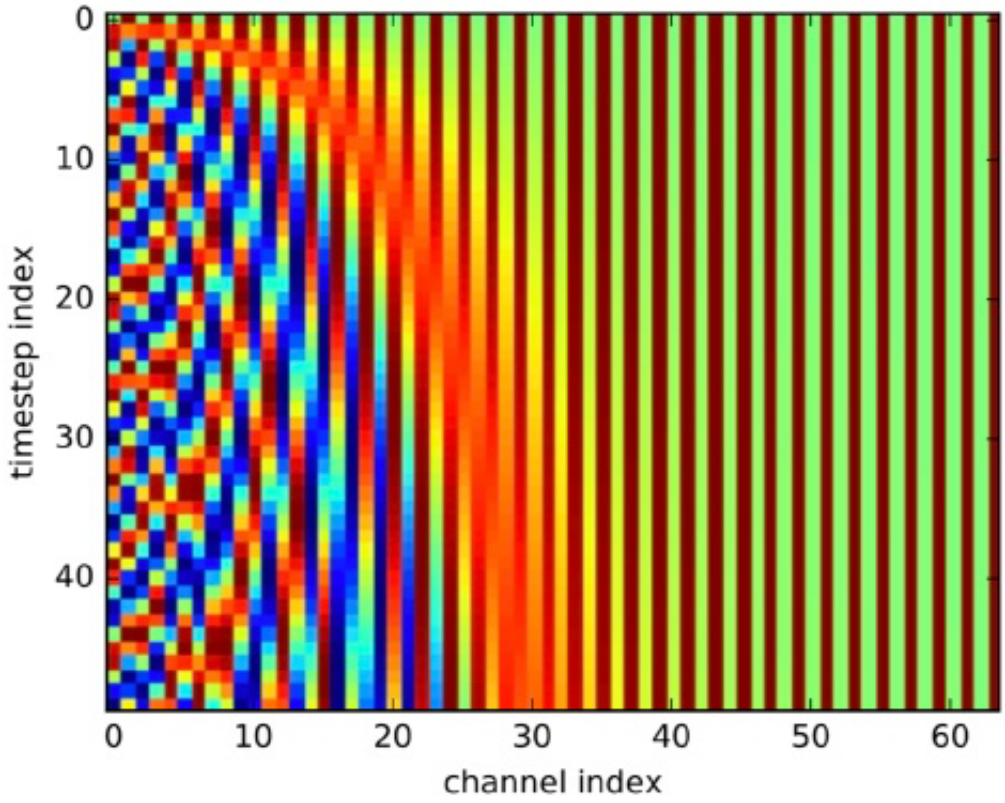
这两个子层都：

- 残差连接以及层归一化
  - LayerNorm(x+Sublayer(x))
  - 层归一化将输入转化为均值是 0，方差是 1，每一层和每一个训练点（并且添加了两个参数）

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \quad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \quad h_i = f\left(\frac{g_i}{\sigma_i}(a_i - \mu_i) + b_i\right) \tag{7}$$

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

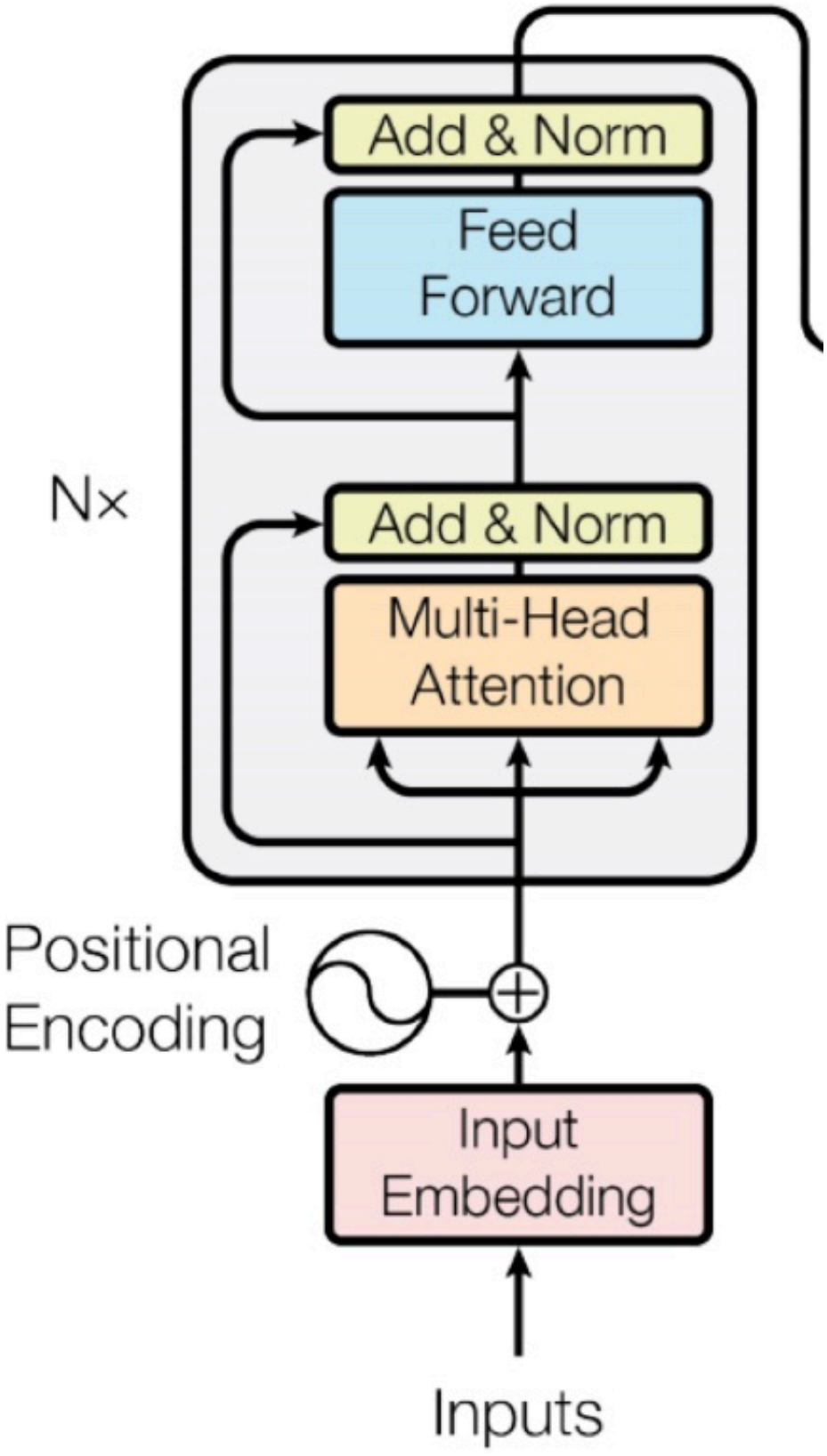
Encoder Input



- 实际的词表示是 byte-pair 编码
- 还添加了一个 **positional encoding** 位置编码，相同的词语在不同的位置有不同的整体表征

$$\begin{cases} PE(pos, 2i) = \sin\left(pos/10000^{2i/d_{model}}\right) \\ PE(\text{pos}, 2i+1) = \cos\left(pos/10000^{2i/d_{model}}\right) \end{cases} \tag{8}$$

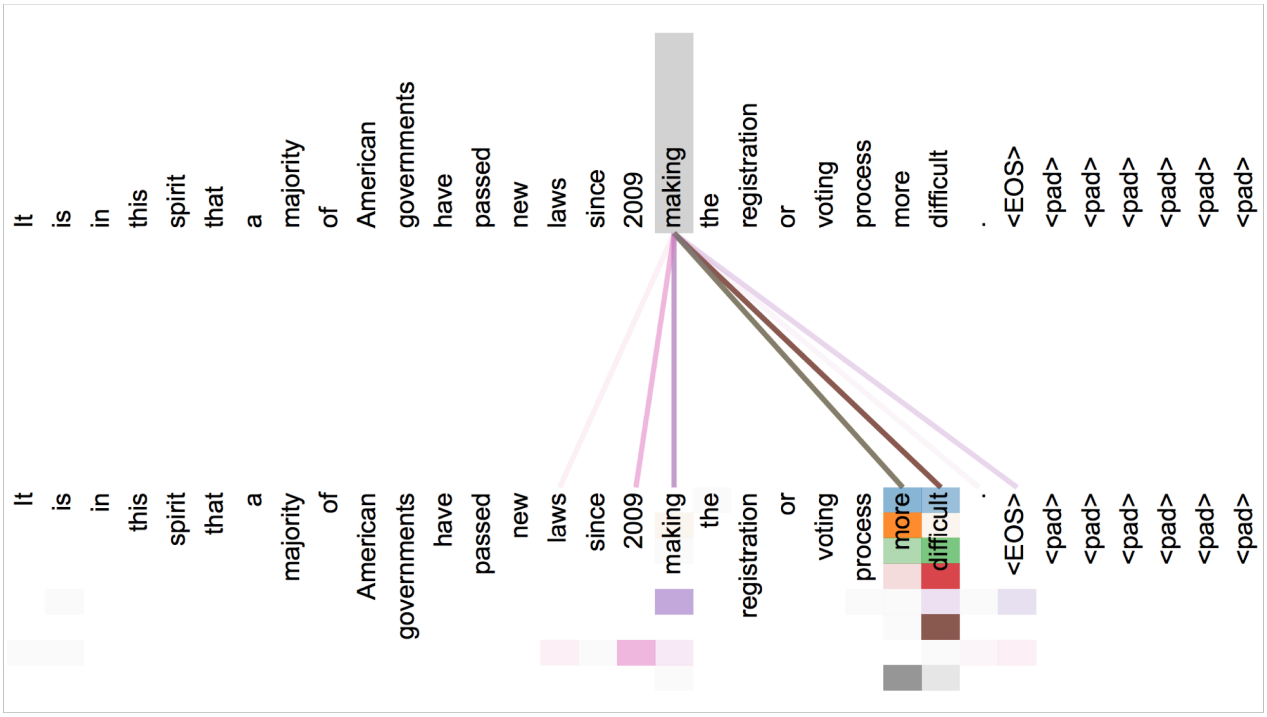
Complete Encoder



- encoder 中，每个 block 都是来自前一层的 Q, K, V
- Blocks 被重复 6 次（垂直方向）
- 在每个阶段，你可以通过多头注意力看到句子中的各个地方，累积信息并将其推送到下一层。在任一方向上的序列逐步推送信息来计算感兴趣的值
- 非常善于学习语言结构

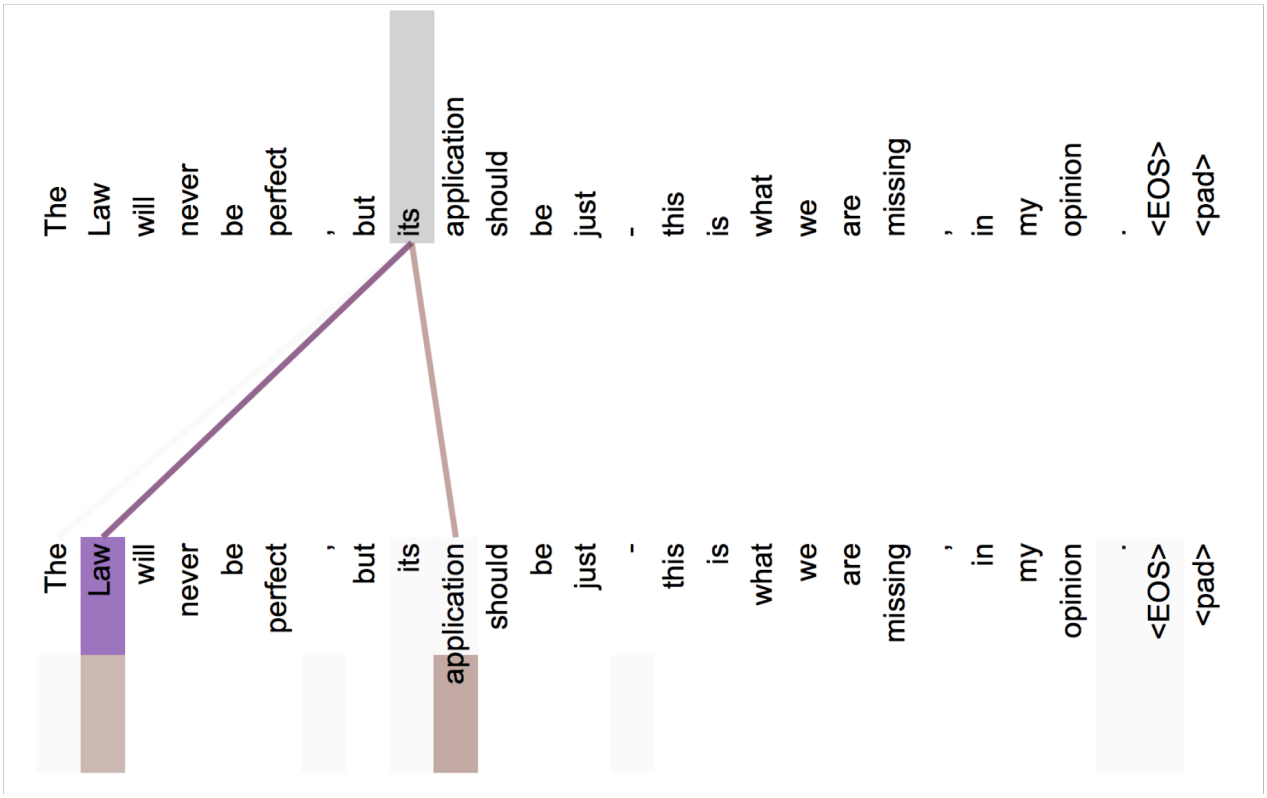
Attention visualization in layer 5

- 词语开始以合理的方式关注其他词语



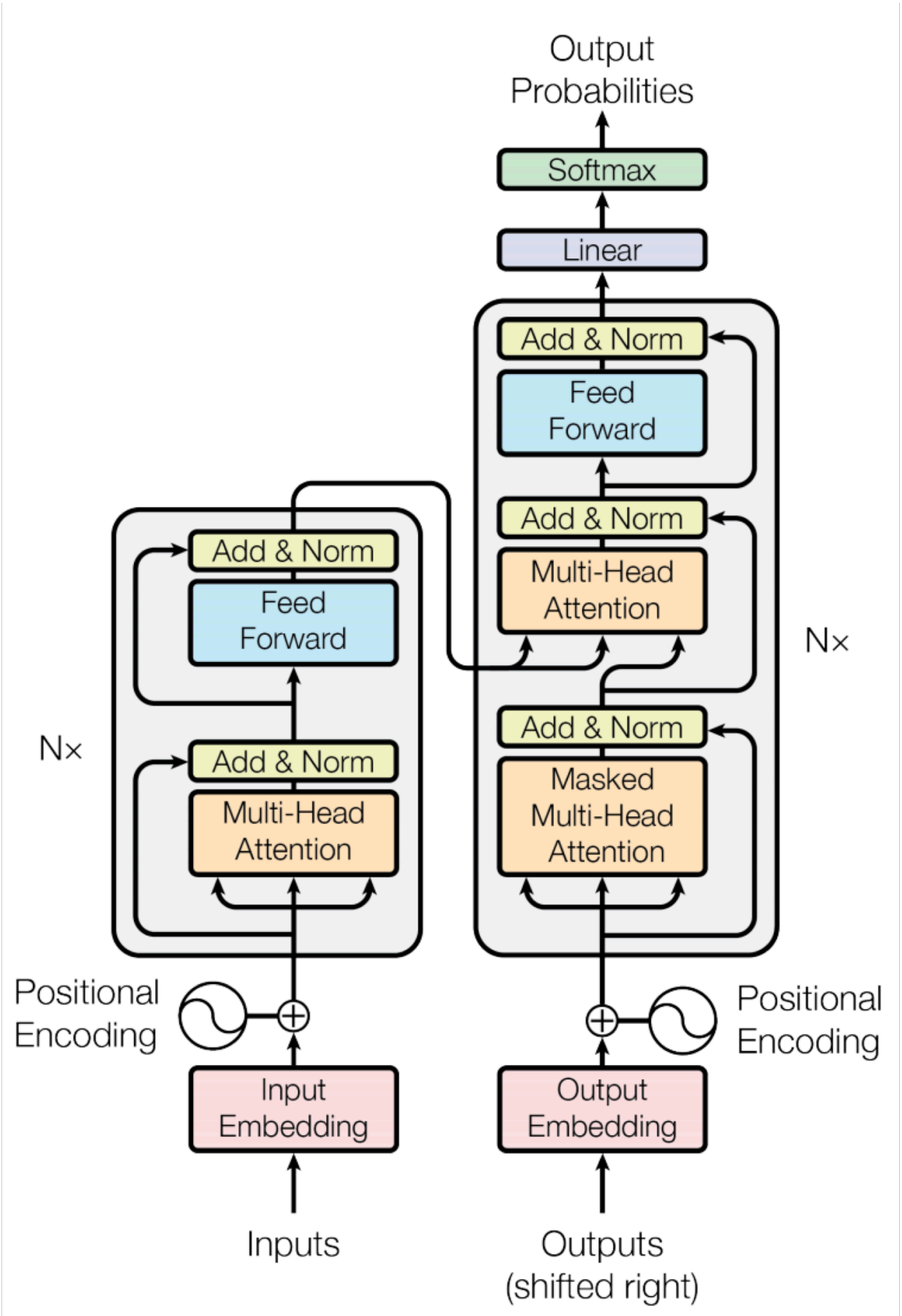
- 不同的颜色对应不同的注意力头

Attention visualization: Implicit anaphora resolution

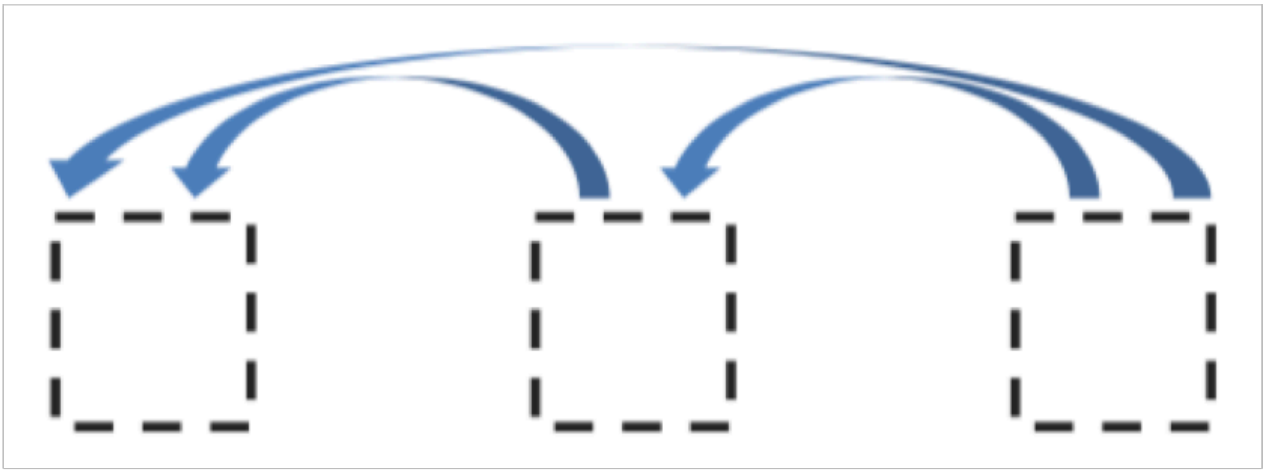


- 对于代词，注意力头学会了如何找到其指代物
- 在第五层中，从 head 5 和 6 的单词“its”中分离出来的注意力。请注意，这个词的注意力是非常鲜明的。

Transformer Decoder



- decoder 中有两个稍加改变的子层
- 对之前生成的输出进行 Masked decoder self-attention



- Encoder-Decoder Attention，queries 来自于前一个 decoder 层，keys 和 values 来自于 encoder 的输出
- Blocks 同样重复 6 次

Tips and tricks of the Transformer

细节(论文/讲座)

- Byte-pair encodings
- Checkpoint averaging
- Adam 优化器控制学习速率变化
- 训练时，在每一层添加残差之前进行 Dropout
- 标签平滑
- 带有束搜索和长度惩罚的 Auto-regressive decoding
- 因为 transformer 正在蔓延，但他们很难优化并且不像LSTMs那样开箱即用，他们还不能很好与其他任务的构件共同工作

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	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Experimental Results for Parsing

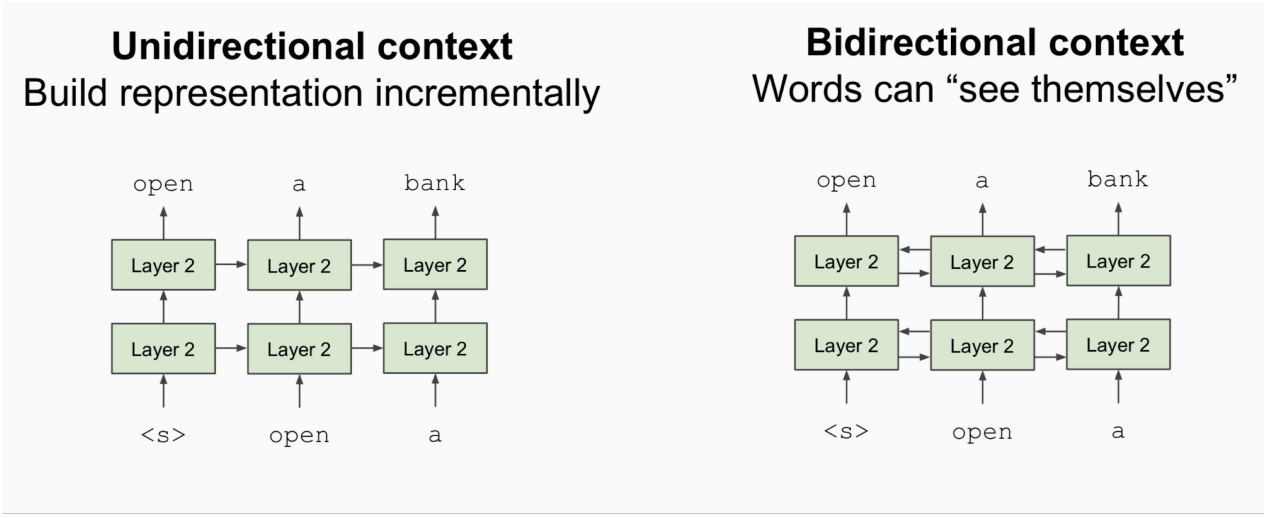
Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

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

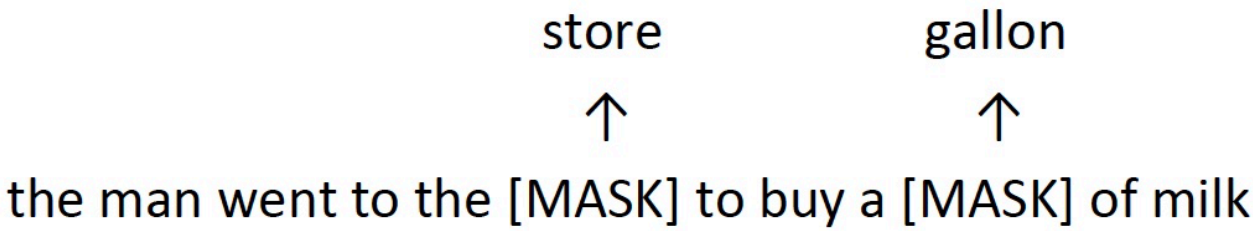
BERT (Bidirectional Encoder Representations from Transformers):

Pre-training of Deep Bidirectional Transformers for Language Understanding

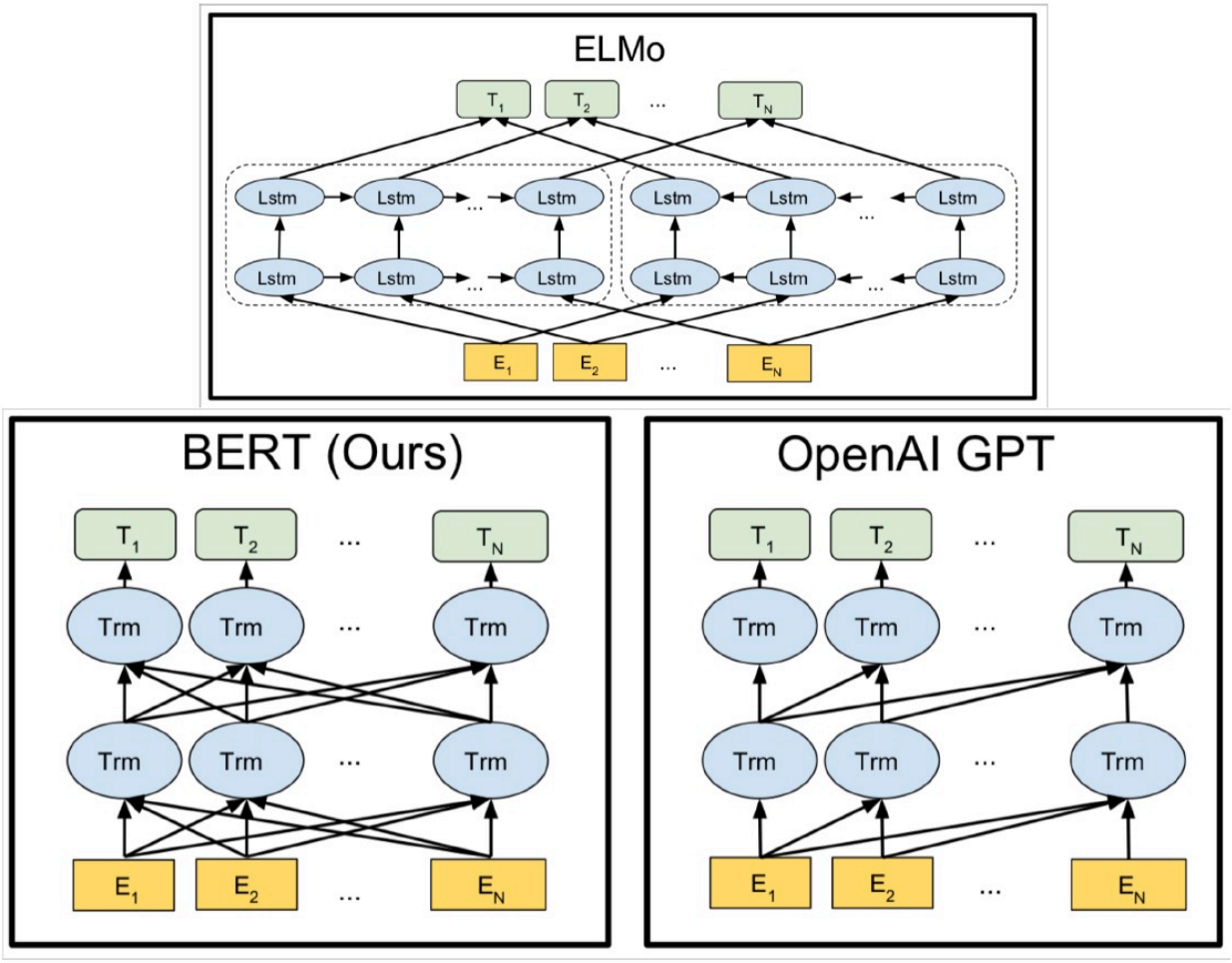
- **问题**：语言模型只使用左上下文或右上下文，但语言理解是双向的
- 为什么LMs是单向的?
- 原因1：方向性对于生成格式良好的概率分布是有必要的
  - 我们不在乎这个
- 原因2：双向编码器中单词可以“看到自己”



- **解决方案**：mask out k % 的输入单词，然后预测 masked words
- 不再是传统的计算生成句子的概率的语言模型，目标是填空
  - 总是使用k = 15%



- Masking 太少：训练太昂贵
- Masking 太多：没有足够的上下文



- GPT 是经典的单项的语言模型
- ELMo 是双向的，但是两个模型是完全独立训练的，只是将输出连接在一起，并没有使用双向的 context
- BERT 使用 mask 的方式进行整个上下文的预测，使用了双向的上下文信息

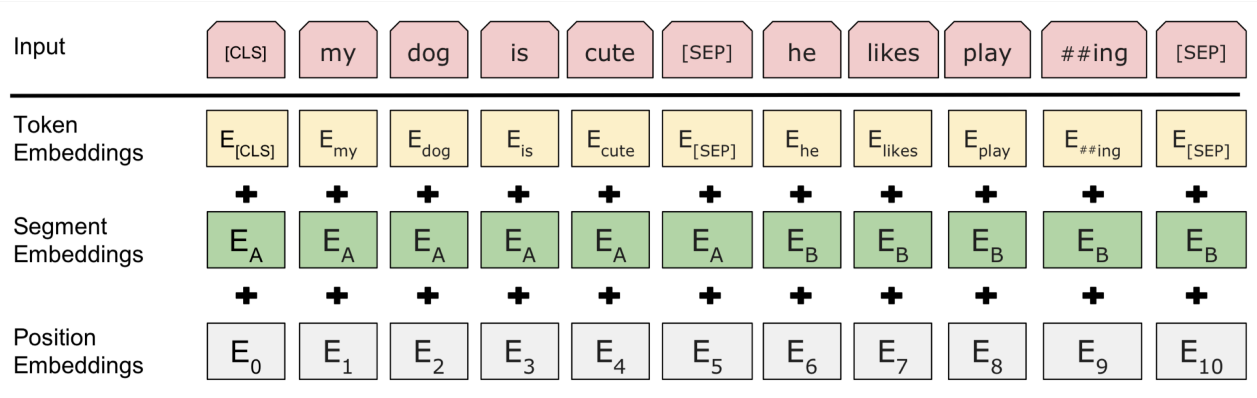
BERT complication: Next sentence prediction

- 学习句子之间的关系，判断句子 B 是句子 A 的后一个句子还是一个随机的句子。

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

BERT sentence pair encoding



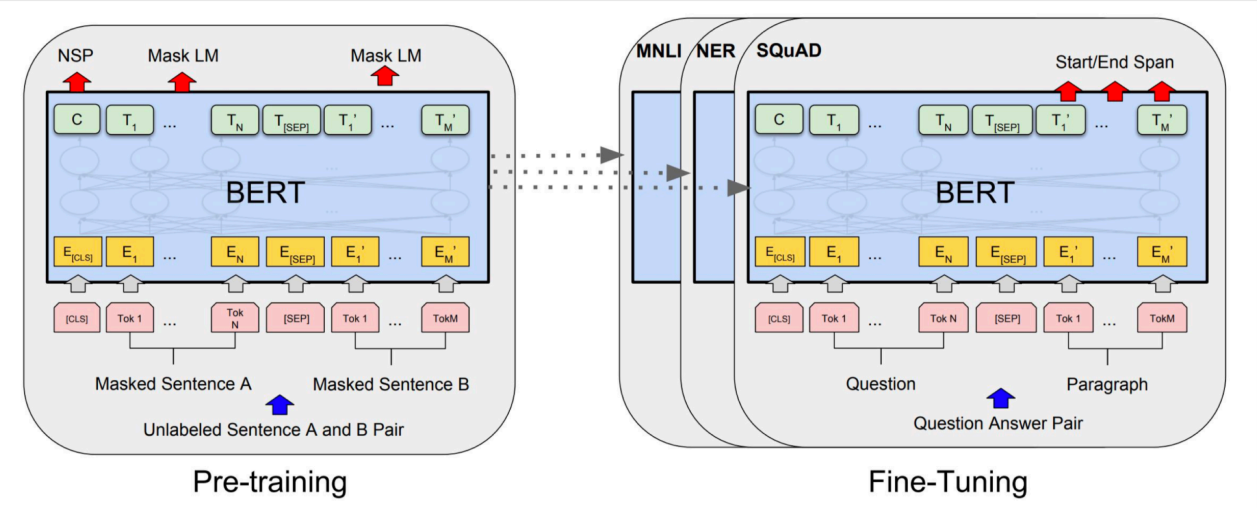
- token embeddings 是 word pieces （paly, ##ing）
- 使用学习好的分段嵌入表示每个句子
- 位置嵌入与其他 Transformer 体系结构类似
- 将以上三种 embedding 相加，作为最终输入的表示

BERT model architecture and training

- Transformer encoder（和之前的一样）
- 自注意力 → 没有位置偏差
  - 长距离上下文“机会均等”
- 每层乘法 ⇒ GPU / TPU上高效
- 在 Wikipedia + BookCorpus 上训练
- 训练两种模型尺寸
  - 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

- 只学习一个建立在顶层的分类器，微调的每个任务



BERT results on GLUE tasks

- GLUE benchmark 是由自然语言推理任务,还有句子相似度和情感
- **MultiNLI**

- Premise: Hills and mountains are especially sanctified in Jainism.

  Hypothesis: Jainism hates nature.

  Label: Contradiction
- **CoLa**
- Sentence: The wagon rumbled down the road. Label: Acceptable
- Sentence: The car honked down the road. Label: Unacceptable

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

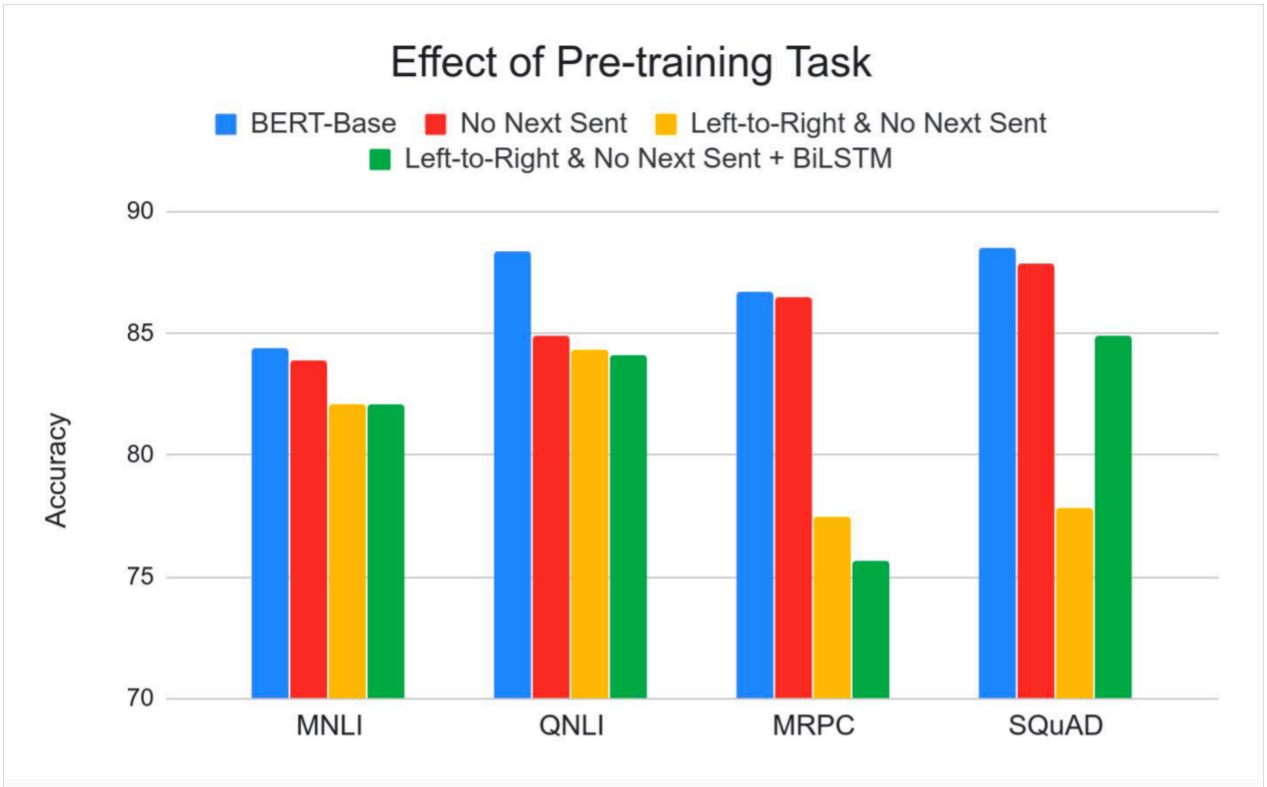
BERT results on SQuAD 1.1

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Oct 05, 2018	BERT (single model) *Google AI Language* https://arxiv.org/abs/1810.04805	85.083	91.835
2 Sep 26, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.954	91.677
5 Sep 09, 2018	nlnet (single model) *Microsoft Research Asia*	83.468	90.133
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

SQuAD 2.0 leaderboard, 2019-02-07

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Jan 15, 2019	BERT + MMFT + ADA (ensemble) *Microsoft Research Asia*	**85.082**	**87.615**
2 Jan 10, 2019	BERT + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	84.292	86.967
3 Dec 13, 2018	BERT finetune baseline (ensemble) *Anonymous*	83.536	86.096
4 Dec 16, 2018	Lunet + Verifier + BERT (ensemble) *Layer 6 AI NLP Team*	83.469	86.043
4 Dec 21, 2018	PAML+BERT (ensemble model) *PINGAN GammaLab*	83.457	86.122
5 Dec 15, 2018	Lunet + Verifier + BERT (single model) *Layer 6 AI NLP Team*	82.995	86.035

Effect of pre-training task



Size matters

- 从 119M 到 340M 的参数量改善了很多
- 改进尚未渐进

Suggested Readings

The Annotated Transformer 代码解析

- https://github.com/rsennrich/subword-nmt
- https://github.com/opennmt/opennmt-py

jalammar 的一系列可视化简单教程

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

The Illustrated Transformer

> **Go Forth And Transform**
>
> I hope you've found this a useful place to start to break the ice with the major concepts of the Transformer. If you want to go deeper, I'd suggest these next steps:
>
> - Read the Attention Is All You Need paper, the Transformer blog post (Transformer: A Novel Neural Network Architecture for Language Understanding), and the Tensor2Tensor announcement.
> - Watch Łukasz Kaiser's talk walking through the model and its details
> - Play with the Jupyter Notebook provided as part of the Tensor2Tensor repo
> - Explore the Tensor2Tensor repo.
>
> **Follow-up works**
>
> - Depthwise Separable Convolutions for Neural Machine Translation

> - One Model To Learn Them All
> - Discrete Autoencoders for Sequence Models
> - Generating Wikipedia by Summarizing Long Sequences
> - Image Transformer
> - Training Tips for the Transformer Model
> - Self-Attention with Relative Position Representations
> - Fast Decoding in Sequence Models using Discrete Latent Variables
> - Adafactor: Adaptive Learning Rates with Sublinear Memory Cost

The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)

Smith, Noah A. Contextual Word Representations: A Contextual Introduction.

Reference

以下是学习本课程时的可用参考书籍：

《基于深度学习的自然语言处理》（车万翔老师等翻译）

《神经网络与深度学习》

以下是整理笔记的过程中参考的博客：

斯坦福CS224N深度学习自然语言处理2019冬学习笔记目录 (课件核心内容的提炼，并包含作者的见解与建议)

斯坦福大学 CS224n自然语言处理与深度学习笔记汇总 {>>这是针对note部分的翻译<<}