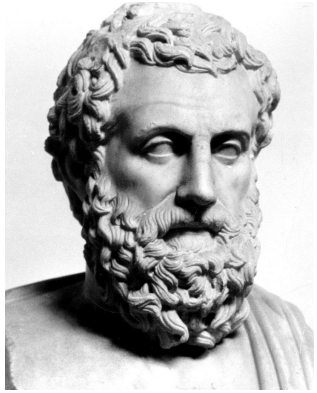
Artificial Intelligence For NLP Lesson-14



人工智能与自然语言处理课程组

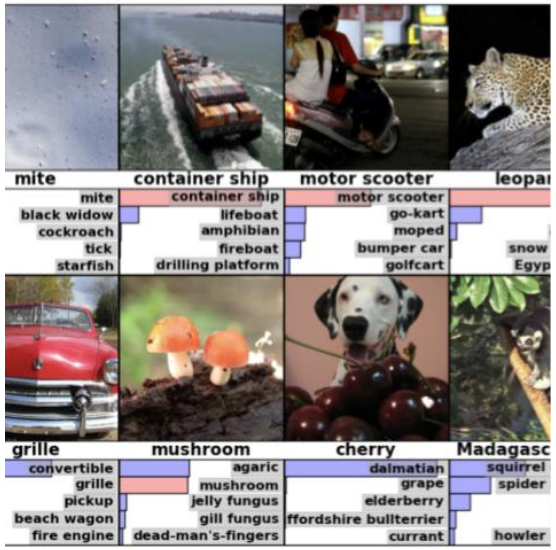
2019.Dec.13

Outline

1. 复习 Word2Vec
2. Sequence to Sequence (Seq2Seq)
3. CoVe, EMLo, FastText, SQUAD
4. 参考文献
5. 关于项目3

Why do we need seq2seq

1. Classification
    - f(vec) -> tag, categorical
2. Regression
    - f(vec) -> numerical value



An example

Given a name, classify the right nationality

> *Dovesky*

(-0.87) Czech

(-0.88) Russian

(-2.44) Polish

> *Jackson*

(-0.74) Scottish

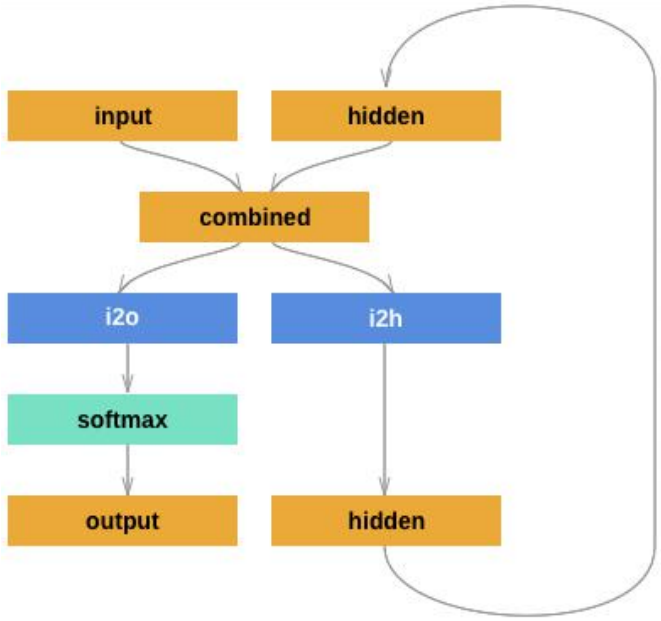
(-2.03) English

(-2.21) Polish

> *Satoshi*

(-0.77) Arabic

(-1.35) Japanese




```
def train(category_tensor, line_tensor):
    rnn.zero_grad()
    hidden = rnn.init_hidden()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()

    optimizer.step()

    return output, loss.data[0]
```

Why not generate a new name based on nation?

> python generate.py Russian

Rovakov

Uantov

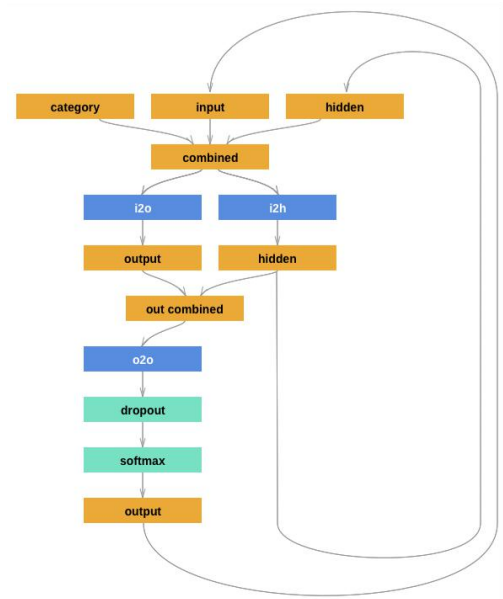
Shavakov

> python generate.py German

Gerren

Ereng

Rosher



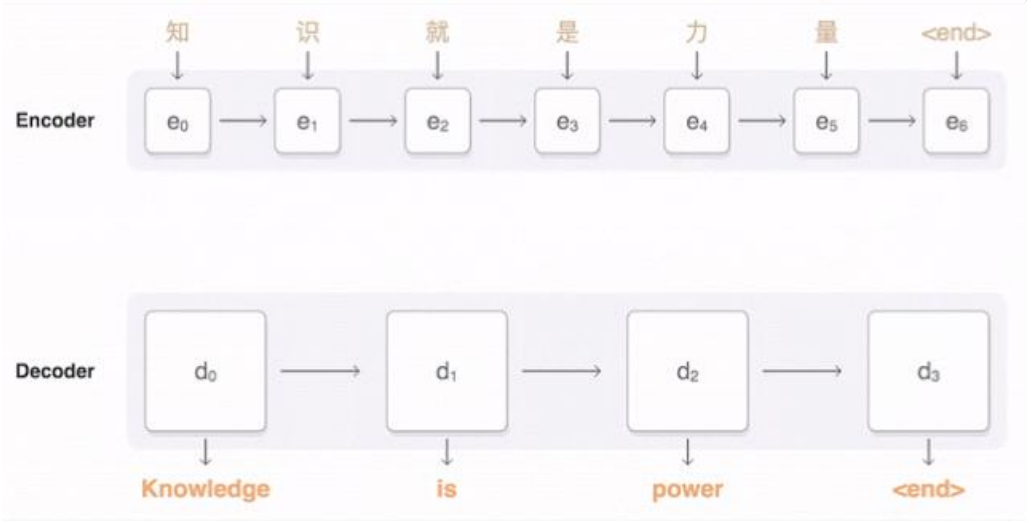

```
def train(category_tensor, input_line_tensor, target_line_tensor):
    hidden = rnn.init_hidden()
    optimizer.zero_grad()
    loss = 0

    for i in range(input_line_tensor.size()[0]):
        output, hidden = rnn(category_tensor, input_line_tensor[i], hidden)
        loss += criterion(output, target_line_tensor[i])

    loss.backward()
    optimizer.step()

    return output, loss.data[0] / input_line_tensor.size()[0]
```

variable length

-> nation

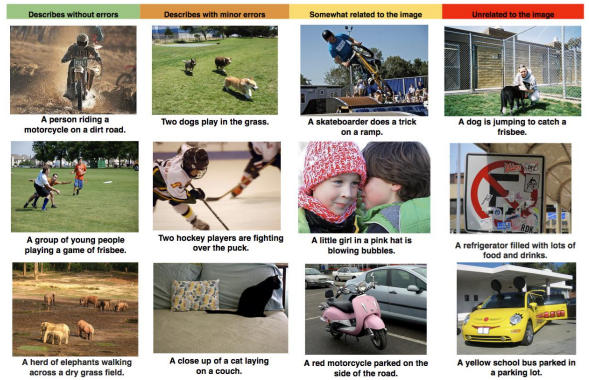
nation

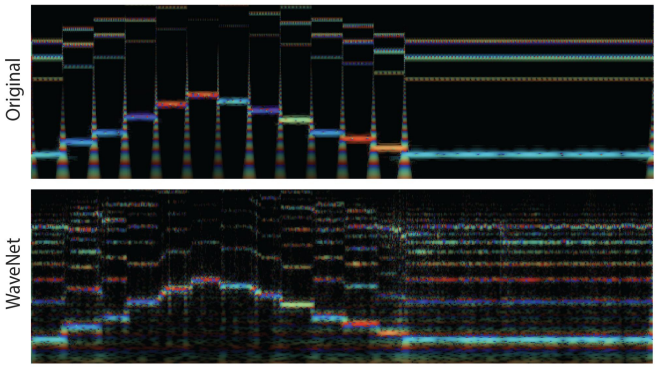
-> variable length



Input sentence:	*Translation (PBMT):*	*Translation (GNMT):*	*Translation (human):*
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

1. Image Caption.
2. Music Generation.
3. Text Genearation.
4. Text Summarisation.
5. Machine Translation.
6. Dialogue System.





Input: <vec1, vec2, .. vec_n> Output: <vecN1, vecN2.. vecNi>

标题：（输入标题可以获得更好的摘要）

突发！林肯公园主唱Chester自缢身亡 年仅41岁

正文

网易娱乐7月21日报道 林肯公园主唱查斯特·贝宁顿Chester Bennington于今天早上在洛杉矶帕洛斯弗迪斯的一个私人庄园自缢身亡，年仅41岁。此消息已得到洛杉矶警方证实。
洛杉矶警方透露，Chester的家人正在外地度假，Chester独自在家，上吊地点是家里的二楼。一说是一名音乐公司工作人员来家里找他时发现了尸体，也有人称是佣人最早发现其死亡。
林肯公园另一位主唱麦克·信田确认了Chester Bennington自杀属实，并对此感到震惊和心痛，称稍后官方会发布声明。Chester昨天还在推特上转发了一条关于曼哈顿垃圾山的新闻。粉丝们纷纷在该推文下留言，不相信Chester已经走了。
外媒猜测，Chester选择在7月20日自杀的原因跟他极其要好的朋友、Soundgarden（声音花园）乐队以及Audioslave乐队主唱Chris Cornell有关，因为7月20日是Chris Cornell的诞辰。而Chris Cornell 于今年5月17日上吊自杀，享年52岁。Chris去世后，Chester还为他写下悼文。
对于Chester的自杀，亲友表示震惊但不意外，因为Chester曾经透露过想自杀的念头，他曾表示自己童年时被虐待，导致他医生无法走出阴影，也导致他长期酗酒和嗑药来疗伤。目前，洛杉矶警方仍在调查Chester的死因。

自动摘要

7月21日报道 林肯公园主唱查斯特·贝宁顿Chester Bennington于今天早上在洛杉矶帕洛斯弗迪斯的一个私人庄园自缢身亡，年仅41岁。外媒猜测，Chester选择在7月20日自杀的原因跟他极其要好的朋友、Soundgarden乐队以及Audioslave乐队主唱Chris Cornell有关，因为7月20日是Chris Cornell的诞辰。

Encoder-decoder model

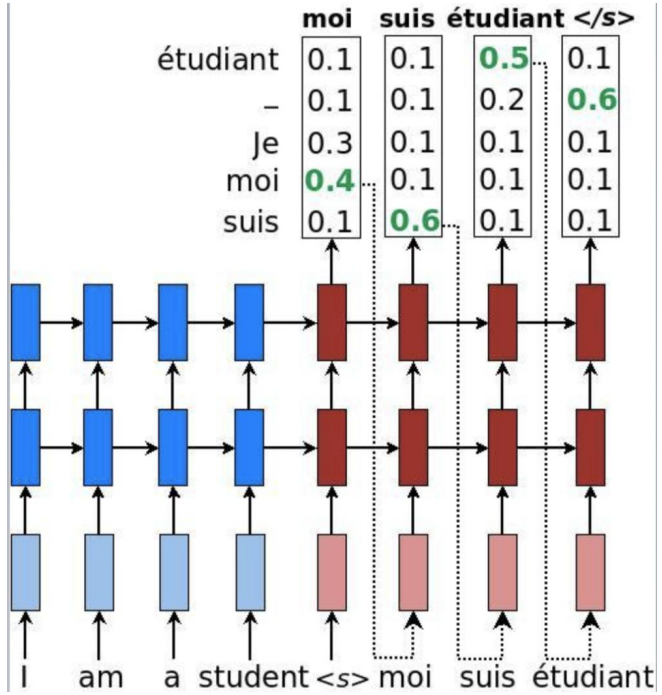
$$\boldsymbol{m}_t^{(f)} = M_{\cdot,f_t}^{(f)}$$

$$\boldsymbol{h}_t^{(f)} = \begin{cases} \mathrm{RNN}^{(f)}(\boldsymbol{m}_t^{(f)}, \boldsymbol{h}_{t-1}^{(f)}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

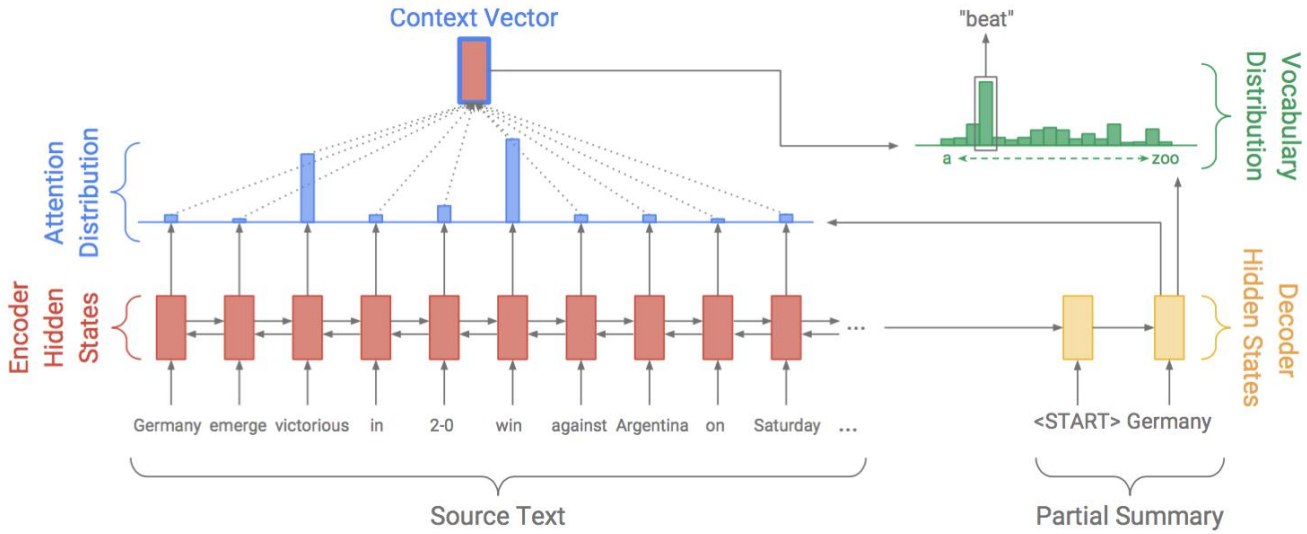
$$\boldsymbol{m}_t^{(e)} = M_{\cdot,e_{t-1}}^{(e)}$$

$$\boldsymbol{h}_t^{(e)} = \begin{cases} \mathrm{RNN}^{(e)}(\boldsymbol{m}_t^{(e)}, \boldsymbol{h}_{t-1}^{(e)}) & t \geq 1, \\ \boldsymbol{h}_{|F|}^{(f)} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t^{(e)} = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t^{(e)} + b_s)$$



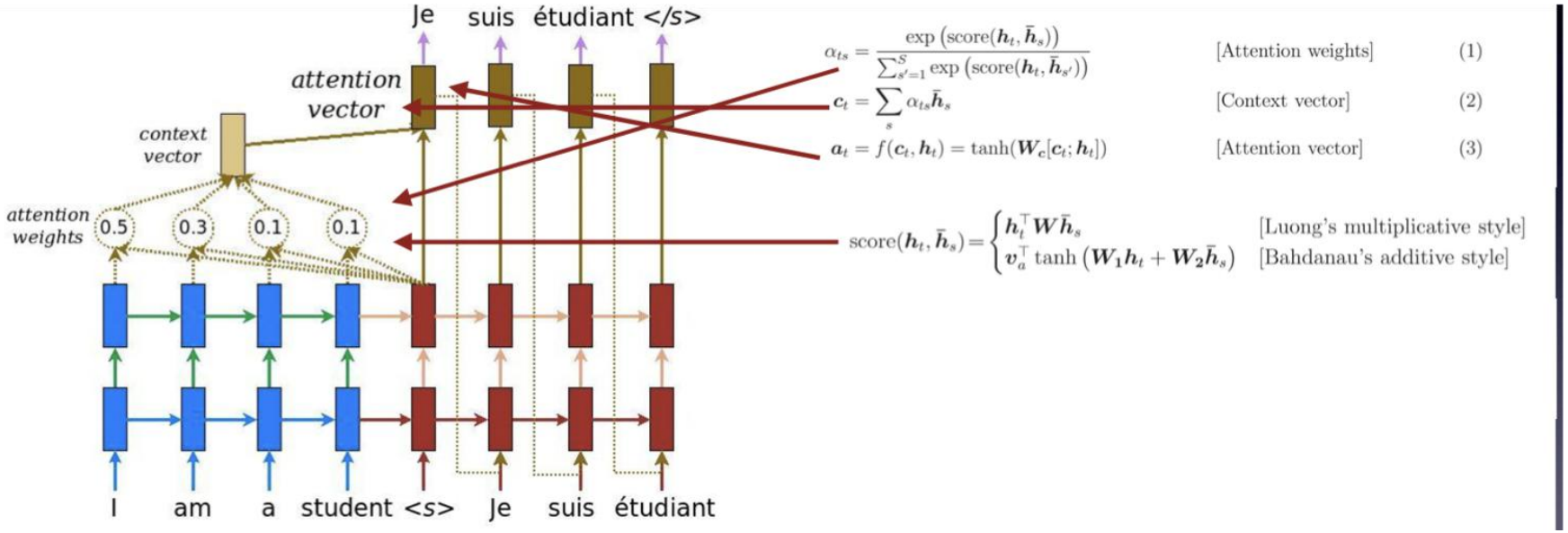
Seq2seq with attention



Look more information from input source, not only based on encoder result.

Dzmitry Bahdanau， et al, Neural Machine Translation by Jointly Learning to Align and Translate，

Attention vector



Attention is our firend

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

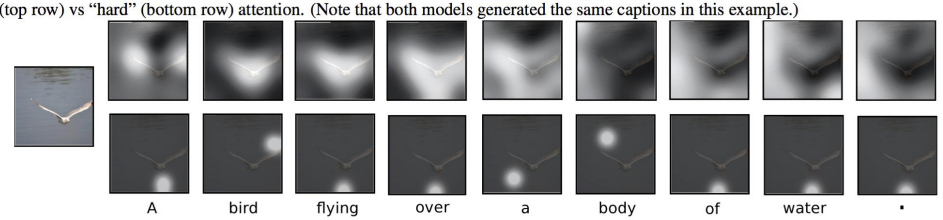
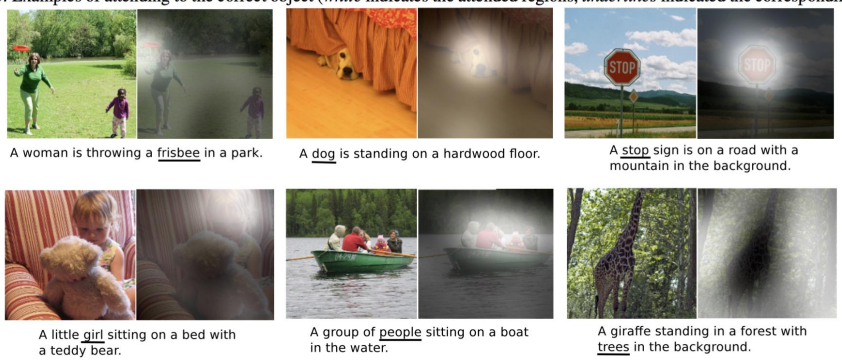
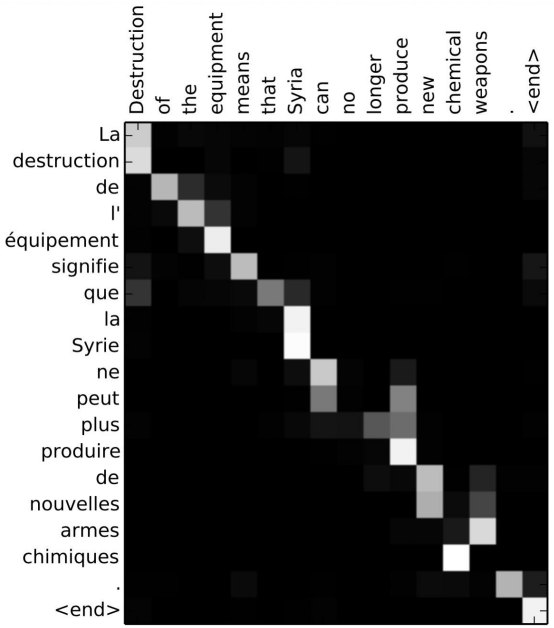


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)





Generate output

1. Random Sampling: Randomly select an output E from the probability distribution P (E | F). This is usually denoted E^ <-- P(E | F).

while $\hat{e}_t \neq$ "$\langle$/s$\rangle$" **do**
  $t \leftarrow t + 1$
  Calculate $\boldsymbol{m}_t^{(e)}$, $\boldsymbol{h}_t^{(e)}$, and $\boldsymbol{p}_t^{(e)}$ from $\hat{e}_{t-1}$
  Sample $\hat{e}_t$ according to $\boldsymbol{p}_t^{(e)}$

Get output_t, based on the probability distribution from context calculated by the time step output_{t-1}.

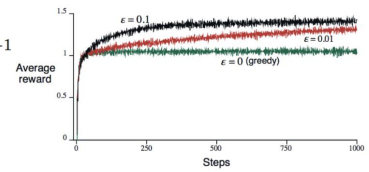
Random Output: Dialogue System Fast Computing.

2. *Greedy 1-best Search*: model output the best result, in which we simply calculate p_t, at every time step, select the word that gives us the highest probability, and use it as the next word in our sequence.

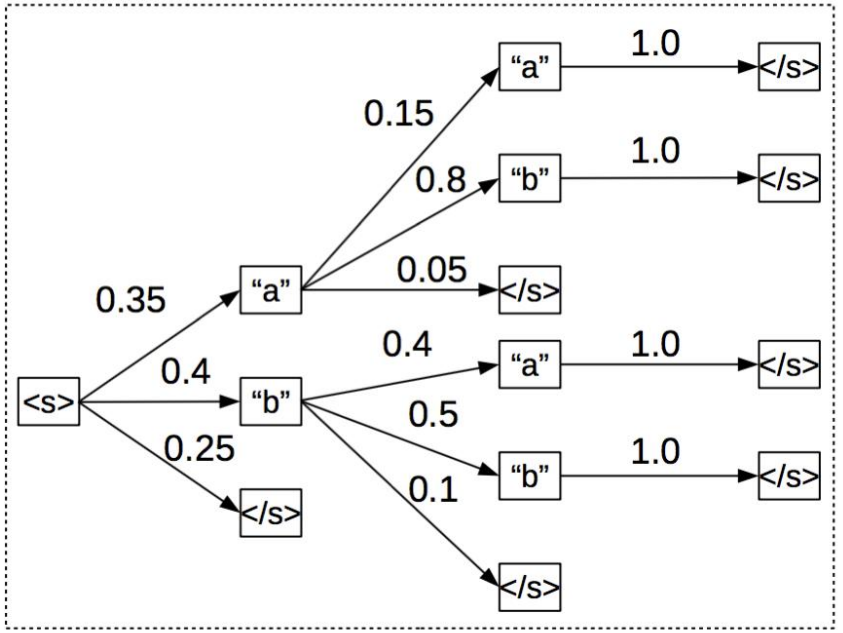
! greedy search is not guaranteed to find the translation with the highest probability

while $\hat{e}_t \neq$ "⟨/s⟩" **do**
$t \leftarrow t + 1$
Calculate $\boldsymbol{m}_t^{(e)}$, $\boldsymbol{h}_t^{(e)}$, and $\boldsymbol{p}_t^{(e)}$ from $\hat{e}_{t-1}$
Sample $\hat{e}_t$ according to $\boldsymbol{p}_t^{(e)}$

$\hat{e}_t = \text{argmax } p^{(e)}$.

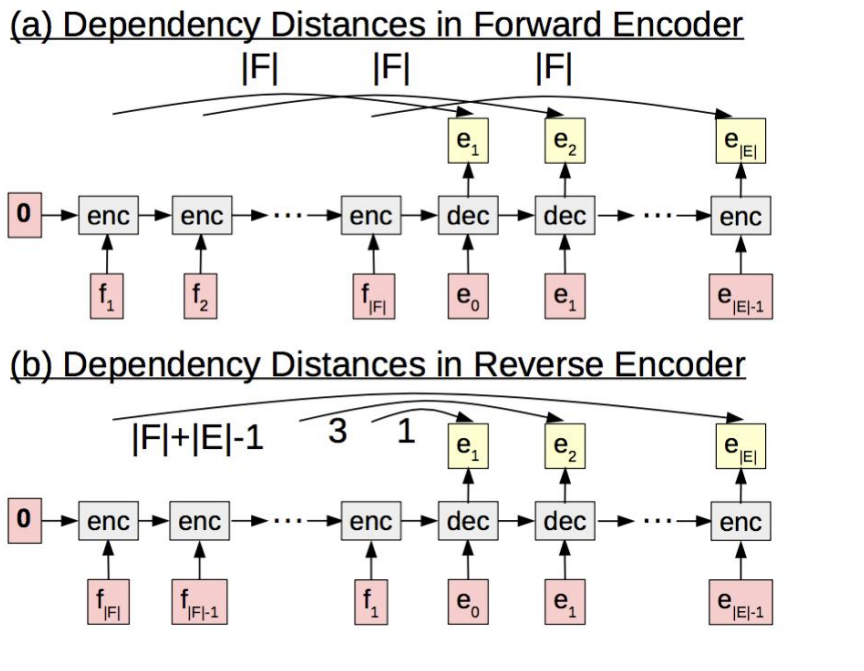


3. *Beam-Search*: we consider b best hypotheses at each time step, where b is the "width" of the beam.



Other ways of encoding sequence

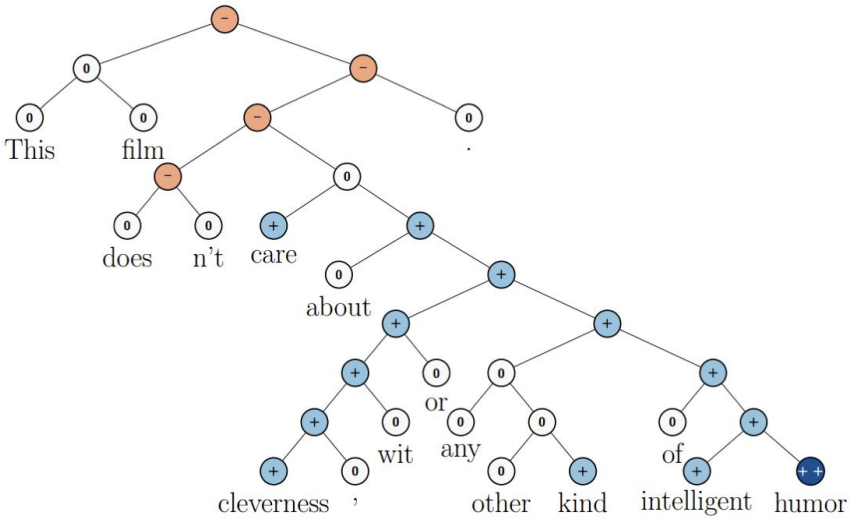
Reverse and Bidirectional



2. Tree-RNN

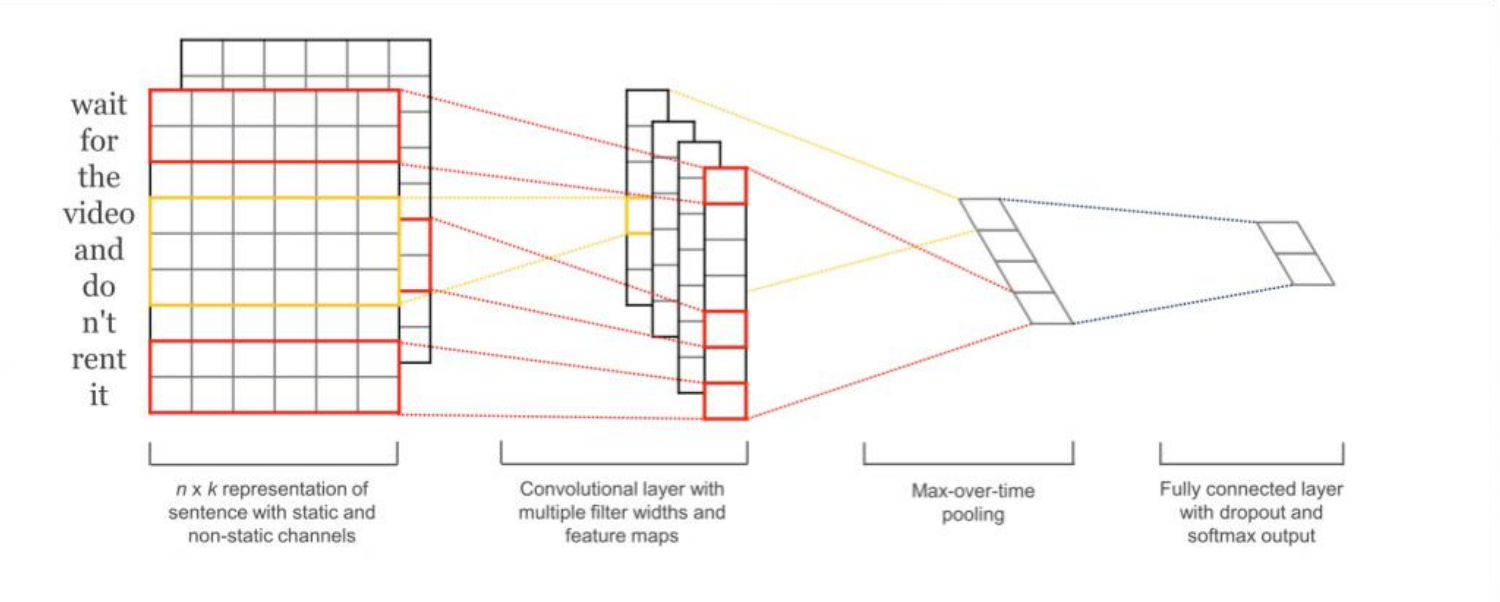
LSTMs or other RNNs are simpler format of Tree-RNN.

Need dynamic graph support.



3. Convolutional Neural Networks.

It’s much faster than using RNN to model Text.



TensorFlow.Seq2Seq

TensorlFlow >= 1.2v
If version is older than 1.2, could look more

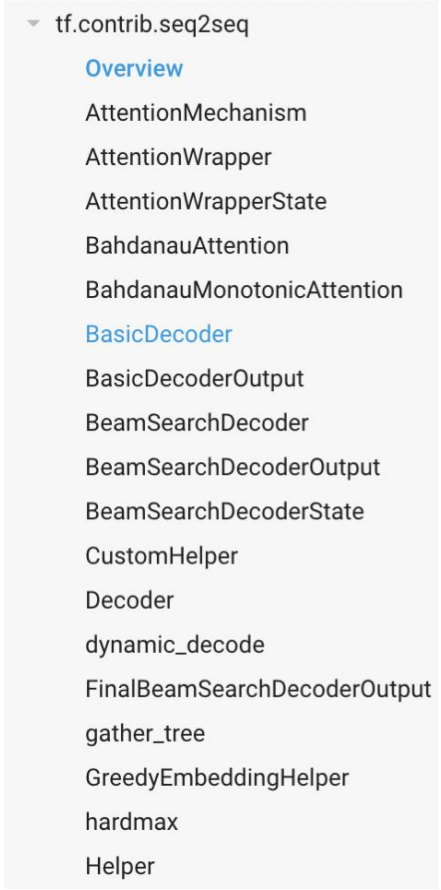
information in https://github.com/google/seq2seq.

- 1. Build Training Batch.

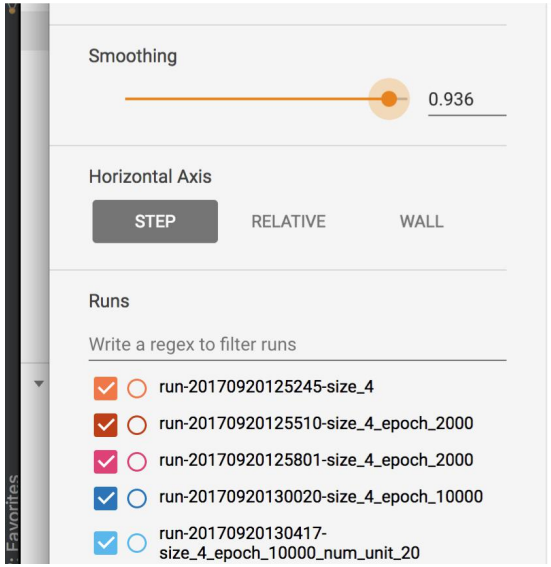
- 2. Embedding.
- 3. Get Encoding.
- 4.AddAttention.

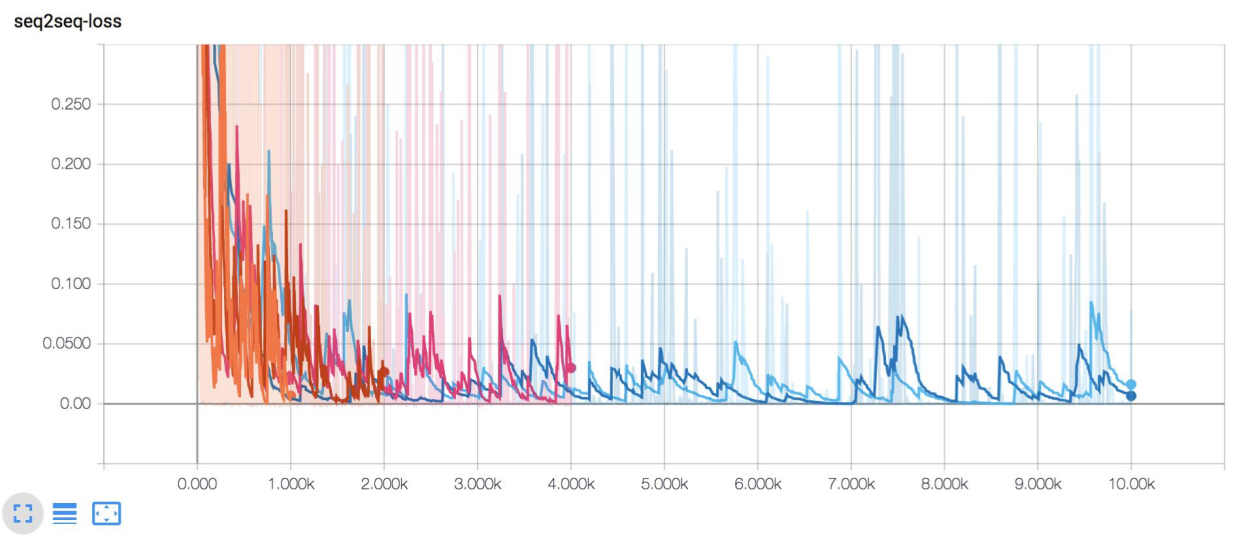
- 5. Decode.
- 6. Train.
- 7. Genrate Outputs.

- 8. Result Analysis.



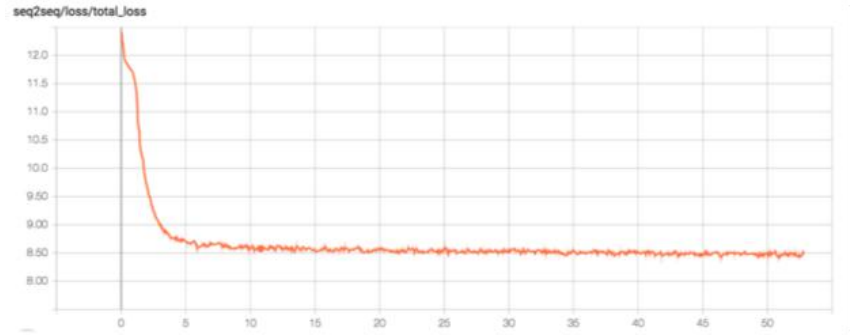
Resulting analysis





2. Training Need Very long time. Aili Mi. Need 30 days for a training.

Training on Nvidia GTX 8 The loss after 24 hours.



Cove

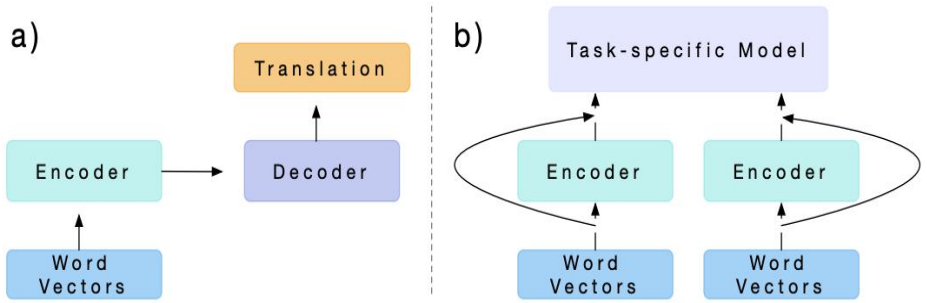


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph. How will your system compare to humans on this task?

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	PAML+BERT (ensemble model)	83.457	86.122

ELMO

一种新型深度语境化词表征

复习

- one-hot编码

卡车 = [1 0 0 0 0]

轮船 = [0 1 0 0 0]

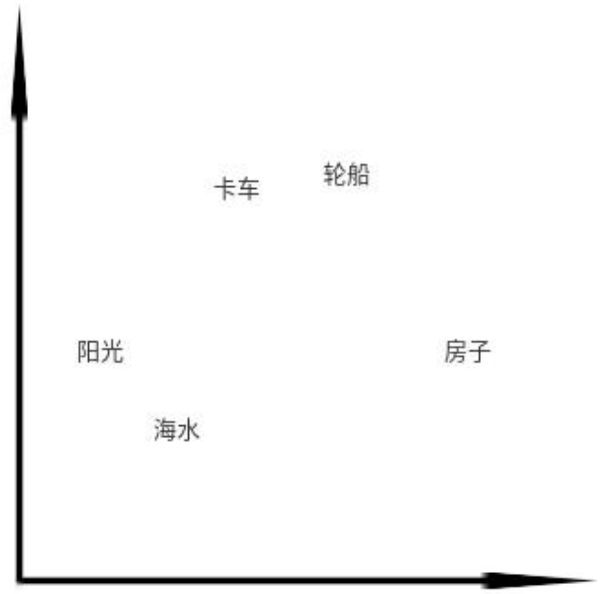
房子 = [0 0 1 0 0]

海水 = [0 0 0 1 0]

阳光 = [0 0 0 0 1]

每个词都是独立的，词与词之间没有关联

- word embedding



根据上下文训练出每个词的向量

相近的词会靠的比较近

问题：同一个词可能有多个意思

【某中文八级题目】写出下面四个句子中的 “意思” ：

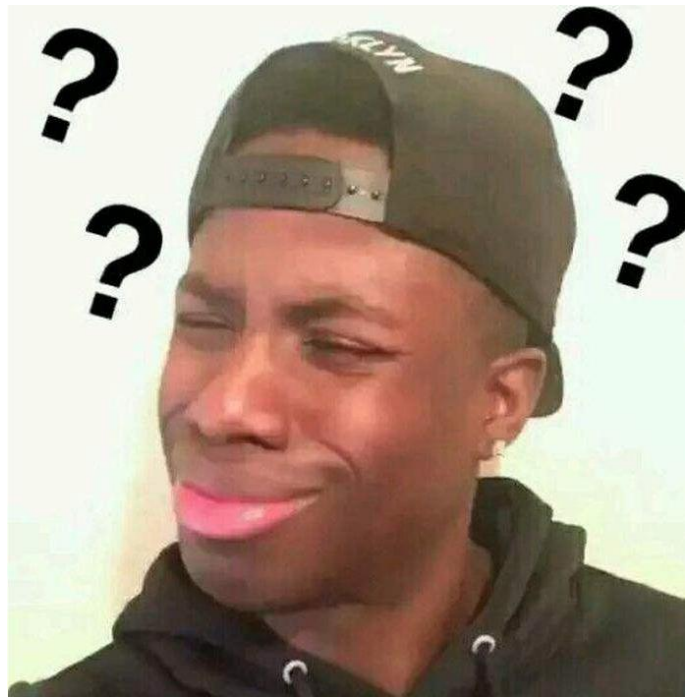
我明白这句话的意思了。

他好像对她有点意思。

这下就有意思了。

你就手下吧，意思意思下。

那么这么多个“意思”，都是什么意思？？？



问题：同一个词可能有多个意思

【某中文八级题目】写出下面四个句子中的 “意思” ：

我明白这句话的意思了。

他好像对她有点意思。

这下就有意思了。

你就手下吧，意思意思下。

之间讲的word embedding虽然解决了同类词的问题，但是对于同一个Token来说，它的embedding结果都是一样的。

所以左边四个句子的“意思”都是同一个向量，但是“中文十级”的我们肯定会发现这四个“意思”是不一样的意思。

那么要怎么解决 同一个词 在不同地方表达不同的意思？ELMO！

ELMO的做法

同一个Token在不同地方的embedding是不同的。

我明白这句话的意思了。 【token1】

他好像对她有点意思。 【token2】

这下就有意思了。 【token3】

你就手下吧，意思意思下。【token4】

ELMO的做法

同一个Token在不同地方的embedding是不同的。

我明白这句话的意思了。 【token1】

他好像对她有点意思。 【token2】

这下就有意思了。 【token3】

你就手下吧，意思意思下。【token4】

contextualized word embedding

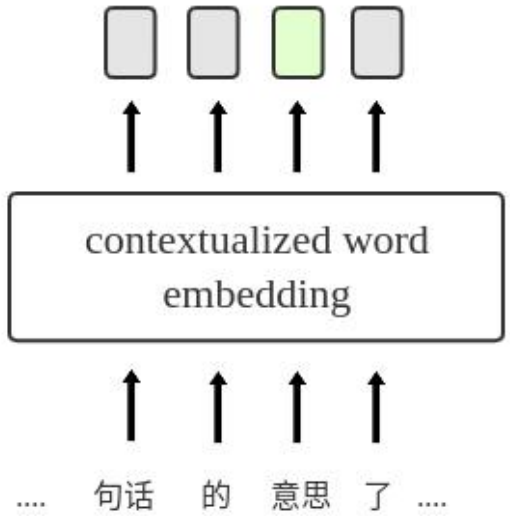
同一个Token在不同地方的embedding是不同的。

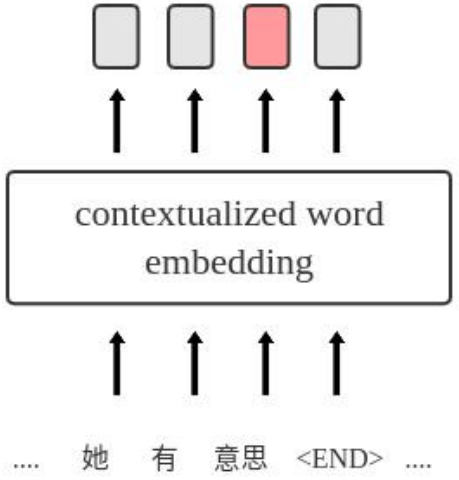
我明白这句话的意思了。 【token1】

他好像对她有点意思。 【token2】

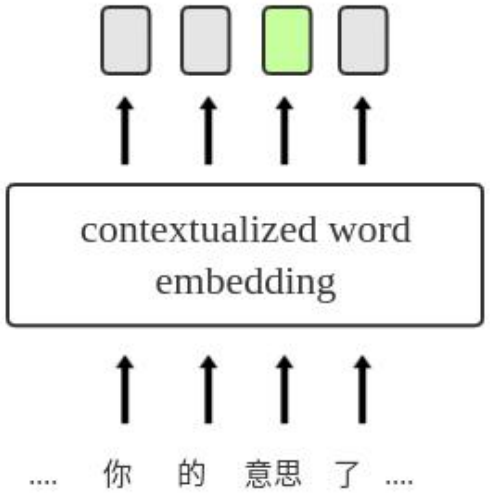
我明白你的意思了。 【token3】

你就手下 4】





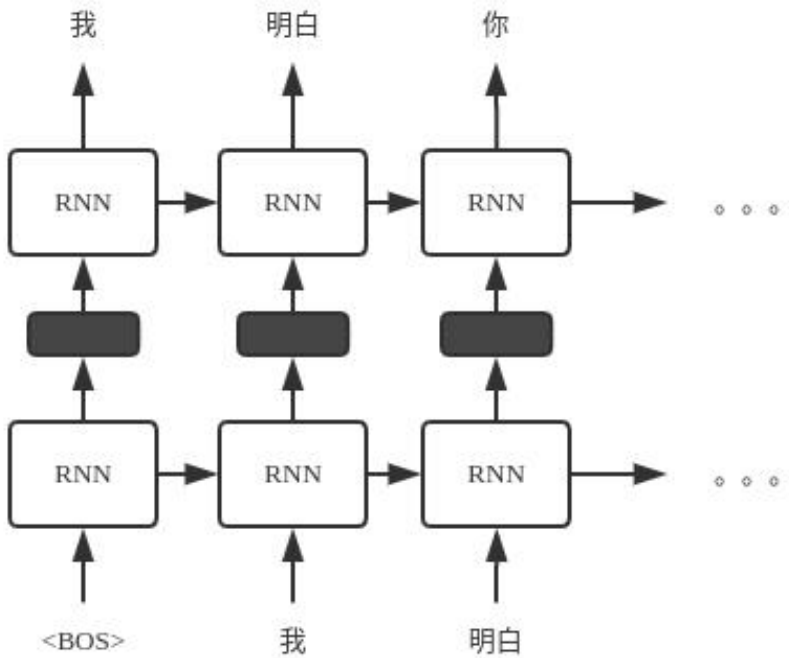
Token1 和Token3的“意思”表达的都是同一个东西，所以，output也会像相似



Embedding from Language Model

ELMO是一种 【RNN-based language models】

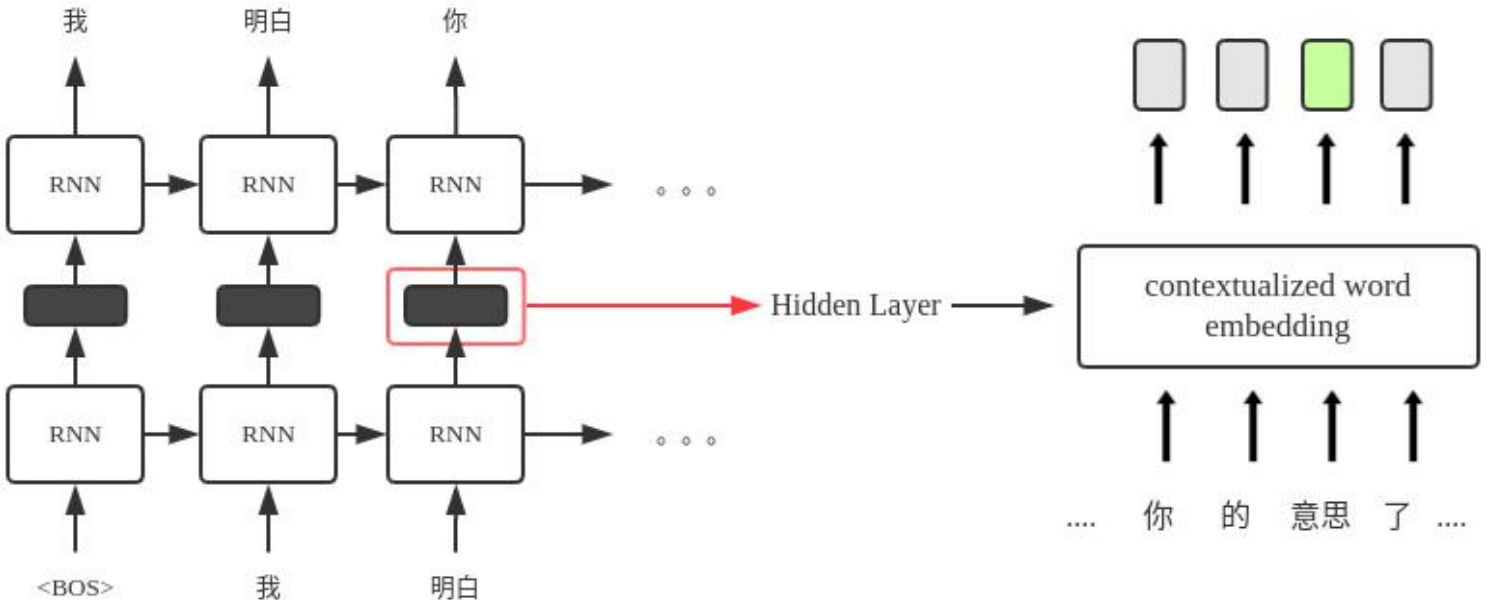
通过RNN的model去预测下一个token



Embedding from Language Model

等RNN学习好后，可以把里面的Hidden Layer

拿出来做 contextualized word embedding了

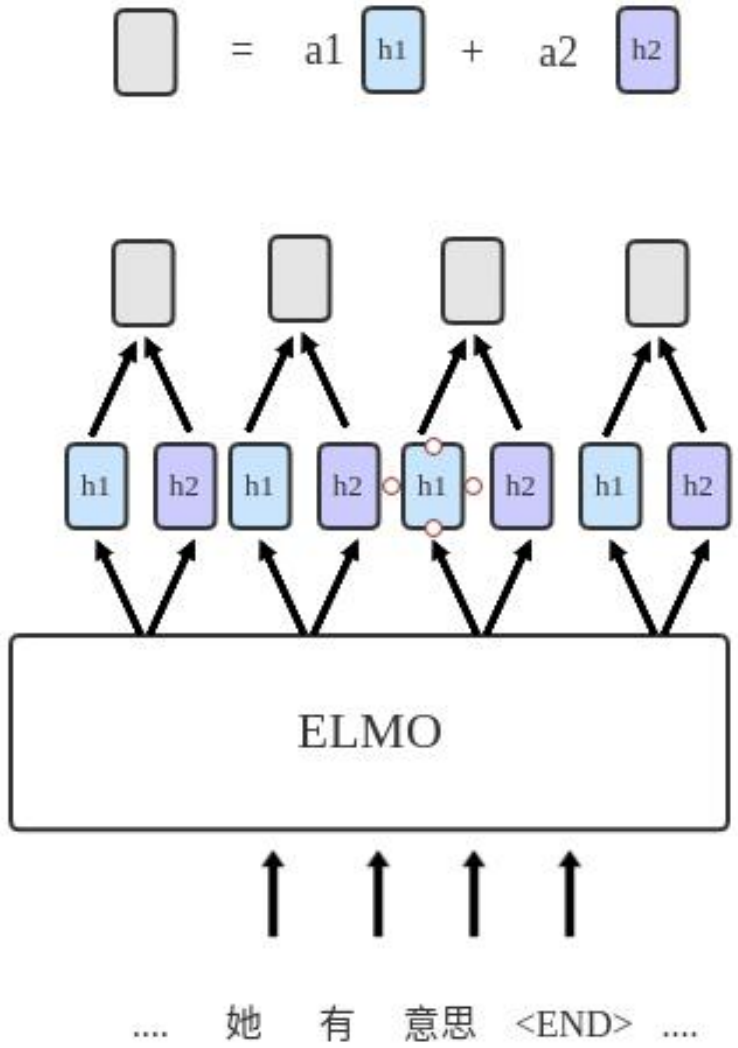


ELMO

一般训练的RNN model都会很deep，需要取出那一层Hidden Layer来呢？

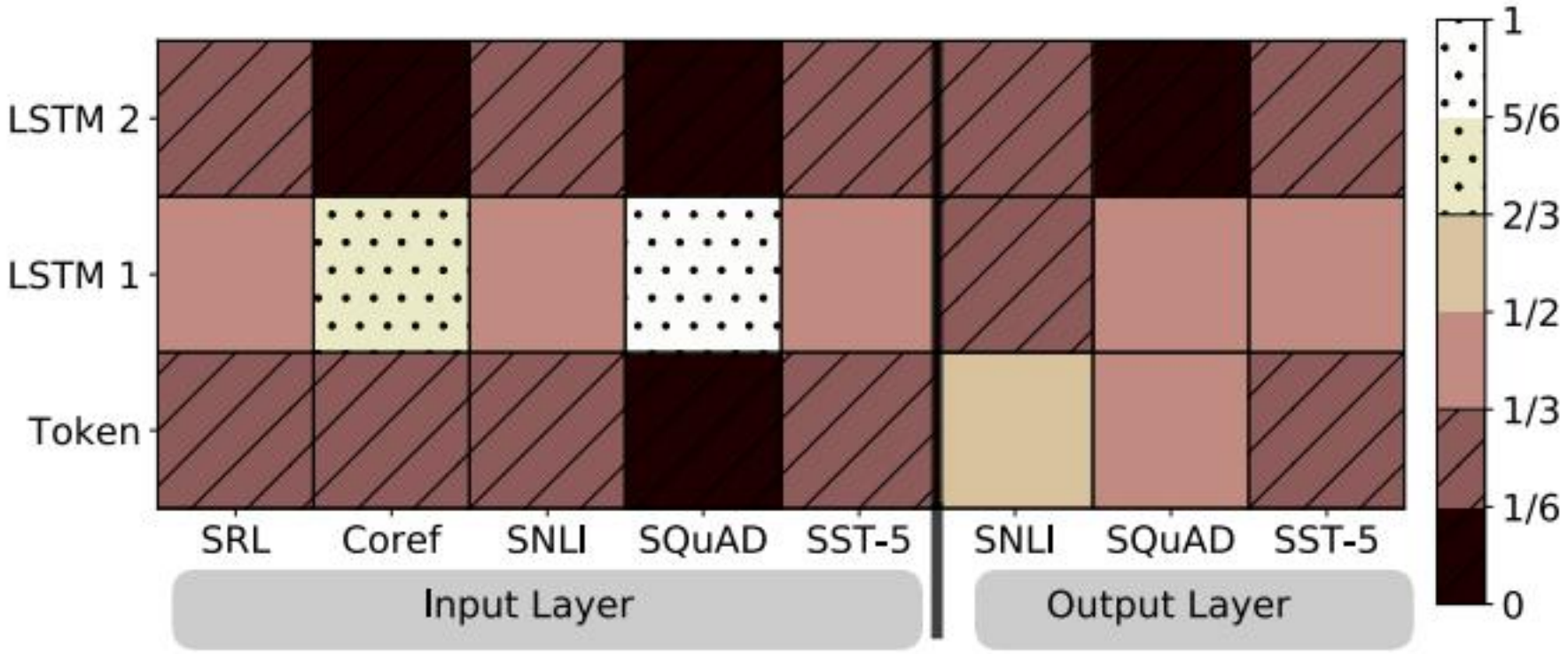
ELMO会把所有的Hidden Layer都取出来，通过加权加起来.

其中a1和a2这两个权重是通过不同的任务训练出来的



这就意味着，对于不同的任务，ELMO都要重新训练这个最后的权重。

在ELMO的Paper里面给出了几个实验的结果。结果里面发现Coref任务和SquaD任务对于Hidden Layer1都很需要，其他任务的权重都比较平均。

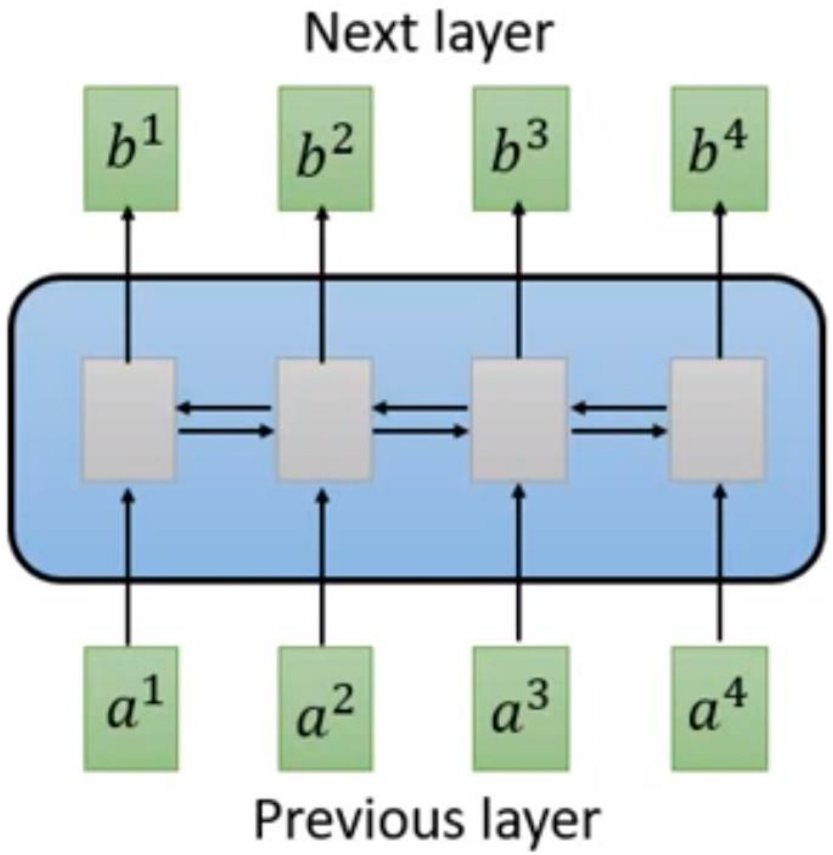


Deep contextualized word representations： https://arxiv.org/abs/1802.05365

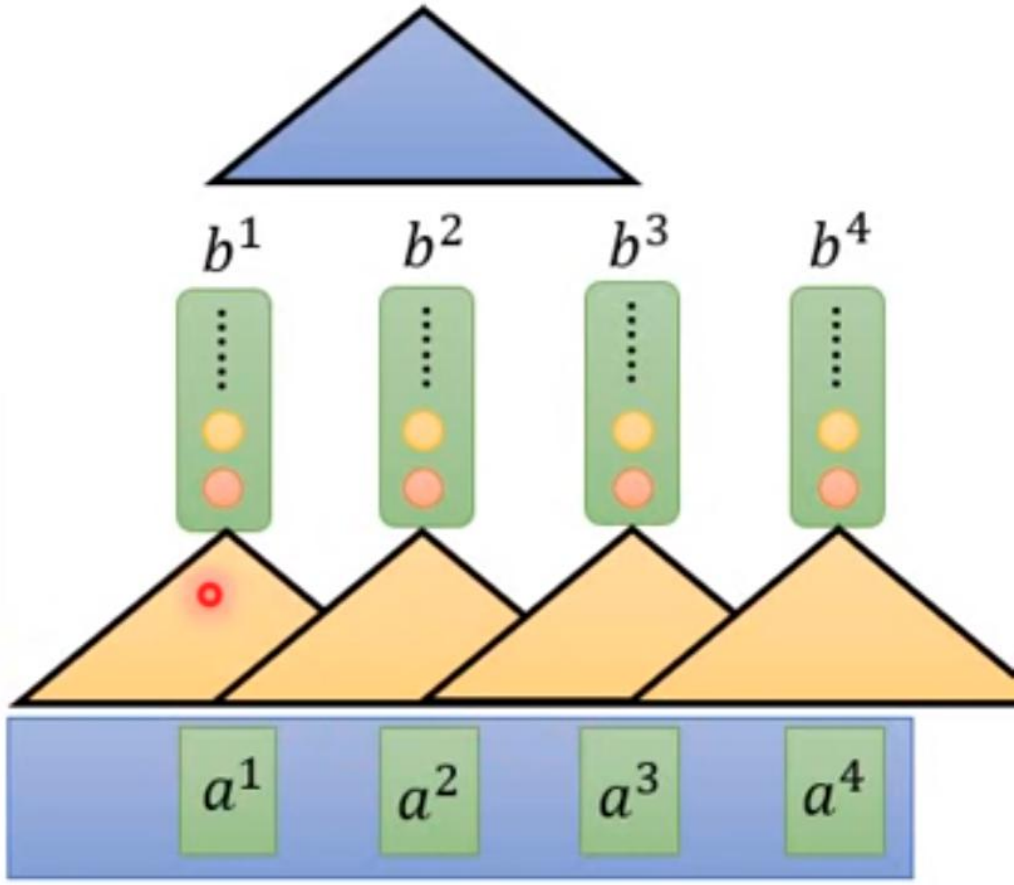
Transformer

Seq2seq model with "self-attention

之前讲的Seq2seq模型里面用的是RNN结构，在这个模型里面，如果要计算b4那么需要先计算b1,b2,b3，整个计算过程都是串行的，所以一般我们在训练RNN的时候都会比较久。



于是有人提出用CNN来替代RNN，因为CNN也可以通过多层卷积来联系句子前后不同的token，同样可以达到输入一个sequence，输出一个sequence的目的。



好处在于可以进行并行计算，在计算b1的时候，同时也能计算b4，这样子就可以直接用GPU来进行加速提高训练速度。

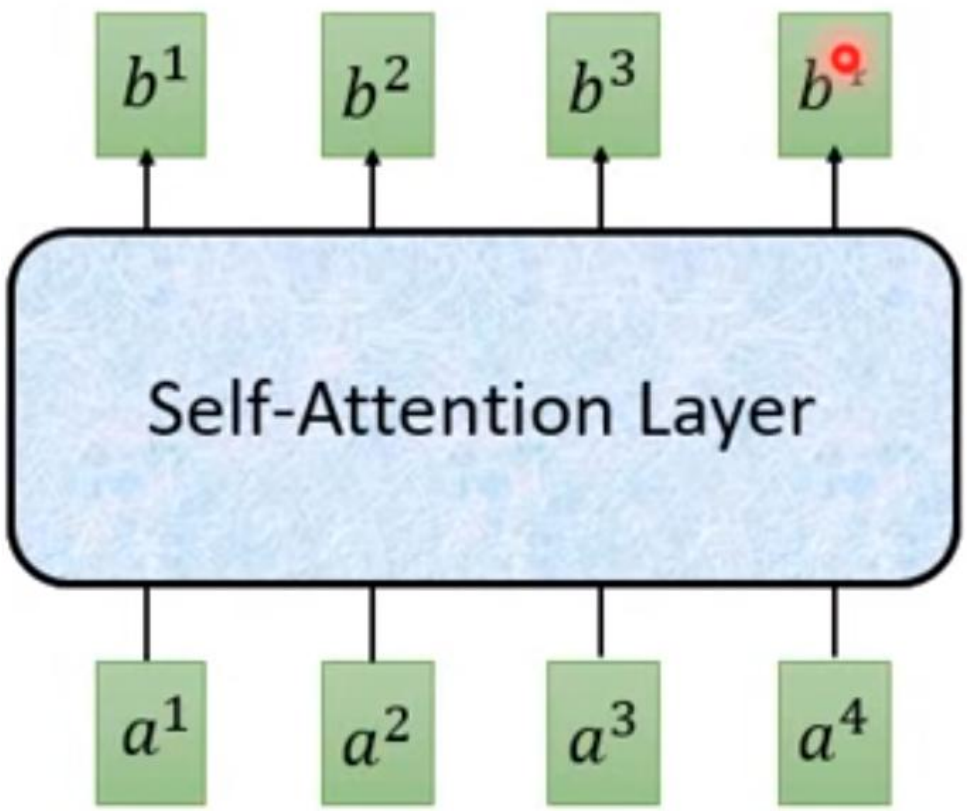
但是有个缺点就是如果句子很长，就是要更多层的CNN，这样子显然会使得模型更难train。

Attention Is All You Need

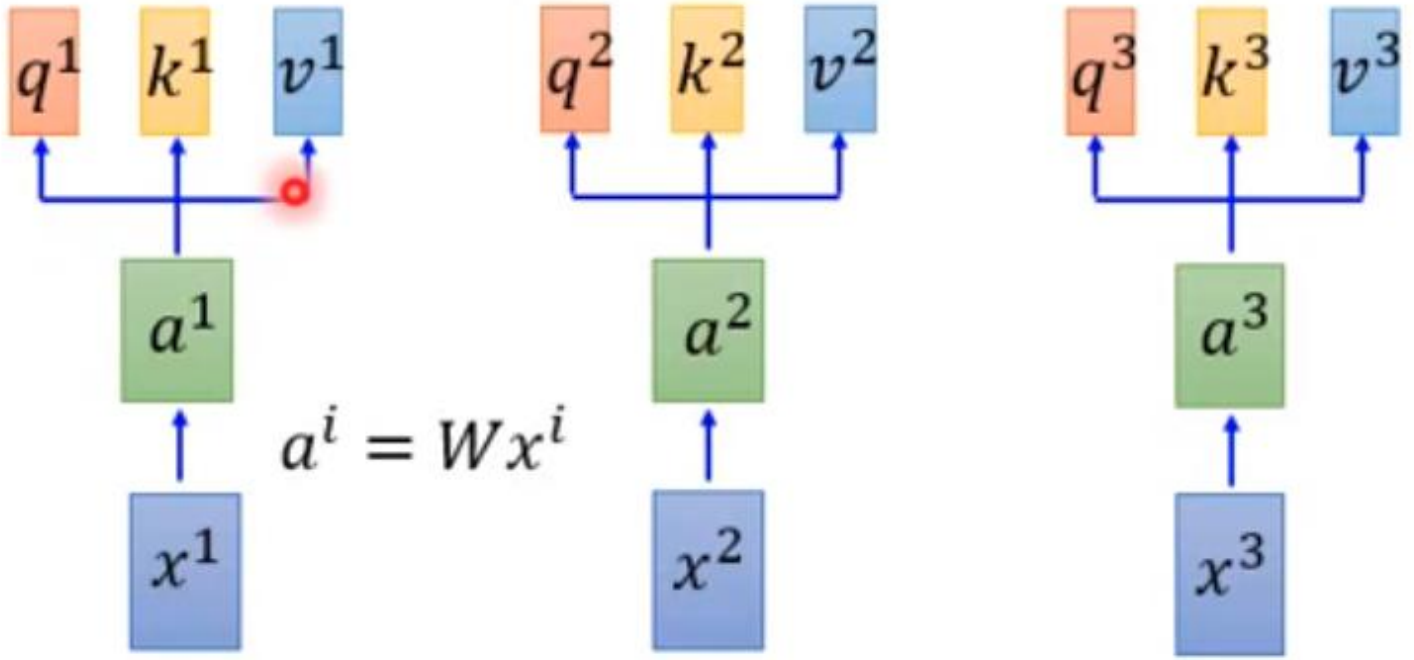
https://arxiv.org/pdf/1706.03762.pdf

于是，就有人提出用Self-attention模型。

而且b1到b4也是可以并行计算的



1. 输入x1到x3先进行一个embedding，
   得到ai
2. 对ai分别乘上qi, ki, vi，得到3个
   matrix： Q, K, V



$$\underbrace{q^1\ q^2\ q^3\ q^4}_{Q} = W^q \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{k^1\ k^2\ k^3\ k^4}_{K} = W^k \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{v^1\ v^2\ v^3\ v^4}_{V} = W^v \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

q: query (to match others)

$$q^i = W^q a^i$$

k: key (to be matched)

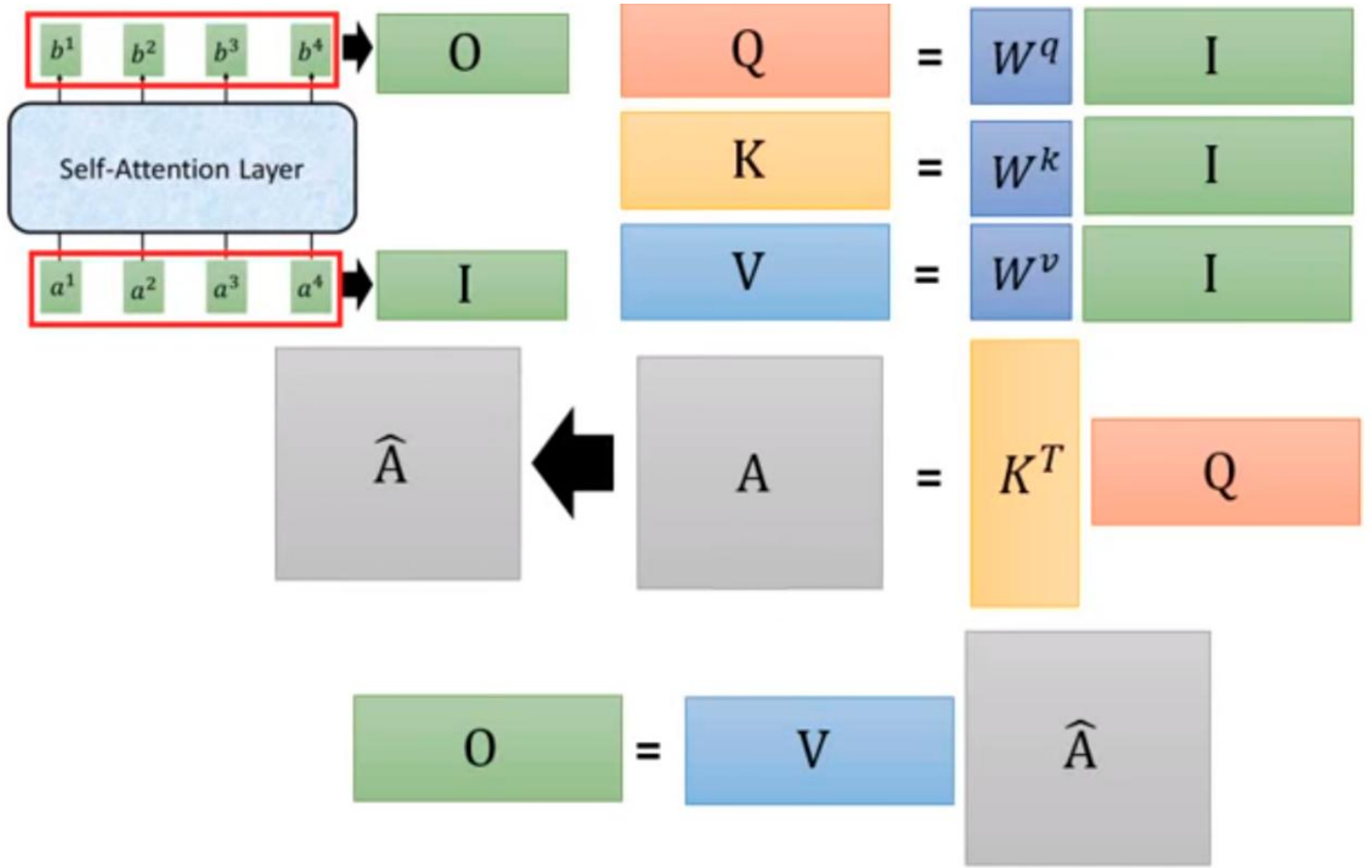
$$k^i = W^k a^i$$

v: information to be extracted

$$v^i = W^v a^i$$

通过KT乘上Q，得到A，矩阵A代表的就是sequence每个位置之间的Attention。

最终，我们得到了self-attention layer的Output O = VA_hat



Attention Visualizations

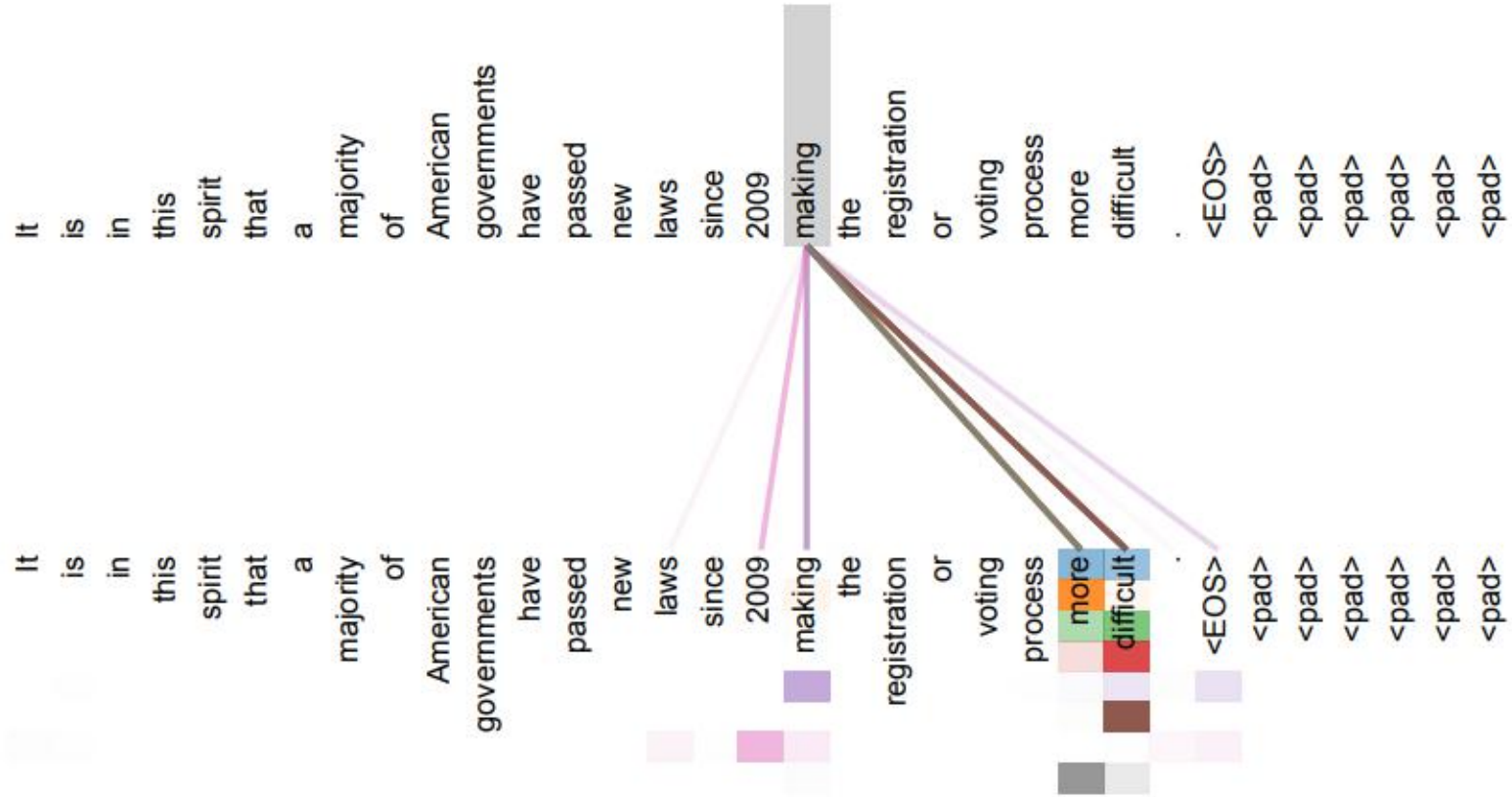


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

Multi-Head Attention

https://arxiv.org/pdf/1706.03762.pdf

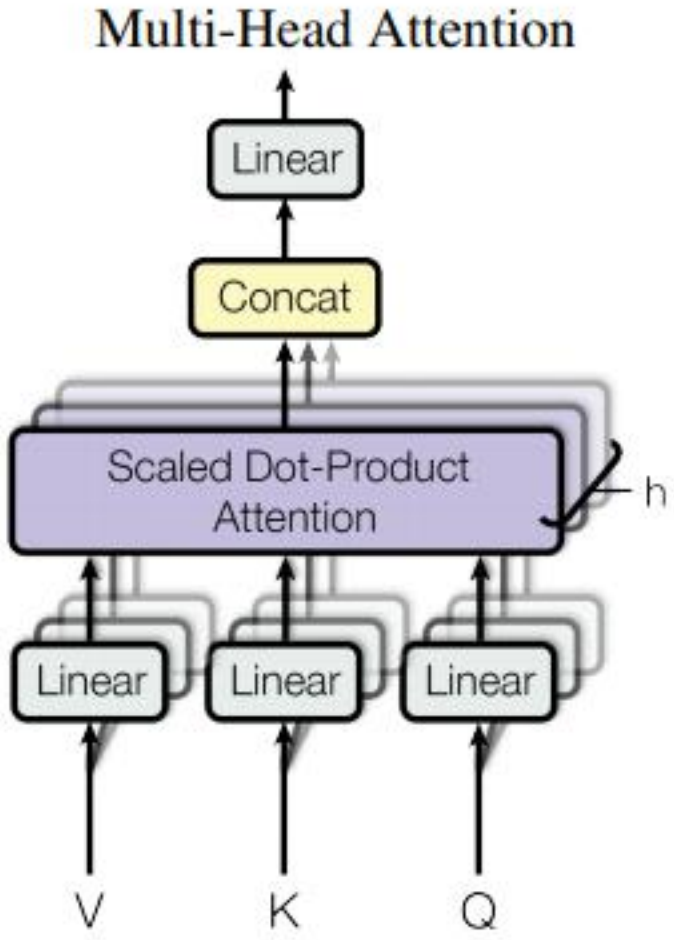
在Transformer 里面用到的是 self-attention 的变型，简单来说就是

self-attention的加强版。

在self-attention里面得到Q,K, V后，通过线性变换得到多个Qi, Ki, Vi

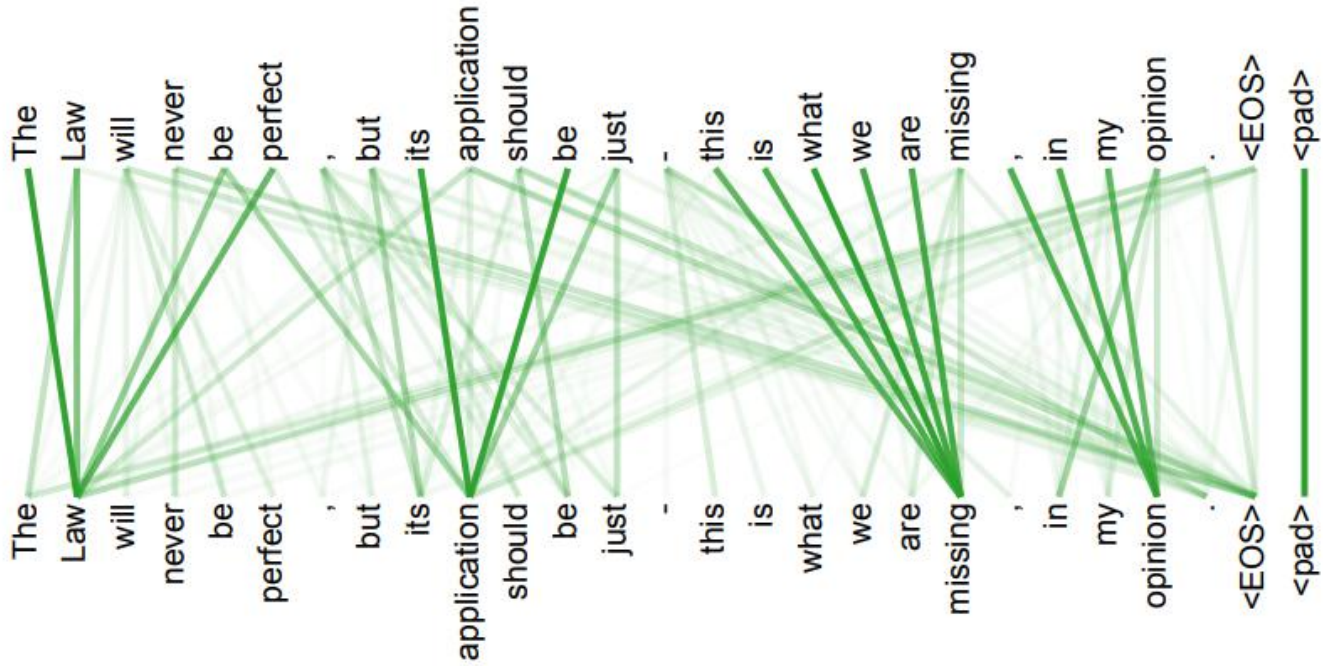
然后通过Scaled Dot-Product Attention 得到新的

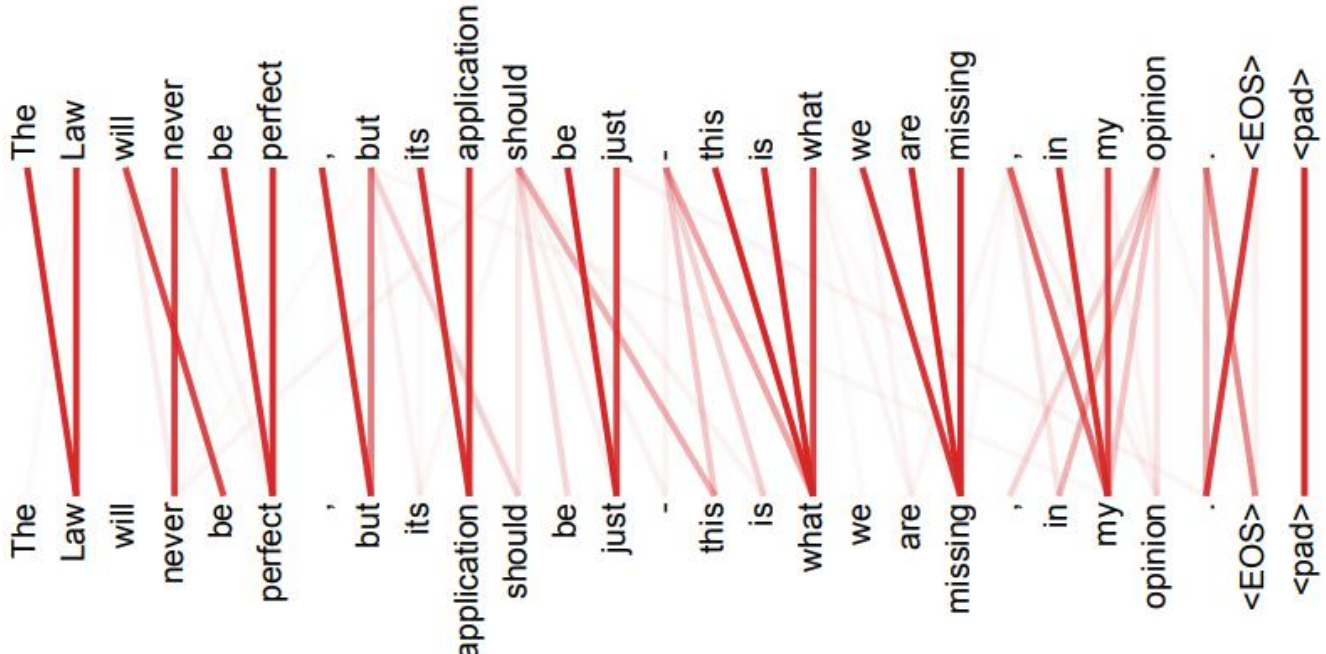
Attentic $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$



Multi-Head Attention

Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.



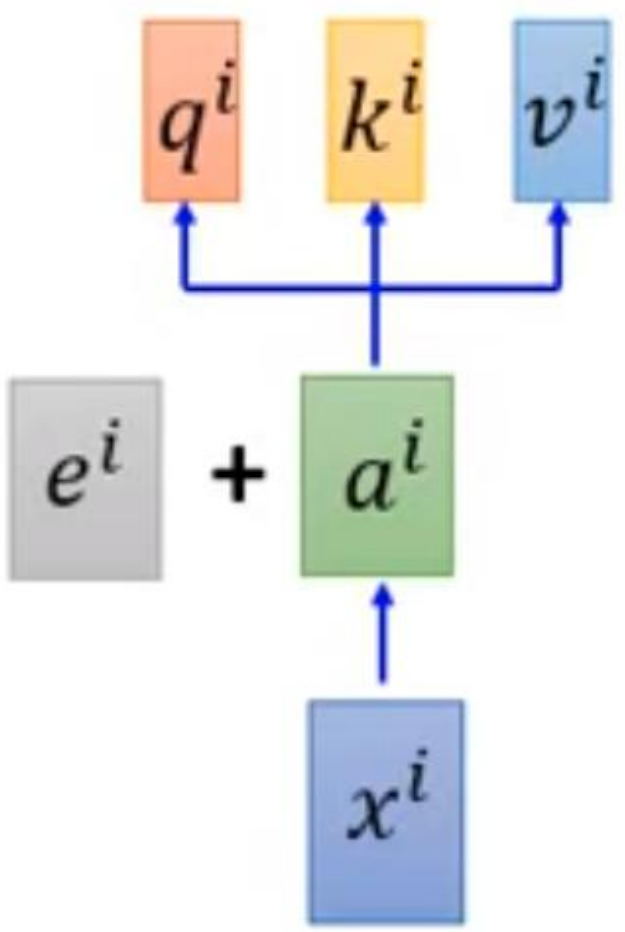


Positional Encoding

https://arxiv.org/pdf/1706.03762.pdf

在这之前我们的model是可以考虑sequence中不同位置之间的关系，但是并不能考虑到当前的token是处于句子的头部，中间还是尾部。

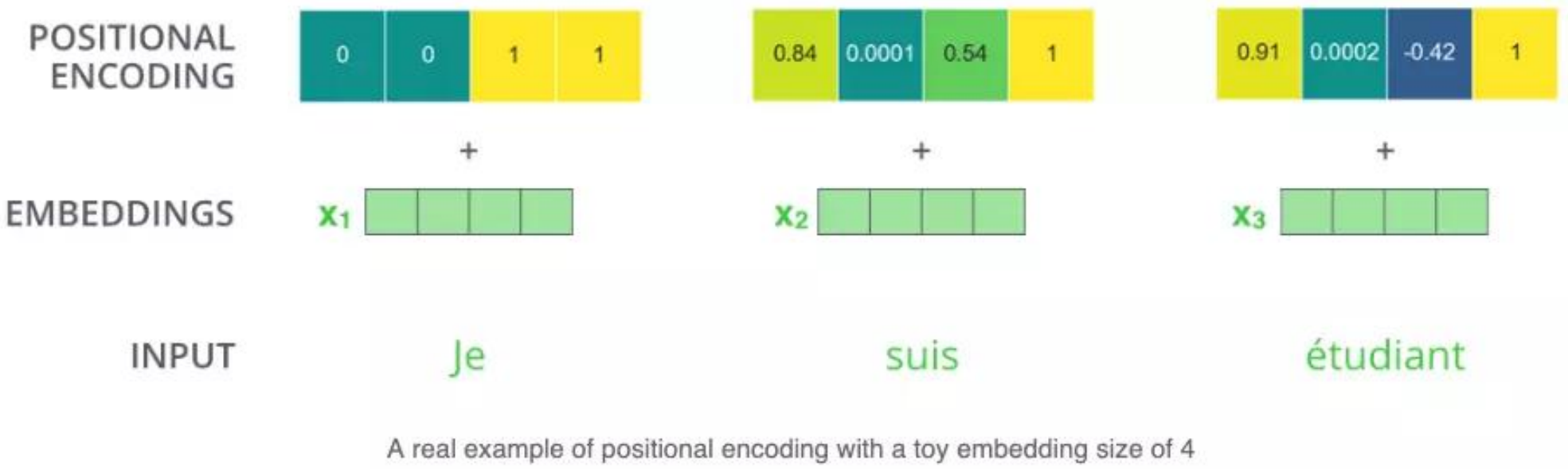
于是Paper里面加入了Positional Encoding的概念，在输入X进行embedding后加上这个位置向量



Positional Encoding

https://arxiv.org/pdf/1706.03762.pdf

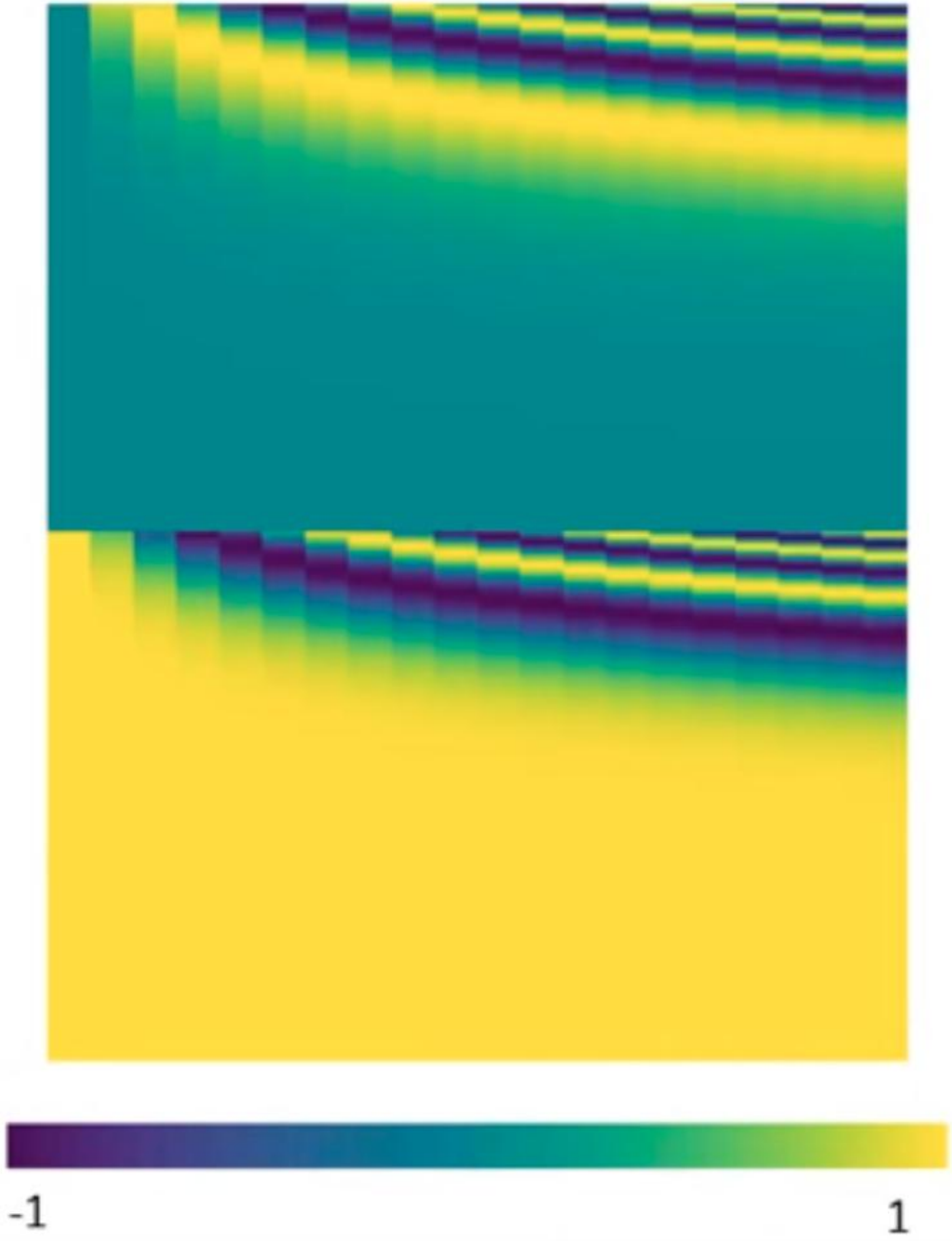
假设输入x的维度为4，那么实际的positional encodings如下所示：



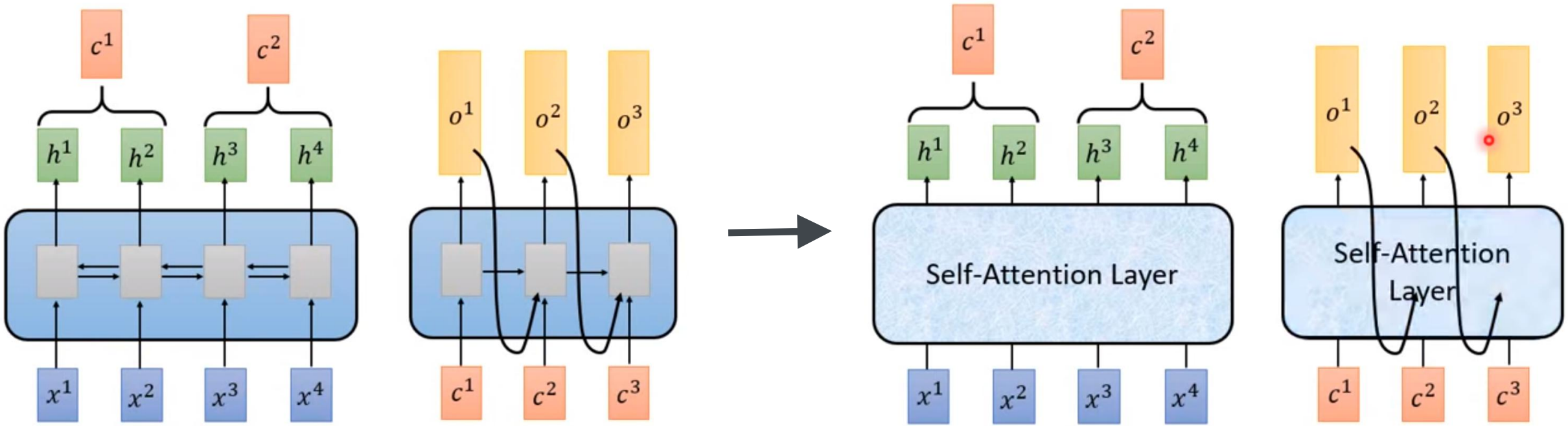
A real example of positional encoding with a toy embedding size of 4

Positional Encoding

最终从句子开头到结束，把每个每个position向量画出来会是这样子



Seq2Seq with Attention

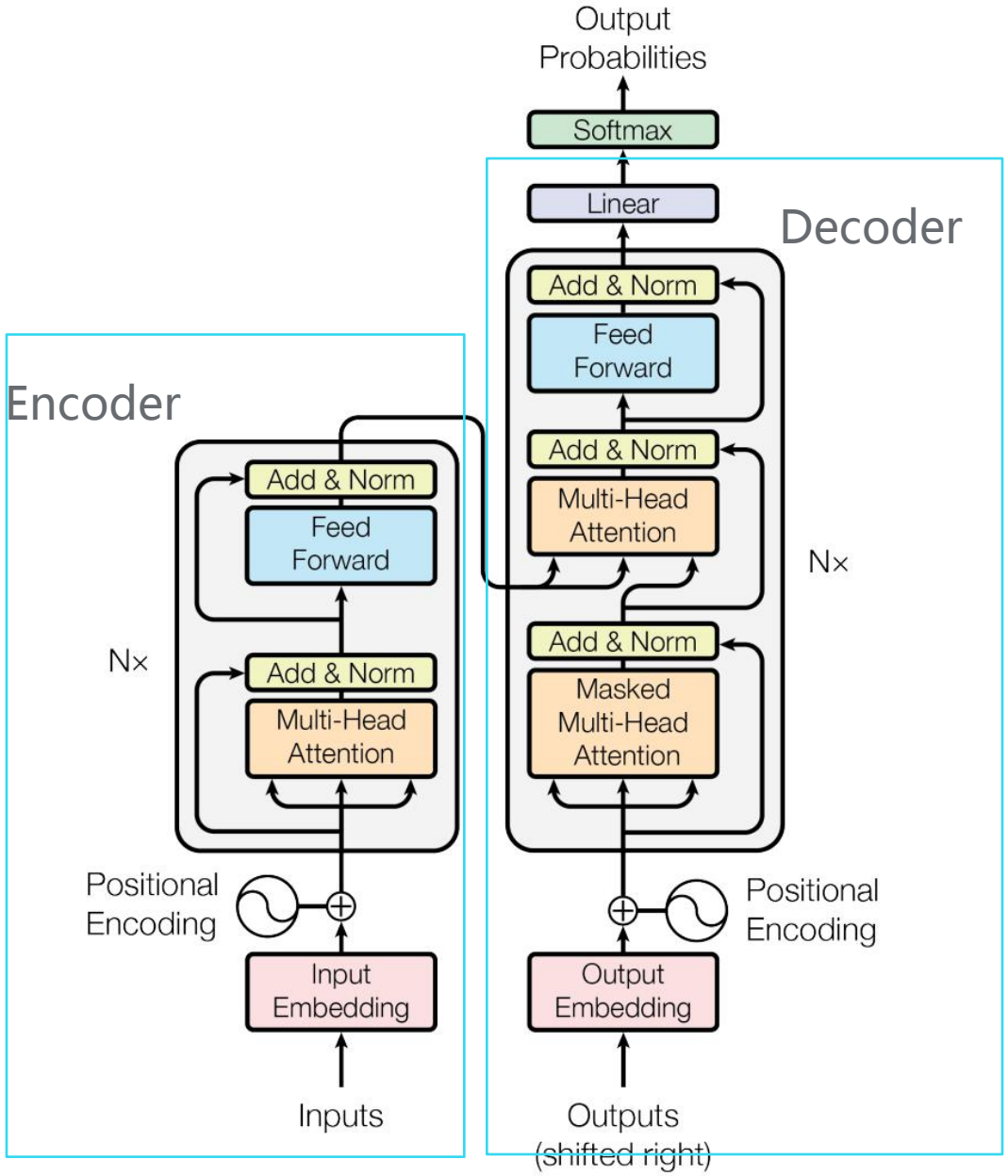


Transformer

在原Paper里面给出来的Transformer，如右图所示:

左边部分就相当于原来seq2seq的Encoder部分

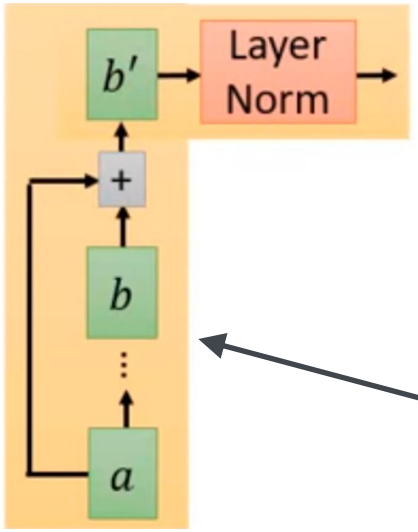
右边部分就相当于原来seq2seq的Decoder部分

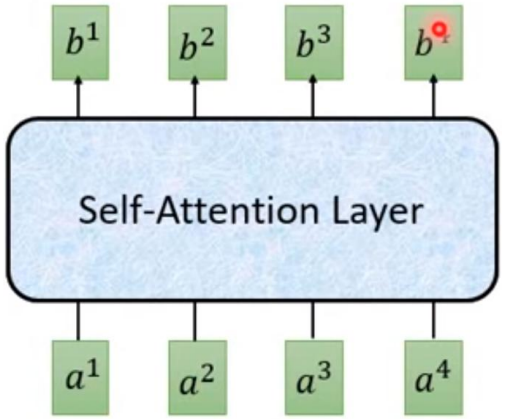


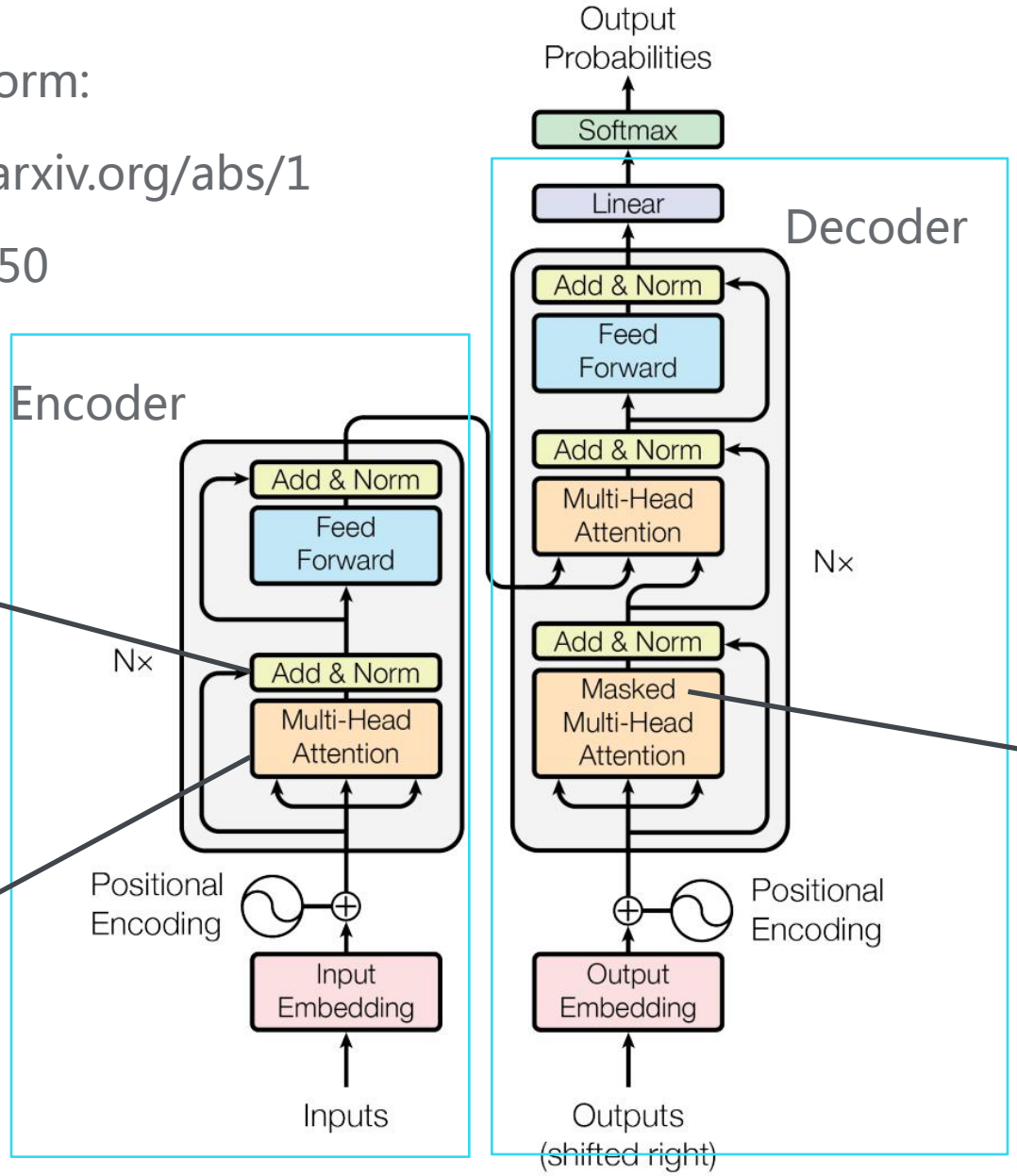
Transformer

Layer Norm:

https://arxiv.org/abs/1607.06450







用Encoder生成的Sequence

Thanks

Assignment

1. Seq2Seq

2. Github Source Code

   - https://github.com/fortyMiles/neural-machine-translation

3. https://arxiv.org/abs/1706.03762

4. http://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors.pdf

5 https://allennlp.org/elmo

6. https://github.com/tensorflow/tensor2tensor