词汇量可以被用于探究从文本中获取知识

成年人词汇量—30000-100000

成熟的演讲者在日常交流中使用的方式，是在生命早期通过与照顾者和同伴的口语交流获得的，通常在开始正规学校教育之前，一般在2000左右——活跃词汇

正常小孩需要在20岁之前每天学习7-10个单词才可以达到词汇量要求，这样的增长肯定不来自课堂教授，因为效率很低

最可能解释就是阅读带来了词汇量，一些研究证明了阅读时间和词汇多样性可能会带来合理的词汇增长速度

这些研究激发了**分布假设distributional hypothesis**，这是一种观点，即即使在现实世界中没有任何基础，也可以学习我们所说的词语含义，仅仅基于我们在生活中遇到的文本的内容。这样的知识基于词汇共现

预训练：通过处理大量的文本来学习单词或句子的某种意义表示。这个过程得到**预训练语言模型**

10.1 Self-Attention Networks: Transformers

相比RNN，都可以处理长文信息，不需要循环连接，可以并行，节省时间

Transformers：将向量$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$映射到$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$。

自注意力允许网络直接从任意大的上下文中提取和使用信息，而不需要像RNN通过中间循环连接传递信息

自注意力两个原则：

- 获得每个输出项时，模型可以获取当前和之前的所有输入项——可以用来构建语言模型和自回归生成
- 每个输出项的计算和其他项的计算是独立的——可以在训练和推理模型时轻松并行

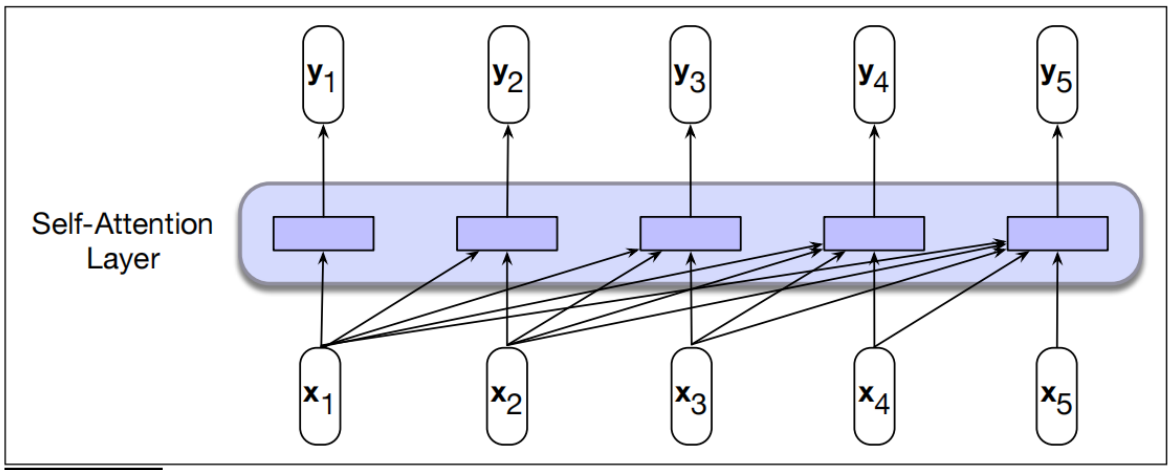

Figure 10.1 Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.

基于注意力的方法的核心

注意力的核心是将感兴趣的项与其他项的集合进行比较，以揭示它们在当前上下文中的相关性。对于**自注意力**来说，对比的是给定序列中的其他元素。对比的结果被用来为当前输入计算输出。

最简单的例子：要计算$\boldsymbol{y}_3$，计算三个分数 $\boldsymbol{x}_3 \cdot \boldsymbol{x}_1$ $\boldsymbol{x}_3 \cdot \boldsymbol{x}_2$ $\boldsymbol{x}_3 \cdot \boldsymbol{x}_3$ 之后将这三个分数softmax归一化得到权重α_{ij}，即每个输入与当前输入的相关性。最后利用这个权重对每个输入向量做加权计算得到输出

$$score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$
$$\alpha_{ij} = softmax(score(\boldsymbol{x}_i, \boldsymbol{x}_j)) \; \forall j \leq i$$
$$\boldsymbol{y}_i = \sum_{j \leq i} \alpha_{ij} \boldsymbol{x}_j$$

注意力核心步骤：

1. 通过上下文中相关元素的相互对比得到相关分数
2. 分数标准化得到概率分布
3. 利用概率分布计算加权和

Transformer的改进

考虑**每个输入编码**在注意力过程中发挥的**不同作用**

query：当前的注意力焦点，即$\boldsymbol{x}_3$

key：将它作为先前输入的角色与当前的注意力焦点进行比较，即$\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$

value：用于计算当前注意力焦点输出的值

为了捕捉这三种不同作用，引入三个权重矩阵$\boldsymbol{W^Q}, \boldsymbol{W^K}, \boldsymbol{W^V}$

$$\boldsymbol{q}_i = \boldsymbol{W^Q} \boldsymbol{x}_i; \boldsymbol{k}_i = \boldsymbol{W^K} \boldsymbol{x}_i; \boldsymbol{v}_i = \boldsymbol{W^V} \boldsymbol{x}_i$$

定义：所有的单个输入输出向量都是$1 \times d$ 权重矩阵$d \times d_k$或$d \times d_v$ key和query的维度d_k value的维度d_v

对于单头注意力来说，$d_k = d_v = d = 1024$

$$score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{q}_i \cdot \boldsymbol{k}_j$$
$$\boldsymbol{y}_i = \sum_{j \leq i} \alpha_{ij} \boldsymbol{v}_j$$

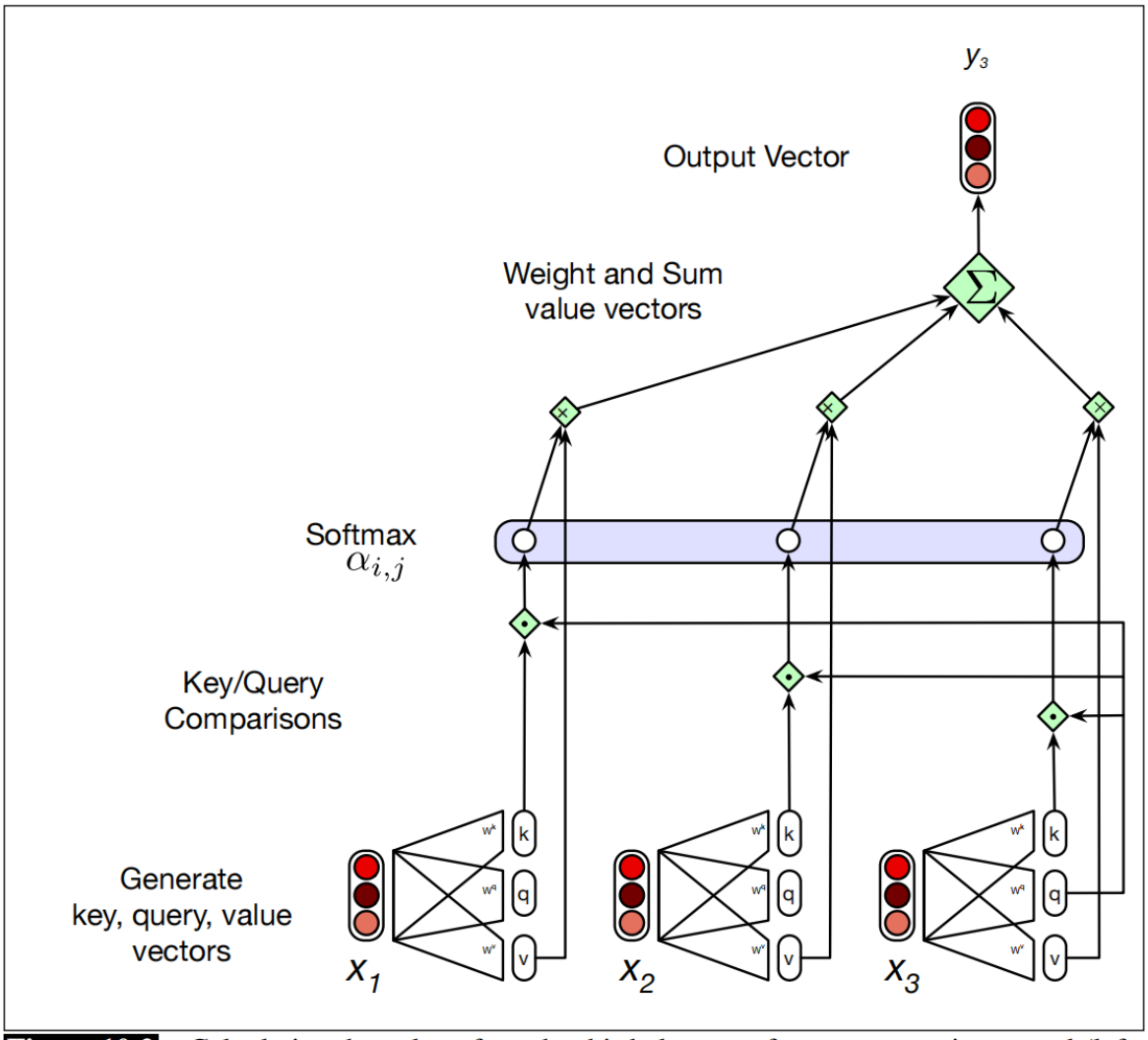

Figure 10.2 Calculating the value of $\mathbf{y}_3$, the third element of a sequence using causal (left-to-right) self-attention.

点积结果可能很大或很小，导致最后产生数值溢出问题，点积结果需要被约束。

$$score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}}$$

由于每个输出的计算相互独立，我们可以将向量拼接为$N \times d$的矩阵$\boldsymbol{X}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{\boldsymbol{Q}}; \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{\boldsymbol{K}}; \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{\boldsymbol{V}}$$

$$SelfAttention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V}$$

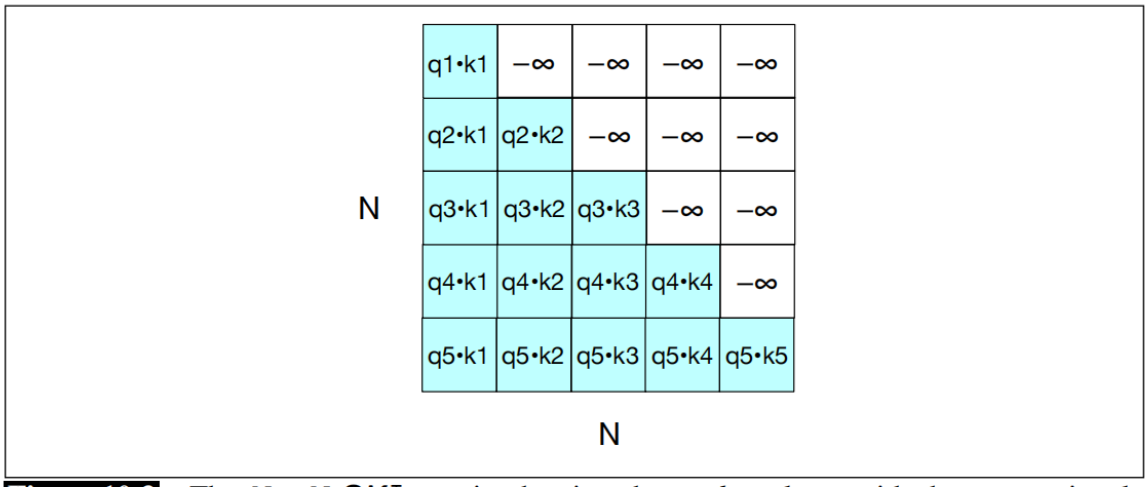

Figure 10.3 The $N \times N$ $\mathbf{QK}^\intercal$ matrix showing the $q_i \cdot k_j$ values, with the upper-triangle portion of the comparisons matrix zeroed out (set to $-\infty$, which the softmax will turn to zero).

计算过程是$O(N^2)$的，时间空间开销大，因此一般需要限制长度

Transformer块

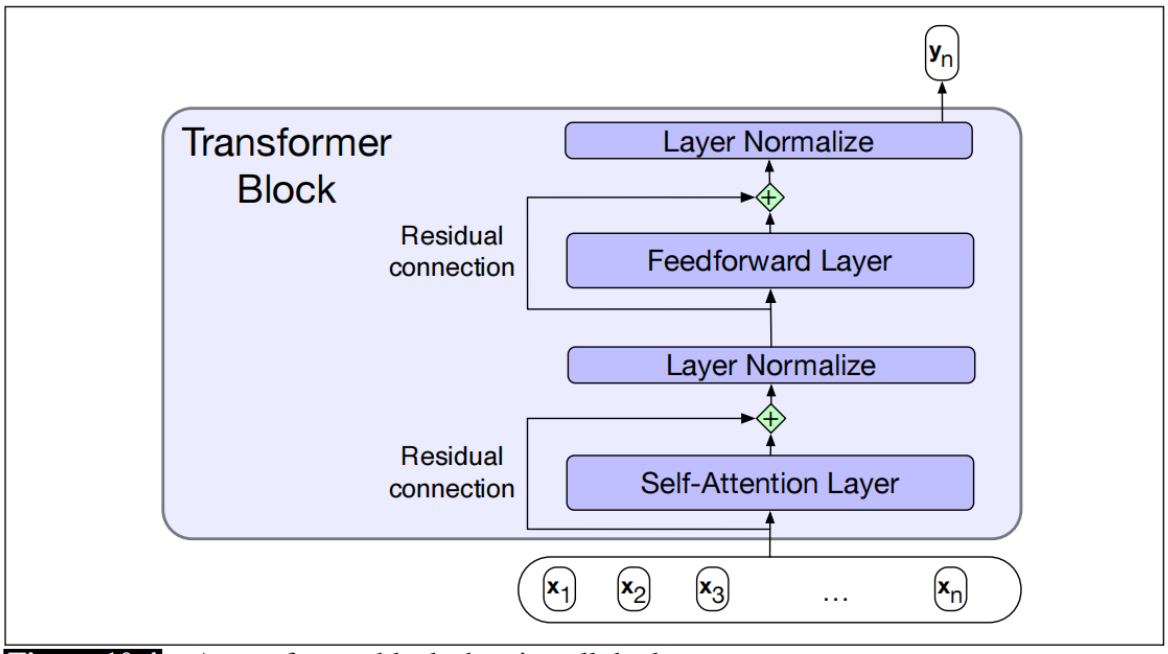

Figure 10.4 A transformer block showing all the layers.

$$\boldsymbol{z} = LayerNorm(\boldsymbol{x} + SelfAttention(\boldsymbol{x}))$$
$$\boldsymbol{y} = LayerNorm(\boldsymbol{z} + FFN(\boldsymbol{z}))$$

前馈神经网络FFN

一个全连接层，过一个激活函数，再过一个全连接层

可以转换为QKV的记忆形式

$$\boldsymbol{a} = Activation(\boldsymbol{q} \cdot \boldsymbol{K} + \boldsymbol{b}_k)$$
$$FFN(\boldsymbol{q}) = \boldsymbol{a} \cdot \boldsymbol{V} + \boldsymbol{b}_v$$

残差块

在深度网络中，残差连接是指将信息从较下层传递到较高层而不通过中间层的连接。允许激活的信息向前和向后传播时跳过一个层，可以提升学习效果，并让高层直接访问来自低层的信息

层规范化

将隐藏层的值保持在有利于基于梯度训练的范围内

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$$

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mu}{\sigma}$$

$$LayerNorm(\boldsymbol{x}) = \gamma\hat{\boldsymbol{x}} + \beta(\gamma和\beta为可学习参数)$$

多头注意力

一句话中不同词可能和其他的词在不同方面相关，比如不同的句法关系、语义关系和语篇关系。因此Transformer提出了用同一层中多个自注意力块来学习不同的关系，每个自注意力块是一个头

假设有h个头，现在每一层得到了h个形状为$N \times d_v$的矩阵，将这些矩阵拼接得到大矩阵，之后使用一个$hd_v \times d$形状的线性投射层将这个大矩阵转换为原来的输出矩阵大小即$N \times d$

$d_k = d_v = 64$

$$\mathrm{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q\ ;\ \mathbf{K} = \mathbf{X}\mathbf{W}_i^K\ ;\ \mathbf{V} = \mathbf{X}\mathbf{W}_i^V$$

$$\mathbf{head}_i = \mathrm{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$

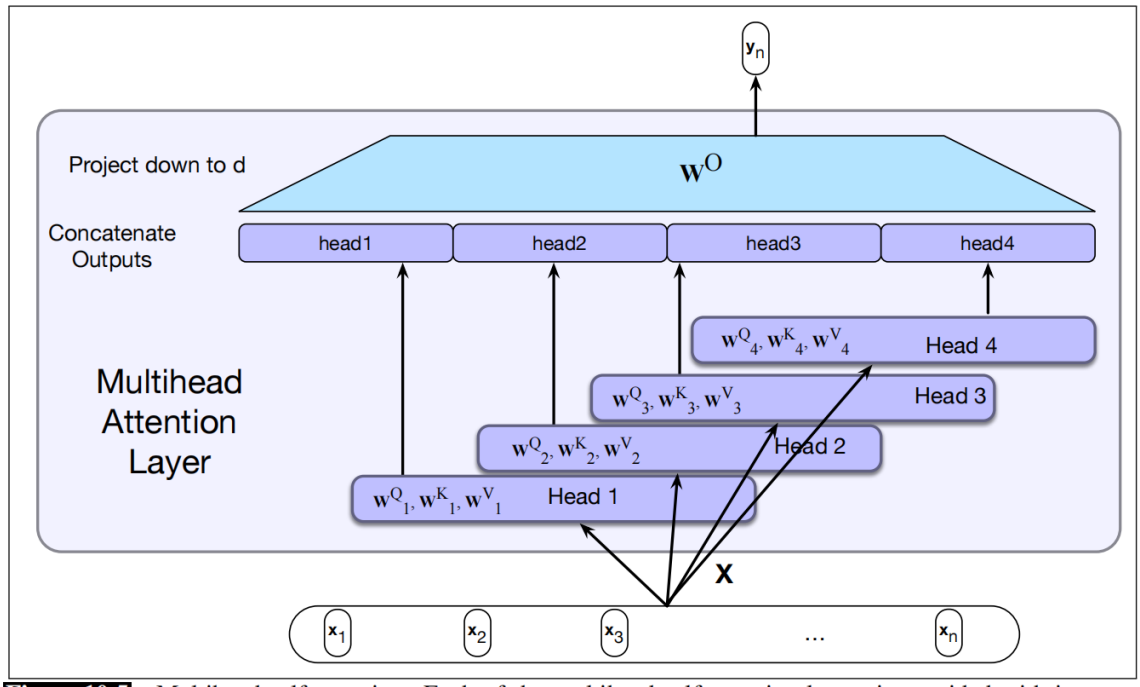

Figure 10.5 Multihead self-attention: Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices. The outputs from each of the layers are concatenated and then projected down to d, thus producing an output of the same size as the input so layers can be stacked.

建模词序：位置编码

目前的架构中，对于每个注意力焦点来说，前面词的词序对结果没有任何影响

最简单的方式是对每个位置的输入编码加入一个和位置相关的编码

一种方法：像对待每个词语一样给每个位置一个初始随机编码用于后续训练。但可能会出现初始位置和后续位置出现次数不一致，导致训练不平衡。

因此使用**静态函数**可能更好。初始Transformer论文中使用了三角函数的组合

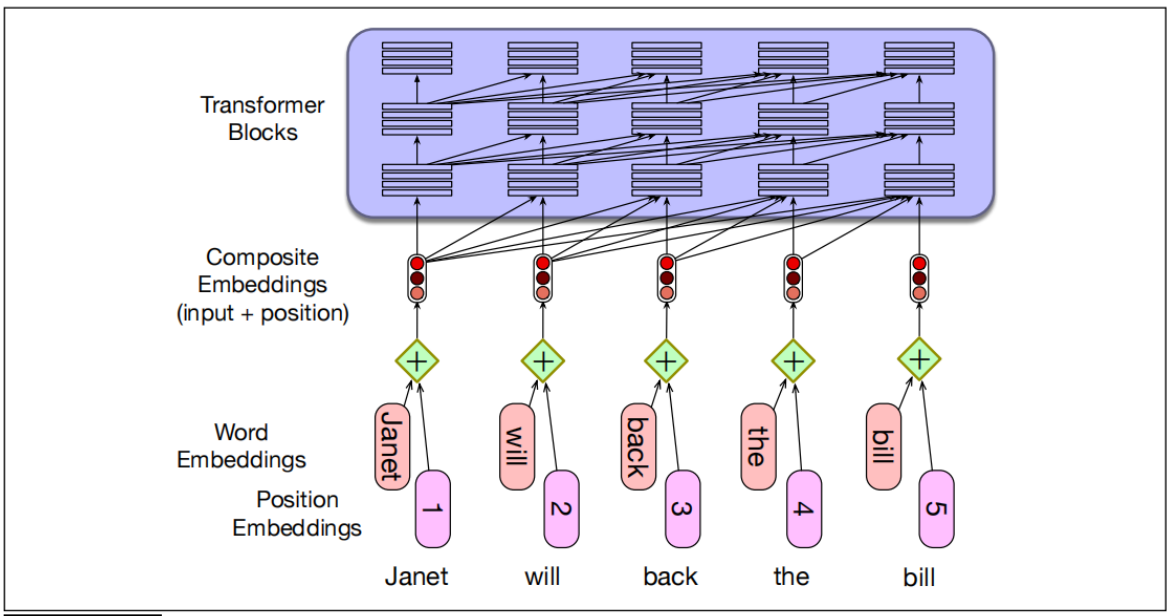

Figure 10.6 A simple way to model position: simply adding an embedding representation of the absolute position to the input word embedding to produce a new embedding of the same dimenionality.

10.2 Transformer as Language Models

强制教学思想，每次使用真实的过去token序列进行下一词预测

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

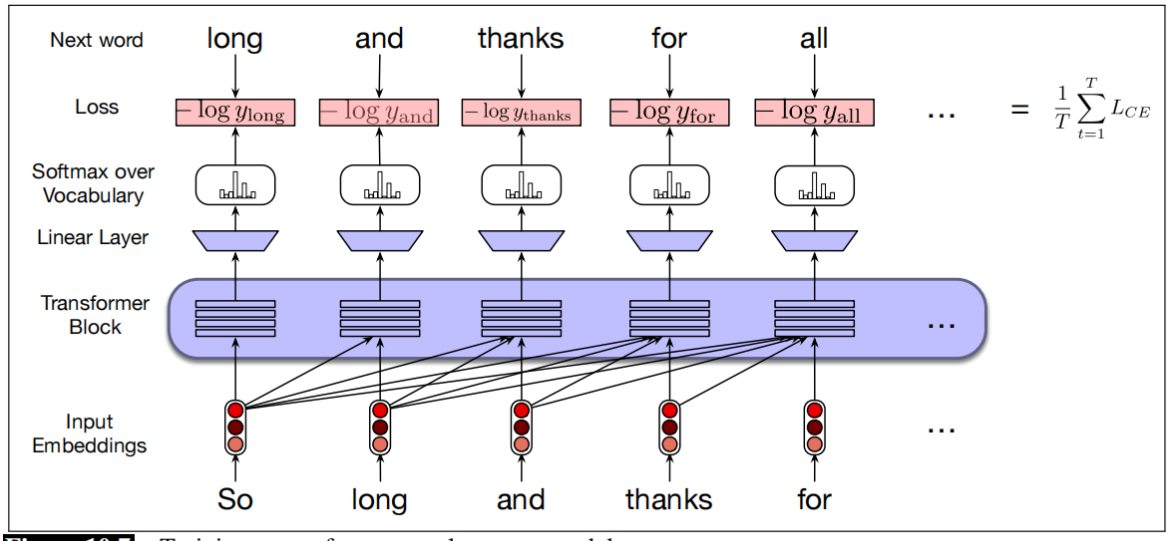

Figure 10.7 Training a transformer as a language model.

从`<s>`开始，不断地根据概率分布选择下一词，再根据之前做出的选择选择新地词，直到设置的句子上限或者遇到了`</s>`：自回归生成或因果性语言模型生成（autoregressive generation or causal LM generation）

10.3 Sampling

TBD: nucleus, top k, temperature sampling.

10.4 Beam Search

贪心搜索可能达不到最优，因此需要束搜索

束宽度为k，始终保持**概率最高的k条路径**

最开始选取k个概率最高的词汇，称为猜想**hypotheses**，一个猜想包含截止目前的**词语序列和序列概率**。下一个时间步得到了k*V个猜想，我们保留其中最高的k个猜想

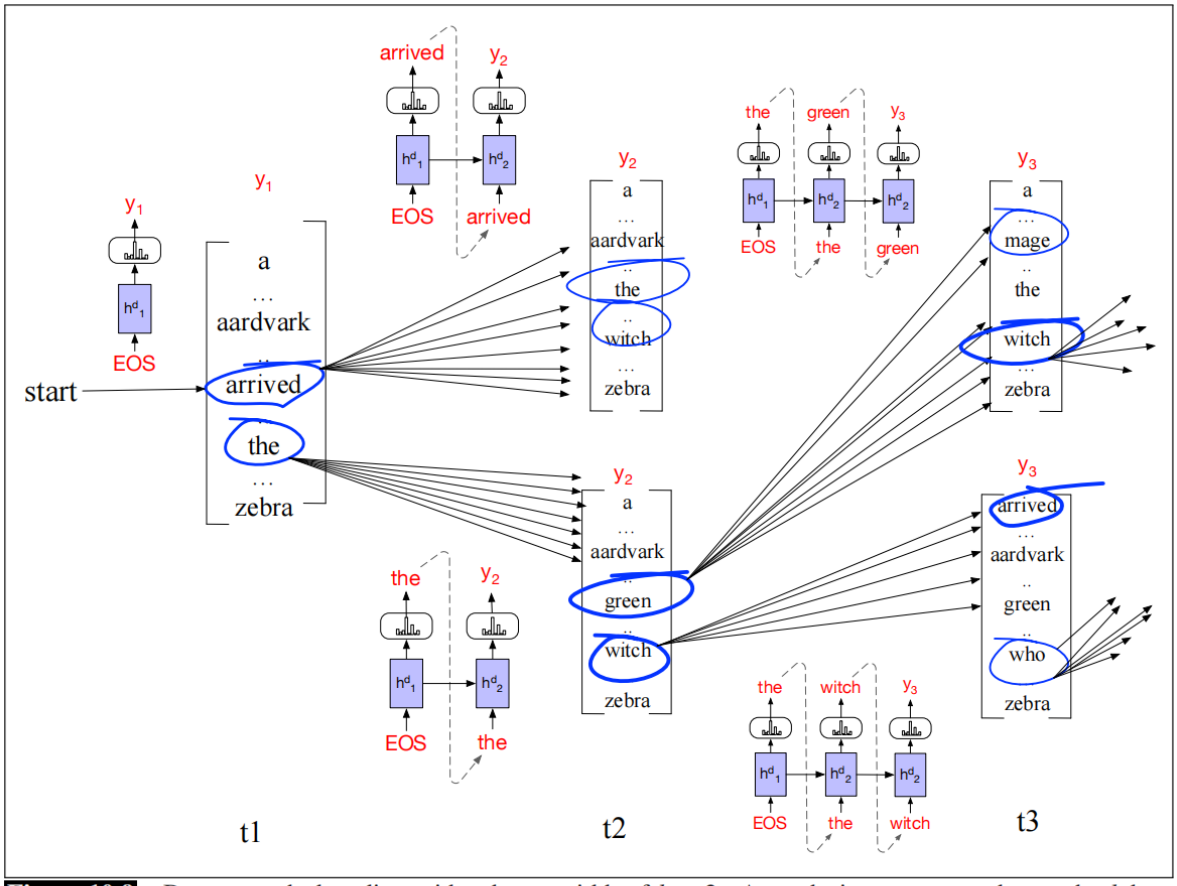

Figure 10.9 Beam search decoding with a beam width of $k = 2$. At each time step, we choose the k best hypotheses, compute the V possible extensions of each hypothesis, score the resulting $k * V$ possible hypotheses and choose the best k to continue. At time 1, the frontier is filled with the best 2 options from the initial state of the decoder: *arrived* and *the*. We then extend each of those, compute the probability of all the hypotheses so far (*arrived the*, *arrived aardvark*, *the green*, *the witch*) and compute the best 2 (in this case *the green* and *the witch*) to be the search frontier to extend on the next step. On the arcs we show the decoders that we run to score the extension words (although for simplicity we haven't shown the context value c_i that is input at each step).

之后一直推进下去，直到遇到了`</s>`，某条路径就结束，束宽度减一

$$
\begin{aligned}
score(y) &= \log P(y|x) \\
&= \log\left(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)...P(y_t|y_1,\ldots,y_{t-1},x)\right) \\
&= \sum_{i=1}^{t} \log P(y_i|y_1,\ldots,y_{i-1},x)
\end{aligned}
$$

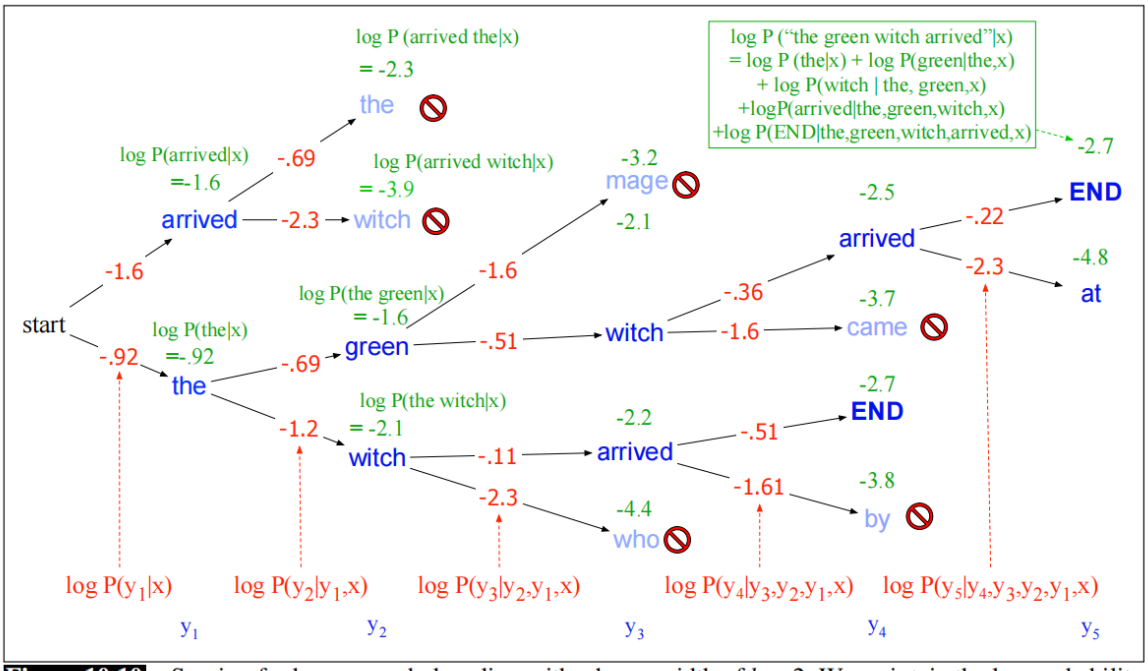

Figure 10.10 Scoring for beam search decoding with a beam width of $k = 2$. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step.

问题：可能会给更长的字符串更低的概率

解决方法：长度规范化

$$score(y) = -\log P(y|x) = \frac{1}{T}\sum_{i=1}^{t} -\log P(y_i|y_1,\ldots,y_{i-1},x)$$

10.5 Pretraining Large Language Models

TBD: corpora, etc.

10.6 Language Models for Zero-shot Learning

TBD: How to recast NLP tasks as word prediction, simple prompting examples (and then to be continued in Chapter 12)

10.7 Potential Harms from Language Models

产生有毒语言。许多无毒提示可能会产生仇恨言论和虐待，或生成的句子显示出对少数族裔身份的消极态度，如黑人或同性恋。

训练数据的分布有偏差。大语言模型使用了很多从被禁止的网站上爬取的有毒的数据比如Reddit。除此之外，训练数据可能来源于发达国家的年轻男性。这种有偏见的样本很可能会偏离那些未被充分代表的人口的观点或主题。

语言模型也可以作为**生成错误信息、网络钓鱼、激进化和其他有害社会活动**的文本的工具

最后，还有一些重要的**隐私问题。**语言模型和其他机器学习模型一样，可以泄露有关其训练数据的信息。因此，不法分子可以从语言模型中提取个体训练数据短语，如个人的姓名、电话号码和地址。如果大型语言模型在电子健康记录（EHRs）等私人数据集上进行训练，这将是一个问题。

减轻这些问题是NLP需要但没有解决的问题。**在无毒语料上的额外预训练**似乎稍微降低了模型生成有毒语言的可能性

分析用于**预训练大型语言模型的数据**对于理解生成过程中的毒性和偏见以及隐私非常重要，这使得语言模型包括**数据表或模型卡**非常重要，以提供用于训练它们的语料库的**完整可复制**信息