中国农业大学
China Agricultural University

人工智能——自然语言处理

胡标

目录
Contents

什么是自然语言处理

- NLP, Natural Language Processing：用机器处理人类语言的理论和技术研究在人与人交际中以及人与计算机交际中的语言问题的一门学科。
- NLP要研制表示语言能力和语言应用的模型，建立计算框架来实现这样的语言模型，提出相应的方法来不断完善这样的模型，并根据语言模型设计各种实用系统，以及对这些系统的评测技术。

------ Bill Manaris,《从人机交互的角度看自然语言处理》

- 其它名称
  - 自然语言理解(Natural Language Understanding)
  - 计算语言学(CL, Computational Linguistics)
  - 人类语言技术(Human Language Technology)

NLP的历史

20世纪50年代起步——机器翻译

- 利用计算机自动地将一种自然语言翻译为另外一种自然语言
- “I like Beijing Tiananmen Square” ⟷ “我爱北京天安门”

50-70年代——模式匹配

- 用户输入→题库查找答案→返回结果

90年代至今——基于统计

- 互联网的发展提供海量的自然语言供研究应(社交对话、邮件、文章等等)，方便学者和研究人员基于大量语料基础之上构建自然语言处模型
- 又称“经验主义”语言模型，使用大规模真实语料库训练，得出结果，避开一些歧义、语境等导致的技术难题

- **语义**
  - 句子一：我们把香蕉给猴子，因为它们饿了
  - 句子二：我们把香蕉给猴子，因为它们熟透了
- **歧义**
  - Time flies like an arrow 怎么翻译?
    - time 名词： “时间” 动词： “测定，拨准”
    - flies 名词： “苍蝇” 动词： “飞”
    - like 动词： “喜欢” 介词： “像”
  - 时间像箭一样飞驰 还是 那些像箭一样的苍蝇
- 分析过程：词意→句意→语义（语境），语义处理相对比较难，需要联系上下文，语境，还有一些情感色彩，这是自然语言处理中的难点

NLP实际问题

□ 在基于文本的NLP应用领域，机器翻译可能受到深度学习的影响最大

- 当前，在实际应用中表现最佳的机器翻译系统是基于深度神经网络的模式，例如，最近，谷歌发布了机器翻译领域最强的BERT的多语言模型。BERT在机器阅读理解顶级水平测SQuAD1.1中表现出惊人的成绩:两个衡量指标上全面超越人类，而且在11种不同NLP测试中同样给出了最好的成绩，其中包括将GLUE基准推至80.4%，MultiNLI准确度达到86.7%等。

□ 在将深度学习应用于NLP问题的过程中，出现的两个重要技术突破是序列到序列学习和注意力建模

- 序列到序列学习引入了一个强大的思想，即利用循环网络以端到端的方式进行编码和解码。虽然注意力建模最初是为了解决对长序列进行编码的困难，但随后的发展显然扩展了它的功能能够对任意两个序列进行高度灵活的排列，且可以与神经网络参数一起进行学习。

目录
Contents

语料

- **语料，即语言材料，包括文本和语音。**语料库(corpus)即语料的集合也可称为自然语言处理领域的数据集，是为一个或者多个应用目标而专门收集的，有一定结构的、有代表的、可被计算机程序检索的、具有一定规模的语料集合。本质上讲，语料库实际上是通过对自然语言运用的随机抽样，以一定大小的语言样本来代表某一研究中所确定的语言运用的总体。
- 语料库具备**三个显著特点**
  1. 存放的是在语言的实际使用中真实出现过的语言材料
  2. 以电子计算机为载体承载语言知识的基础资源，但并不等于语言知识
  3. 真实语料需要经过加工(分析和处理)，才能成为有用的资源

语料库的分类(多种划分方式)

- 语言种类
  1. 单语语料库(Monolingual Corpus)
  2. 双语/多语语料库(Bilingual/Multi-lingual Corpus)
- 加工深度
  1. 非标注语料库(Non-Annotated Corpus)
  2. 标注语料库(Annotated Corpus)
- 用途
  - 通用语料库(General Corpus)
  - 专用语料库(Specialized Corpus)

语言知识库

- 语言知识库包括词典、词汇知识库、句法规则库、语法信息库、语义概念等各类语言资源，是自然语言处理系统的必要组成部分。
- 语言知识库可分为两类，一类是显性语言知识表示库，如词典、规则库、语义概念库等可以采用形式化结构描述；另一类是隐式的语言知识库，这类语料库的主体是文本，即语句的集合。
- 著名的显性语言知识库有：
  1. 词网
  2. 北京大学综合性语言知识库
  3. 知网

词网(WordNet)

- 词网(WordNet)是由美国普林斯顿大学认知科学实验室领导开发，由心理学家，语言学家和计算机工程师联合设计的一种基于认知语言学的英语词典。
- WordNet按照单词的意义组成一个“**单词的网络**”，WordNet将名词动词，形容词和副词分别组织成一个同义词的网络，每个同义词集合都代表一个基本的语义概念，并且这些集合之间也由各种关系连接，这些关系包括同义关系(synonymy)、反义关系(antonymy)、整体与部分关系(meronymy)和继承关系(entailment)等。
- 通俗地来说，WordNet是一个结构化知识库，它不仅包括一般的词典功能，还包括词的分类信息。

北京大学综合型语言知识库

- 北京大学综合型语言知识库(Peking University ComprehensiveLanguage Knowledge Base)，简称CLKB，由北京大学俞士汶教授领导建立。
- 该语言知识库从词、词组、句子、篇章各粒度和词法、句法、语义各层面进行语言资源的整理，涵盖了现代汉语语法信息词典、汉语短语结构规则库、现代汉语多级加工语料库(词语切分及词类标注)、多语言概念词典、平行语料库(英汉对照语句)、多领域术语库(英汉对照术语)。
- HowNet还是最著名的义原知识库，义原在语言学中被定义为最小的、不可再分割其语义的语言单位，例如:“人”虽然是一个非常复杂的概念，它可以是多种属性的集合体，但也可以把它看作一个义原，“男孩”“女孩”的义原都可归为“人”。HowNet通过对全部的基本义原进行观察分析并形成义原的标注集，然后再用更多的概念对标注集进行考核，从而建立完善的标注集。

知网(HowNet)

- 知网(HowNet)是由机器翻译专家董振东和董强创建的语言知识库。
- HowNet以汉语和英语的词语所代表的概念为描述对象，将概念与概念之间以及概念所具有的属性之间的关系构成一个网状的知识系统。知网所要反映的是概念的共性、个性，以及概念之间的关系，例如，对于“医生”和“人”是他们的共性，而“医生”的个性是“医治”的施者“患者” “患者”的个性是“患病”的经验者，“医生”和“患者”之间的关系是“医生”医治“患者”。
- 该知识库是目前国际上规模最大且获得广泛认可的汉语语言知识资源库。

目录
Contents

独热编码

- 在一个语料库中，给每个词编码一个索引，根据词索引建立词表，并进行独热(One-Hot)编码，下面通过一个例子进行解释，假设语料库有三句话：
  - 我爱中国
  - 我爸爸妈妈爱我
  - 爸爸妈妈爱中国
- 将上述语料库进行分词和编号得到长度为5的索引序列，也是该语料库的词表：
  - 0 我;　　1 爱;　　2 爸爸;　　3 妈妈;　　4 中国
- 其中每个单词都可以用One-Hot的方法表示，One-Hot是指一个词的One-Hot向量中只有该词对应索引位置的值为1，其他值都为0：
  - 我:[1,0,0, 0, 0] 爱:[0,1,0, 0,0]　爸爸:[0,0,1,0,0]　妈妈:[0,0,0, 1,0]　中国:「0,0.0,0,1]
- 对于上述语料库的第一个句子来说，可表示为一个向量序列:
  - 我爱中国 [[1,0, 0, 0, 0], [0, 1, 0,0, 0], [0, 0, 0, 0, 1]]

独热编码

- **优点**：原理简单、易于实现
- **缺点**：

1. 无法表示词之间的关系
   - 不同单词的One-Hot向量相互正交，即词与词之间是完全独立的，无法衡量不同词之间的关系。例如，“爸爸”和“妈妈”两词很明显关系紧密，而“爸爸”和“中国”则没有什么关联，但若采用One-Hot向量表示这些词之后，两对词之间的距离是相同的，无法计算词与词之间的关系。
2. 无法衡量不同词的重要程度
   - One-Hot向量只能反映某个词是否在句中存在，而并未表示出单词出现的频率无法衡量不同词的重要程度。
3. 稀疏向量，资源浪费
   - 当语料库非常大时，需要建立一个词表对所有单词进行索引编码，假设有100万个单词，每个单词就需要表示成100万维的向量，而且这个向量是很稀疏的，只有一个位置为1其他全为0，高维稀疏向量导致机器的计算量大并且造成了计算资源的浪费。

词袋表示

- 词袋(Bag of Words，BOW)表示，也称为计数向量表示，下面通过一个例子进行解释，假设使用上一小节的语料库:
  - 我爱中国
  - 我爸爸妈妈爱我
  - 爸爸妈妈爱中国
- 将上述语料库进行分词和编号得到长度为5的索引序列，也是该语料库的词表:
  - 0 我;　　1 爱;　　2 爸爸;　　3 妈妈;　　4 中国
- 然后对每句话进行向量转换，转换后每句话的维度为语料库词表的长度，每个索引位置的值为索引序列中对应词在该句话中出现的次数，以语录库的第一句话“我爱中国”为例，该语句的词袋表示为一个5维的向量，其中“我”"爱"“中国”三个词在该句话中分别出现一次，将这三个词对应的词表索引位置标为1，其他未出现的词标为0得到[1,1,0,0,1]。上述语料库的三条语句的BOW表示分别为:
  - 我爱中国 [1,1,0, 0, 1]，我爸爸妈妈爱我 [2, 1, 1, 1, 0]，爸爸妈妈爱中国 [0, 1 1 1 1]

词频-逆文档频率

- 词频-逆文档频率(Term Frequency-Inverse Document Frequency, TF-IDF)的核心思想是:若一个词在一篇文章中出现次数较多且在其他文章中很少出现，则认为这个词具有很好的类别区分能力，该词的重要性也越高。
- 词频

$$TF_w = \frac{\text{单词}w\text{在该文章中出现的次数}}{\text{该文章中所有单词的数目}}$$

- 逆文档频率:

$$IDF_w = \log\left(\frac{\text{所有文章的数目}}{\text{单词}w\text{在该所有文章中出现的次数}+1}\right)$$

$$\text{TF}-\text{IDF} = TF_w \times IDF_w$$

- 假设共有10,000,000篇文章，其中有一篇文章，该文章包含100个单词，其中“猫”在这篇文章中出现3次，有1,000篇文章包含“猫”;在同一篇文章中，“的”这个词出现了20次，并且该词所有文章中都出现过，求“猫”的TF-IDF值和“的”的TF-IDF值并比较

词频-逆文档频率

- 优点：
  - 在词袋表示的基础上进行了一定的改进，在保留文章的重要词的同时可以过滤掉一些常见的、无关紧要的词
- 缺点：
  - 忽略了句子中词的位置信息
    - ✓ 与词袋模型相同，也未考虑词的位置信息；
  - 精度问题
    - ✓ IDF是一种试图抑制噪声的加权，更倾向于文中频率较小的词，这使得TF-IDF算法的精度不够高。

词义假说

- 分布式表示方法的理论基础来自Harris在1954年提出的分布假说(Distributional Hypothesis)：上下文相似的词，其语义也相似。
- Firth在1957年对分布假说进行了进一步阐述和明确：词的语义由其上下文决定(A word is characterized by the company it keeps)。词的分布式表示可以将每个词都映射到一个较短的词向量上来，所有的这些词向量就构成了向量空间，并用普通的统计学的方法来研究词与词之间的关系。

全局词向量表示GloVe

Example corpus:

- I like deep learning.
- I like NLP.
- I enjoy flying.

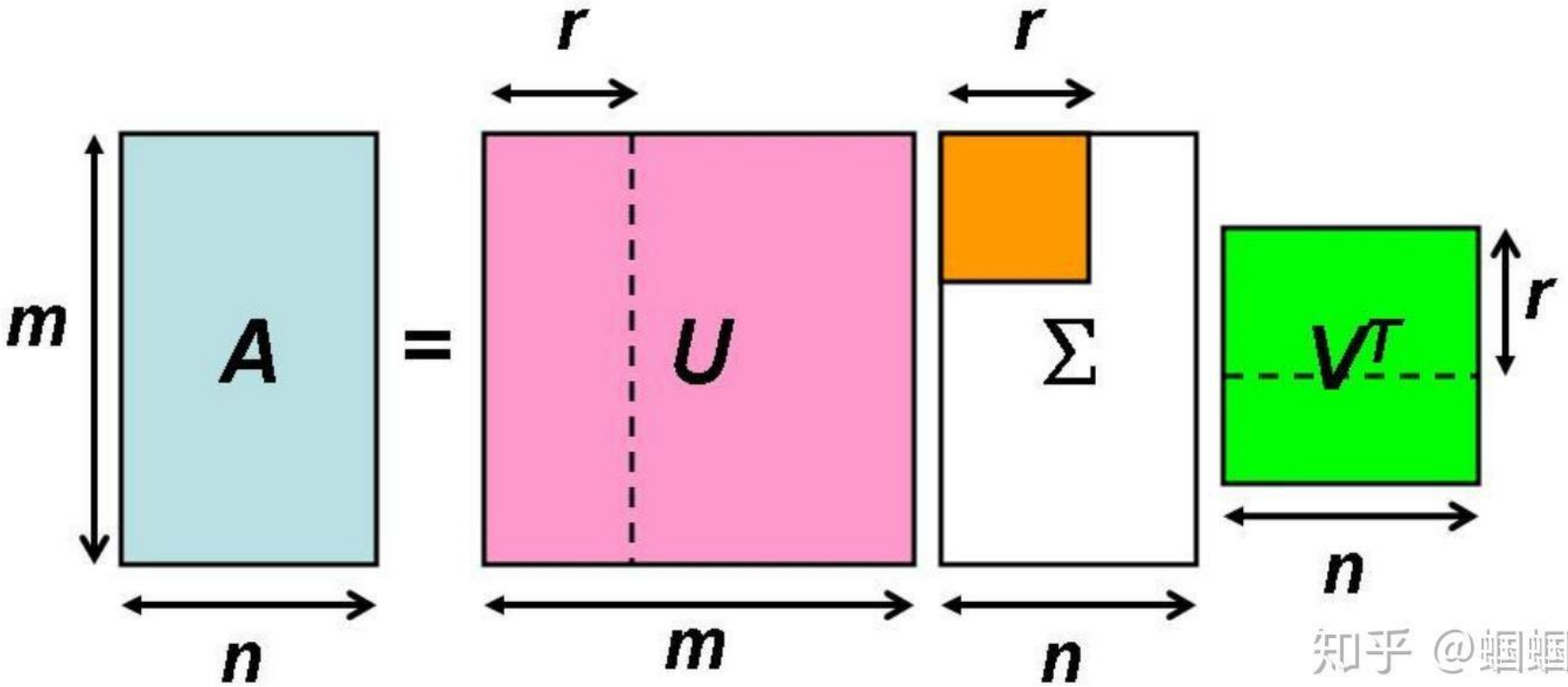
给出了三句话，假设这就是我们全部的语料。我们使用一个size=1的窗口，对每句话依次进行滑动，相当于只统计紧邻的词。这样就可以得到一个共现矩阵。

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

然后这样表示还有有一些问题：

- 维度=词汇量大小，还是太大了；
- 还是太过于稀疏，在做下游任务的时候依然不够方便。

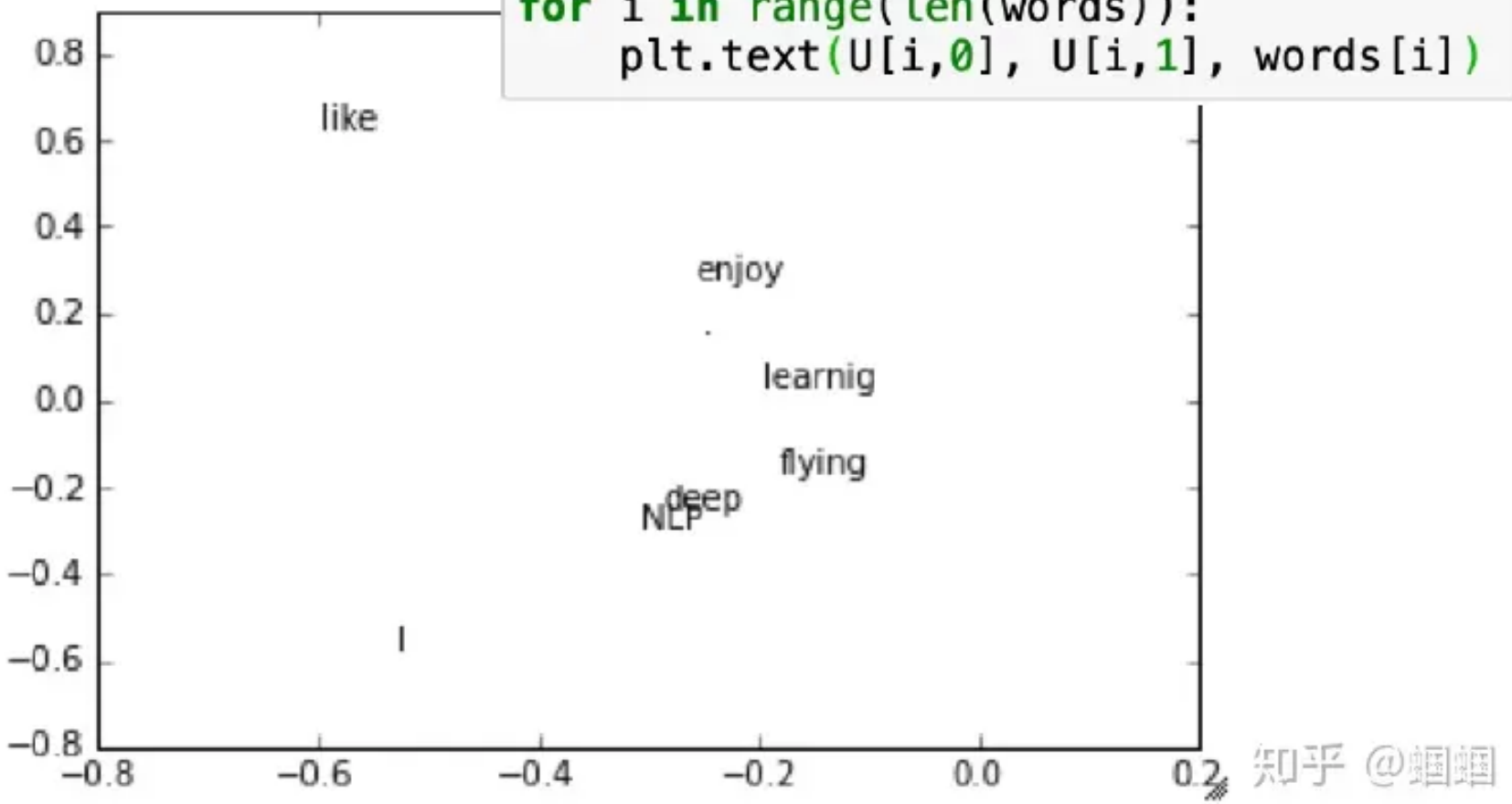
维度问题，我们有解决方法——「**SVD矩阵分解**」！ 我们将巨大的共现矩阵进行SVD分解后，只选取最重要的几个特征值，得到每一个词的低维表示。



图中的A在我们的场景中就是共现矩阵，U、∑、V就是分解出的三个矩阵。我们只选择U矩阵的前r维来作为词的向量表示。

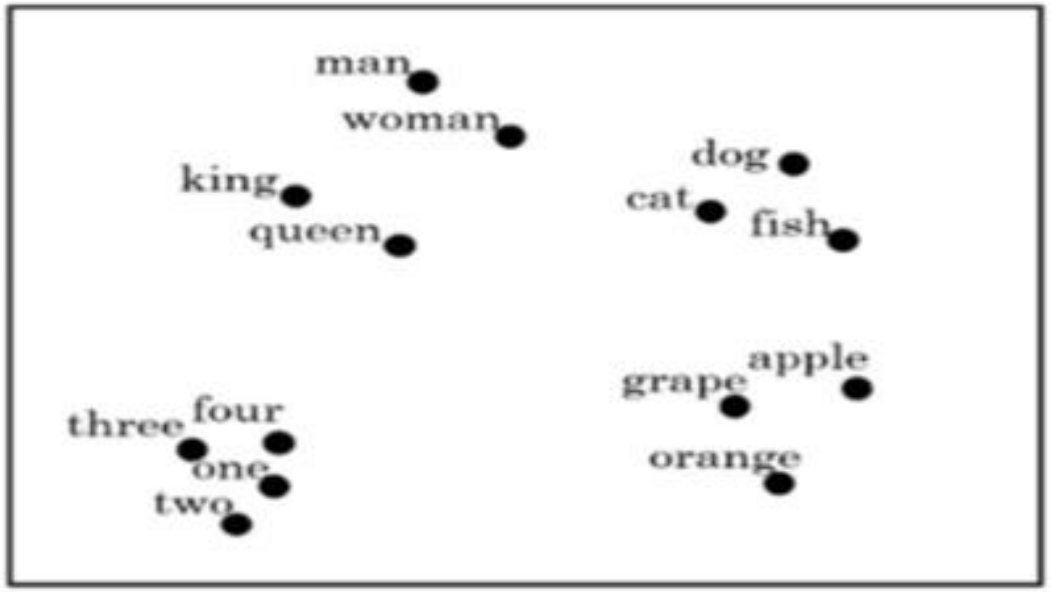
全局词向量表示GloVe


```
for i in range(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```

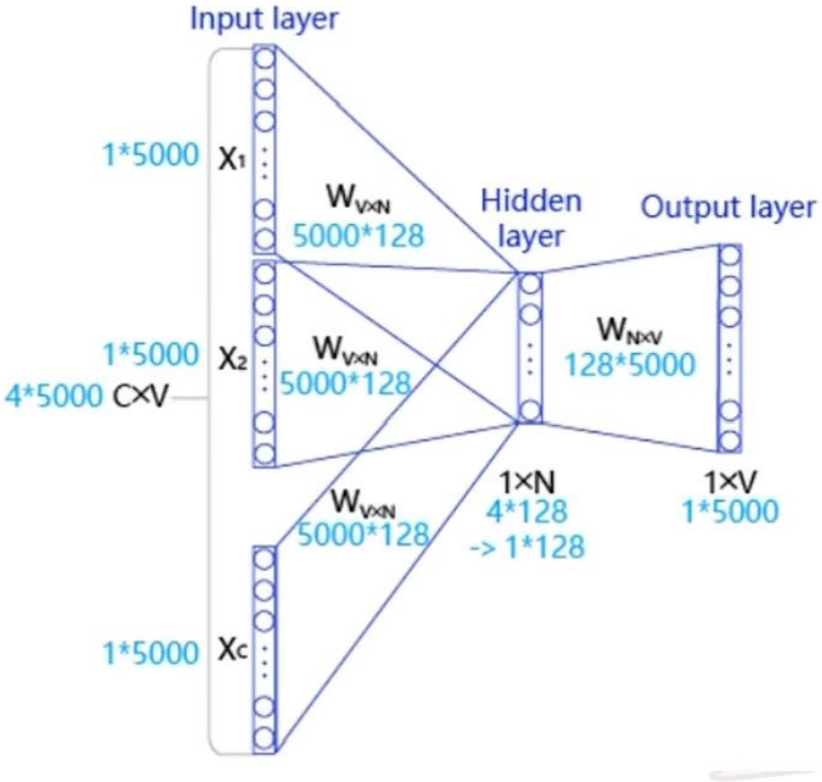


由于共现矩阵巨大，SVD分解的计算代价也是很大的。另外，像a、the、is这种词，与其他词共现的次数太多，也会很影响效果。

- Word2Vec是2013年谷歌提出的一种基于神经网络的分布式文本表示方法，该方法将单词映射成一个指定维度的稠密向量(典型取值为300维)，这个向量是对应单词的分布式表示(Distributional Representation)，并且能够在一定程度上表达单词的语义信息以及单词与单词之间的相似程度。



- Word2Vec的核心思想是通过词的上下文得到词的向量化表示，包括：
  - 连续词袋(Continuous Baq-of-Words,CBOW)：根据上下文词预测中间的词；
  - Skip-Gram模型：根据中间词预测上下文的词；



- **输入层**：一个形状为$C \times V$的one-hot张量，其中C代表上下文单词的个数，V代表词表大小
- **隐藏层**：一个形状为 $1 \times N$ 的张量，其中 N 表示每个词的词向量维度
- **输出层**：一个形状为 $1 \times V$ 的向量，使用上下文去推理中心词时，每个候选词的得分

Word2Vec模型

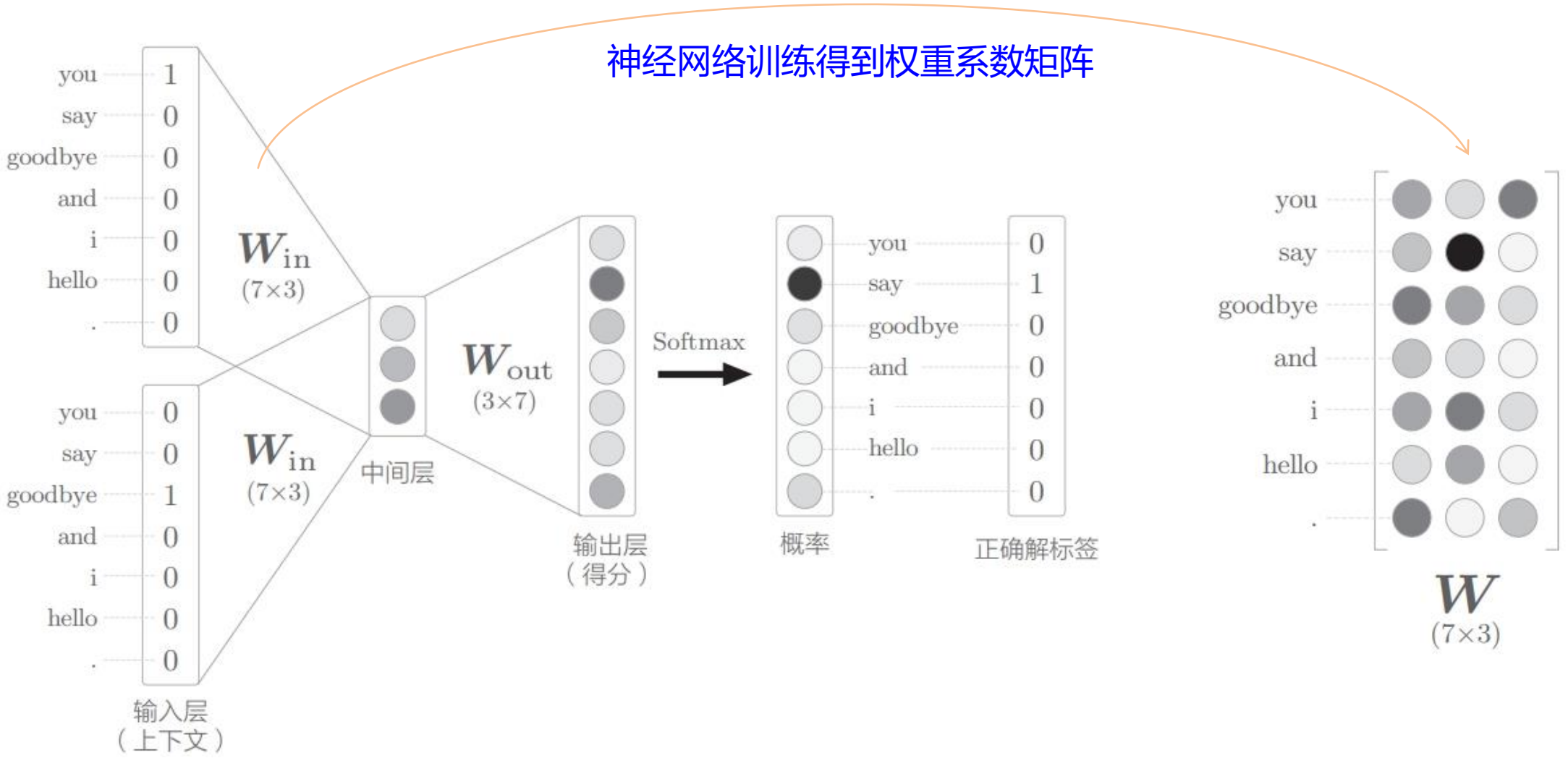
- 语料：you say goodbye and i say hello.
- 单词、单词ID以及它们的one-hot表示

单词	单词ID	one-hot表示
you goodbye	0 2	(1, 0, 0, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0, 0)

- 构建带标签的样本集

corpus	contexts	target
you say goodbye and i say hello .	you, goodbye	say
you say goodbye and i say hello .	say, and	goodbye
you say goodbye and i say hello .	goodbye, i	and
you say goodbye and i say hello .	and, say	i
you say goodbye and i say hello .	i, hello	say
you say goodbye and i say hello .	say, .	hello

三层神经网络


神经网络训练得到权重系数矩阵
you
say
goodbye
and
i
hello
.
1
0
0
0
0
0
0
W_{in}
(7×3)
W_{out}
(3×7)
中间层
Softmax
输入层
（上下文）
输出层
（得分）
概率
正确解标签
W
(7×3)

Word2Vec模型

- 在Word2Vec之后的第二年，GloVe (Global Vectors for Word Representation，全局词向量表示)被提出，该方法也属于分布式表示方法，但与Word2Vec不同的是该方法并未使用神经网络模型，而是基于全局词频统计构建词汇共现(共同出现)矩阵，并对共现矩阵降维。Word2Vec与GloVe方法是2018年之前最为常用的两种文本表示方法，其优缺点如下：
  - 优点：解决了数据稀疏、向量维度过高、字词之间的关系无法度量的问题
  - 缺点：Word2Vec与GloVe得到的词向量都是静态词向量(词向量在训练结束之后不会根据上下文进行改变)，静态词向量无法解决多义词的问题，例如:斤苹果”和“苹果7”中的“苹果”就是一个多义词，而这两种方法中“苹果”词的向量表示是相同的。
- 解决方案:ELMO、GPT、BERT、XLNet等模型中的动态词向量，区别于直接训练得到静态词向量的方式，动态词向量是在后续使用中把句子传入语言模型，并结合上下文语义得到更准确的词向量表示。动态词向量解决了多义词的问题，并凭借其在多项自然语言处理任务中取得的优异表现，迅速成为目前最流行的文本向量化表示方法。

目录
Contents

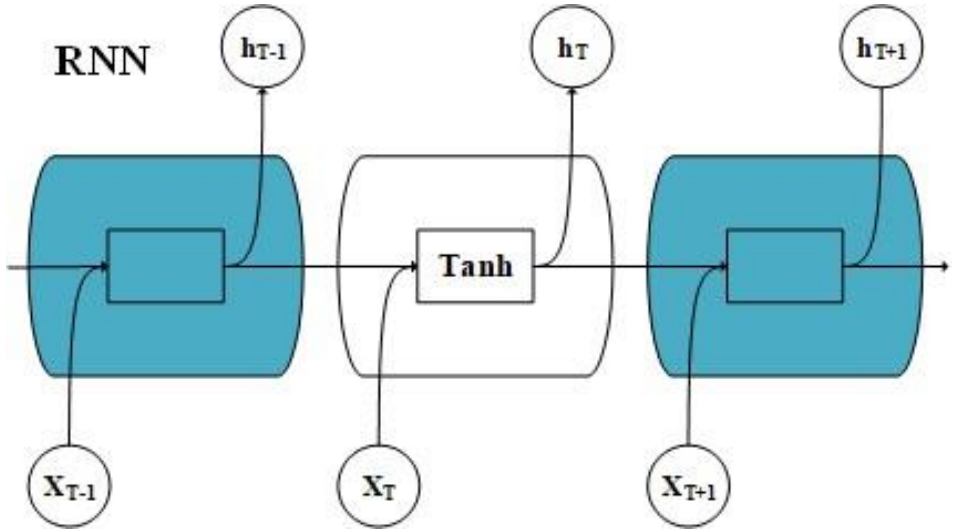
注意力机制

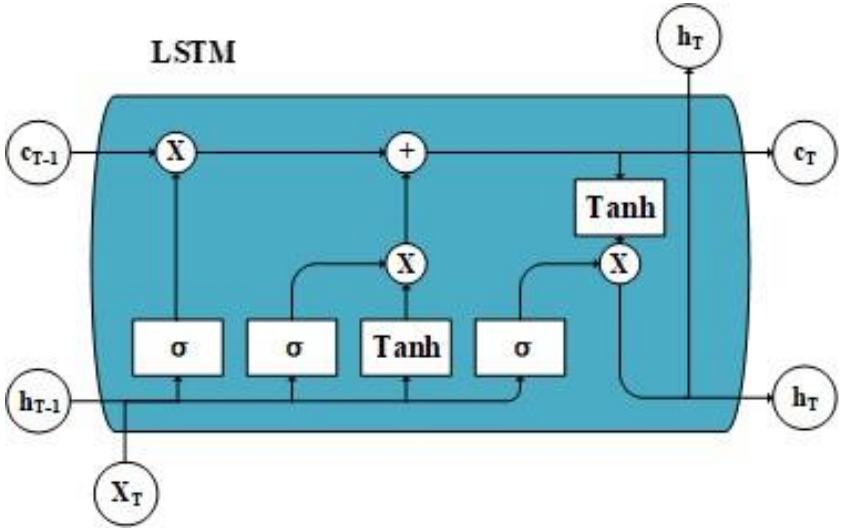
请集中你所有的**注意力**，数一数，穿白色运动服的人传了多少次球？

注意力之前的神经网络模型

- 尽管 RNN 和 LSTM 结构已经取得了良好的效果，但仍然存在许多缺点：

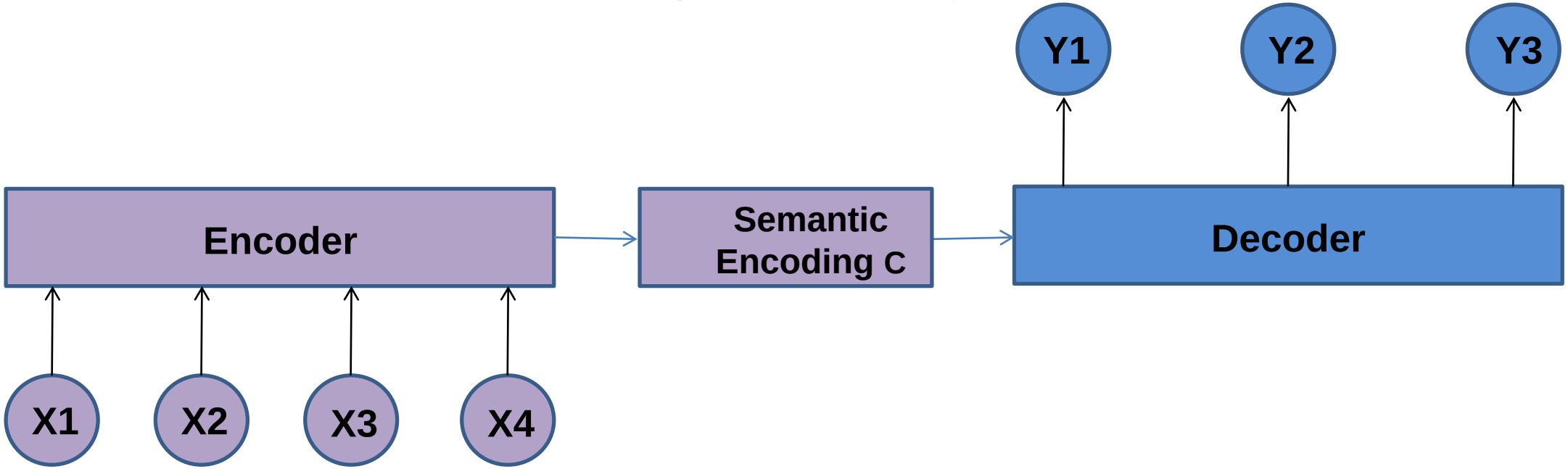
1. 在 RNN 和 LSTM 模型中，输入产生的输出会被计算为下一时刻的输入。因此，这种模型无法实现并行化。
2. RNN 和 LSTM 模型的长程依赖能力不足。





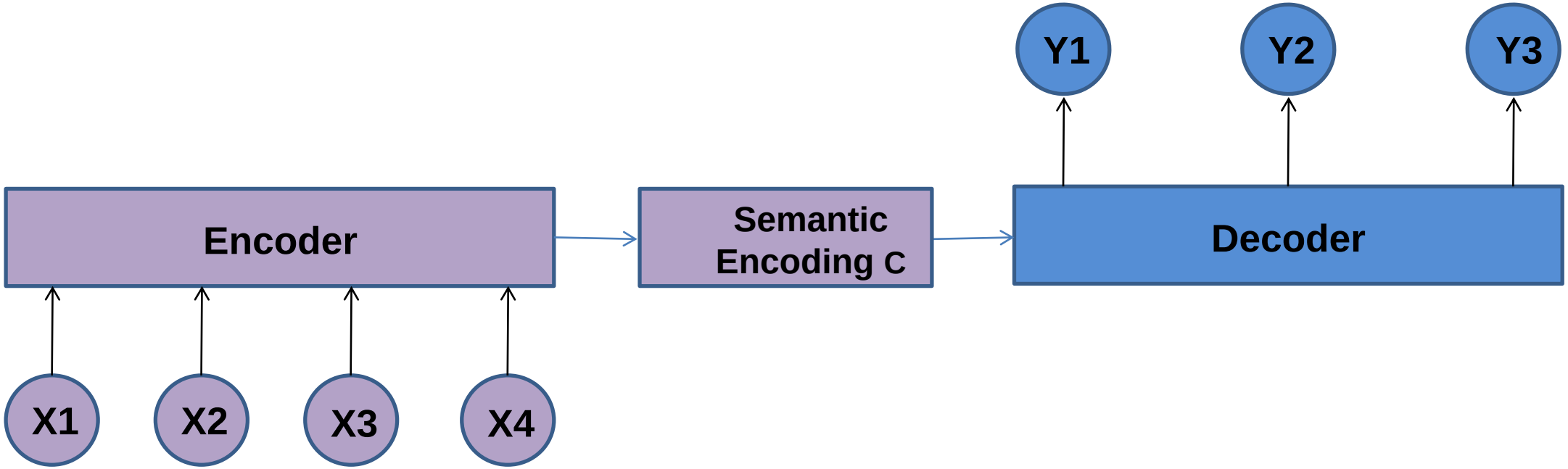
注意力之前的神经网络模型

- Seq2Seq 模型从一个句子生成另一个句子（或段落）的通用处理模型。其编码器和解码器部分可以采用 RNN、LSTM 或 CNN 结构。
- 编码器会将输入句子 X 编码成一个固定长度的隐藏向量（语义编码 c），这可能导致隐藏向量无法完全表示输入句子 X 的全部信息。
- 解码器会解码语义编码 c 的内容，并将其输出为一个序列。



注意力之前的神经网络模型

- 隐藏向量的大小是有限的，无法表示语义丰富的句子。这会导致输入内容所携带的信息被稀释或被后续输入的信息覆盖，输入序列越长，这种问题越严重。
- 输入句子中的每个单词被赋予相同的权重，缺乏区分性。例如，在机器翻译中，通常是一个或几个单词的输入对应一个或多个单词的输出，这种处理方式会降低模型的性能。



注意力的引入

- 为了解决 Seq2Seq 的问题：在 2015 年的 ICLR 会议上，文章《NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE》提出了一些关于注意力机制的想法。
- 在 2016 年的 ICLR 会议上，文章《FEED-FORWARD NETWORKS WITH ATTENTION CAN SOLVE SOME LONG-TERM MEMORY PROBLEMS》正式提出了注意力机制的概念。

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Workshop track - ICLR 2016

FEED-FORWARD NETWORKS WITH ATTENTION CAN SOLVE SOME LONG-TERM MEMORY PROBLEMS

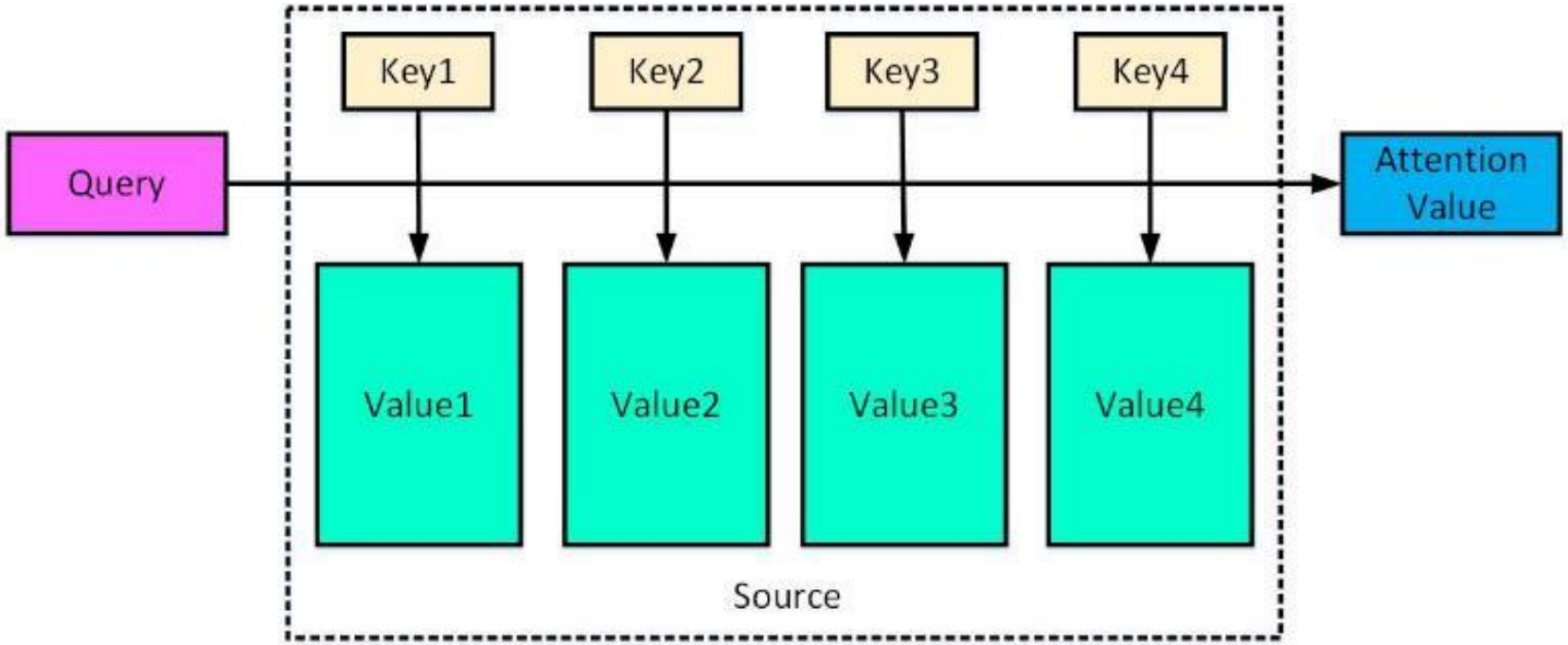
Colin Raffel
LabROSA, Columbia University
craffel@gmail.com

Daniel P. W. Ellis
LabROSA, Columbia University
dpwe@ee.columbia.edu

注意力的计算

$$\text{Attention}(\text{Query}, \text{Source}) = Softmax\left[\sum_{i=1}^{I_x} similiarity(Query, Key)\right] \times Value$$

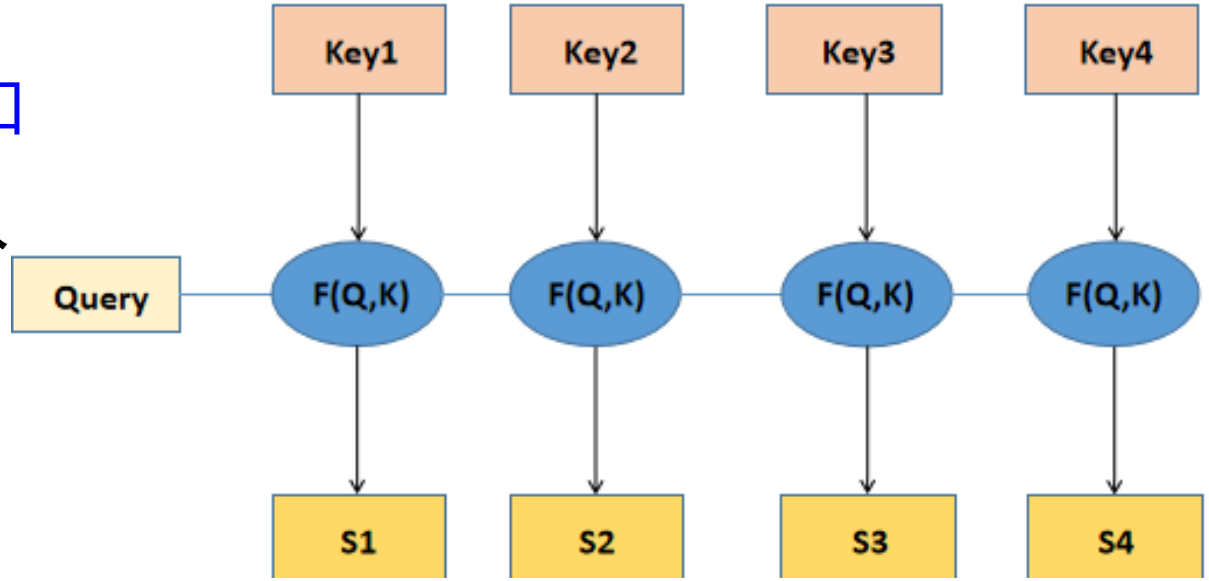
源数据中的组成元素被视为一系列<Key, Value>（键值对）结构



注意力模型的作用是学习序列中每个元素的重要性，然后根据重要性对这些元素进行组合。

注意力的计算

- 在第一阶段，根据查询（Query）和键（Key）计算之间的相似性，引入不同的函数和计算机系统。



- 常见的方法包括：

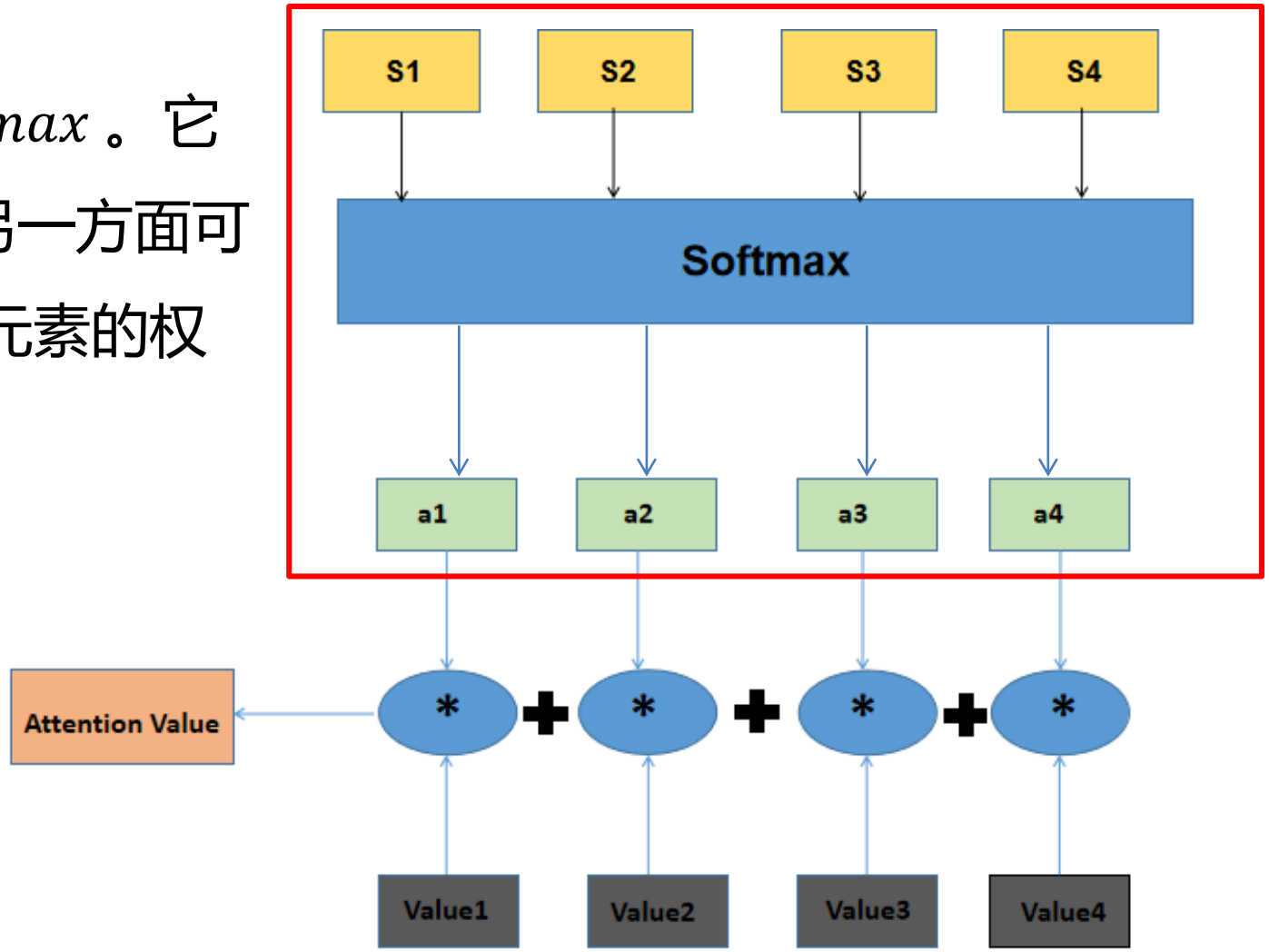
1. 计算两者的向量点积：$similarity(Query, Key_i) = Query \cdot Key_i$

2. 计算两者的向量余弦相似度：$similarity(Query, Key_i) = \frac{Query \cdot Key_i}{||Query|| \cdot ||Key_i||}$

3. 引入额外的神经网络进行评估：$similarity(Query, Key_i) = MLP(Query, Key_i)$

注意力的计算

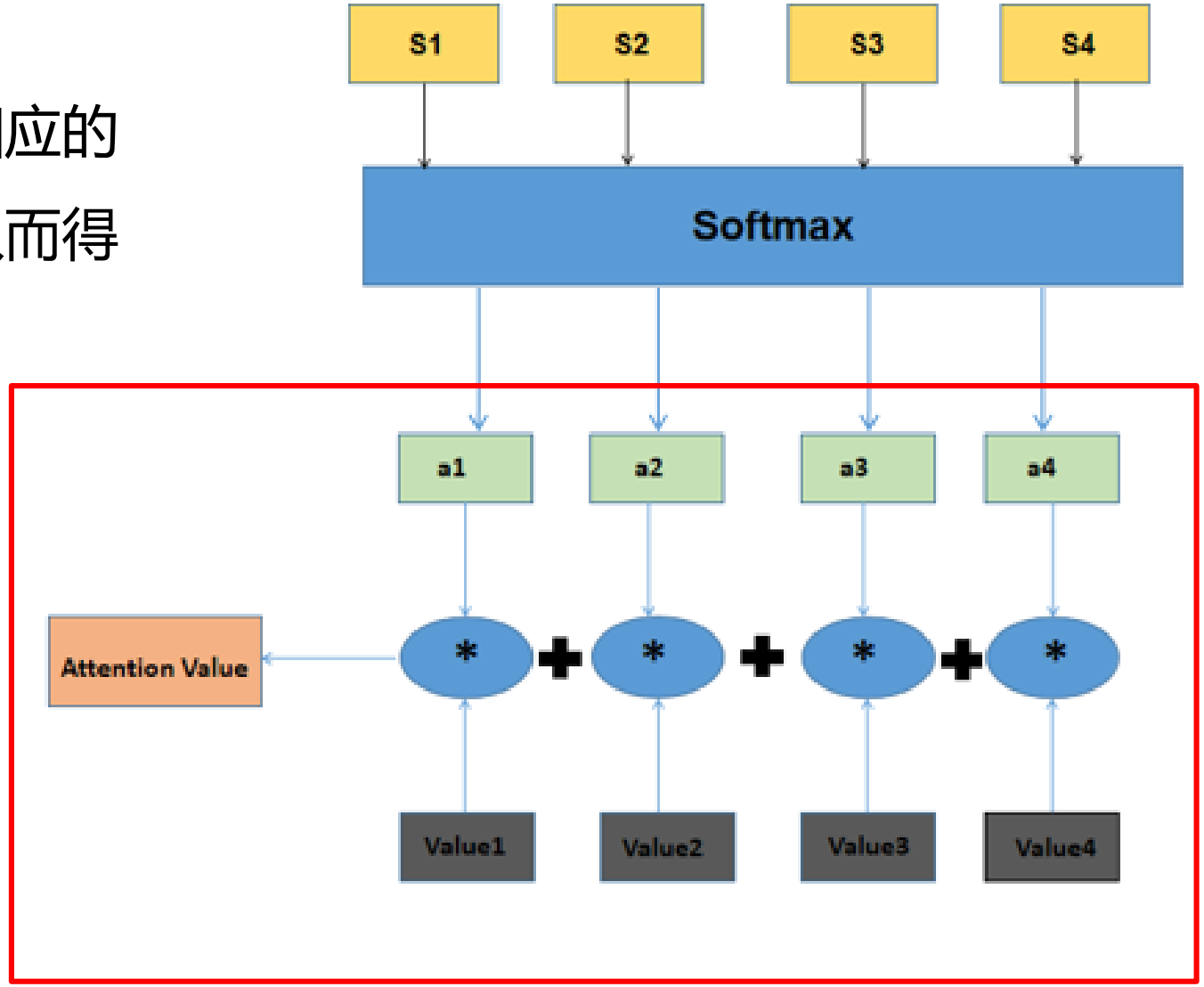
- 在第二阶段，引入了 $Softmax$ 。它一方面可以进行归一化，另一方面可以通过 $Softmax$突出重要元素的权重。



注意力的计算

- 第三步将前一步的计算结果作为相应的权重系数，然后进行加权求和，从而得到注意力值（Attention value）。

$$\text{Attention}(\text{Query}, \text{Source}) = \sum_{i=1}^{I_x} a_i \cdot Value_i$$

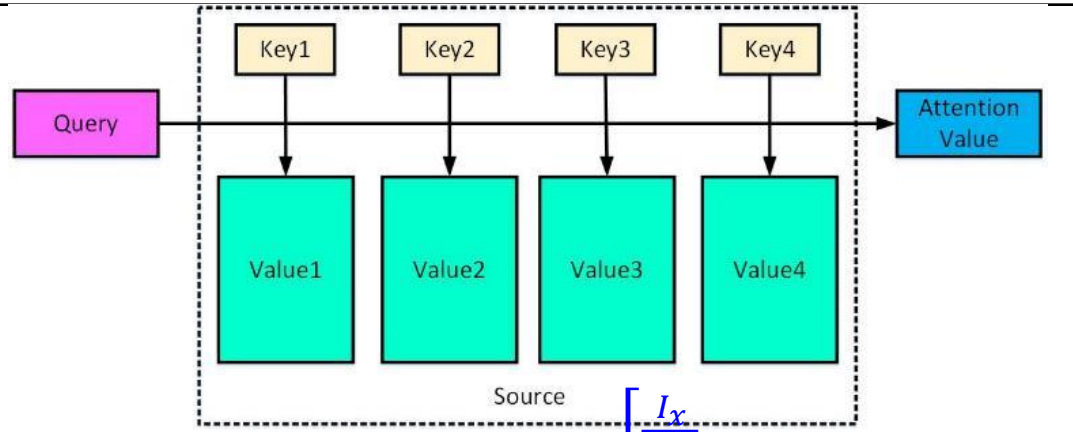
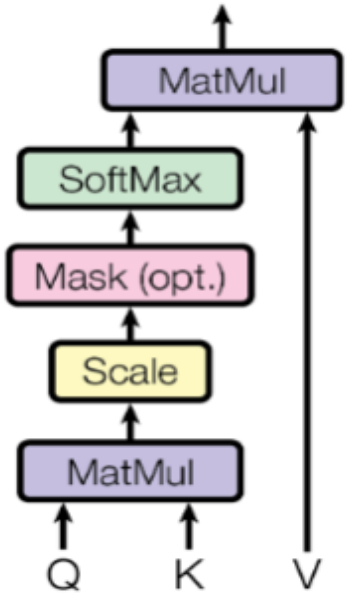


通用注意力和自注意力

- 自注意力机制（Self-Attention）是发生在源数据（Source）内部或目标数据（Target）内部元素之间的注意力机制。它也可以理解为 Target = Source 注意力计算系统的特例。

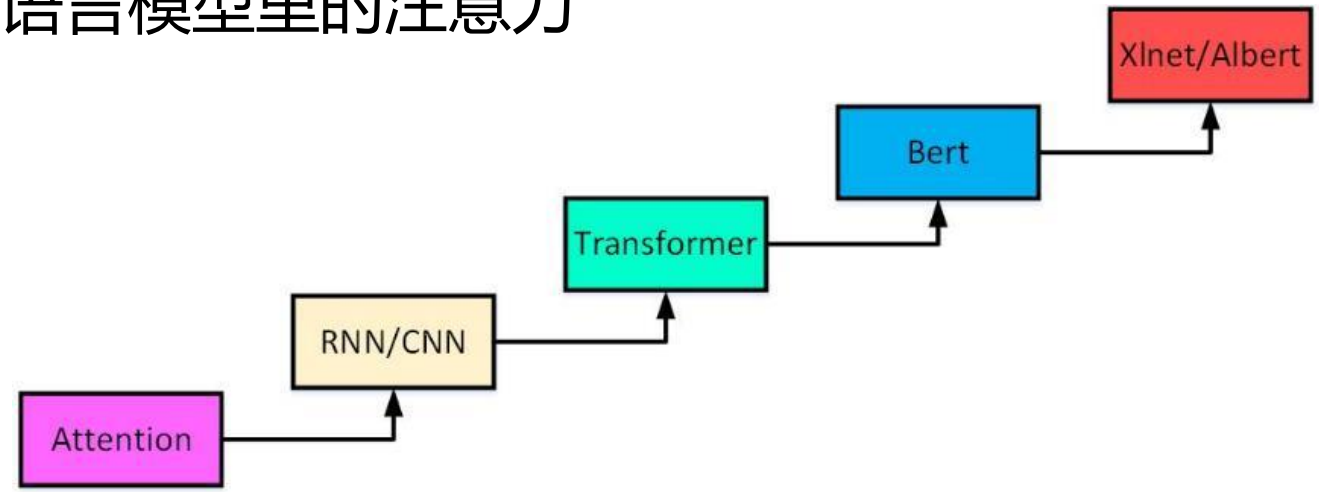
模型	Transformer	LSTM	CNN
基本机制	自注意力	循环神经网络	卷积神经网络
并行性	是	否	是
长程依赖	是	是，但不够	否

模型的对比

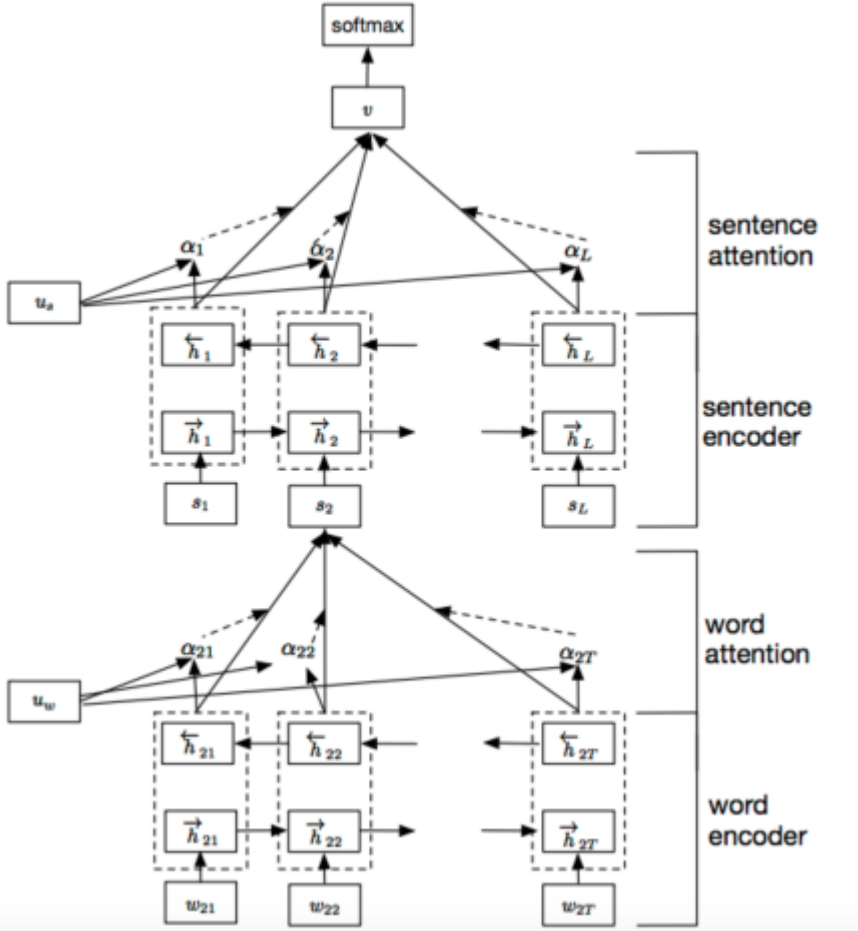
General Attention	 $\text{Attention}(\text{Query}, \text{Source}) = Softmax\left[\sum_{i=1}^{I_x} similiarity(Query, Key)\right] \times Value$
Self-Attention	 $\text{Attention}(\text{Q}, \text{K}, \text{V}) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

注意力的应用

语言模型里的注意力



文本分类



机器翻译

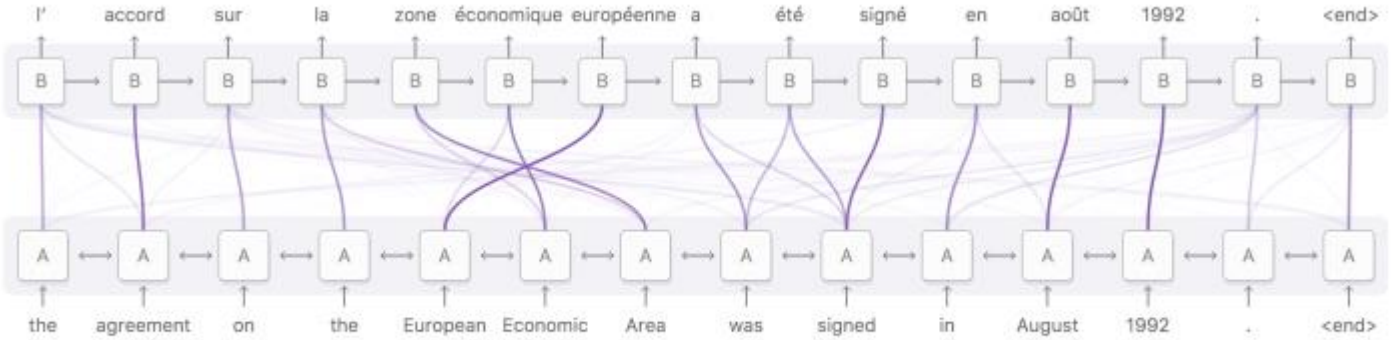


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

目录
Contents

Transformer模型结构

- Transformer 是 Google 研究人员在 2017 年 NIPS 会议上发表的文章《Attention Is All You Need》中提出的一个用于 Seq2Seq 任务的模型。
- 它不具有 RNN 的循环结构或 CNN 的卷积结构。
- 在机器翻译等任务中取得了显著的改进。

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com
Noam Shazeer[*] Google Brain noam@google.com
Niki Parmar[*] Google Research nikip@google.com
Jakob Uszkoreit[*] Google Research usz@google.com
Llion Jones[*] Google Research llion@google.com
Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com
Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

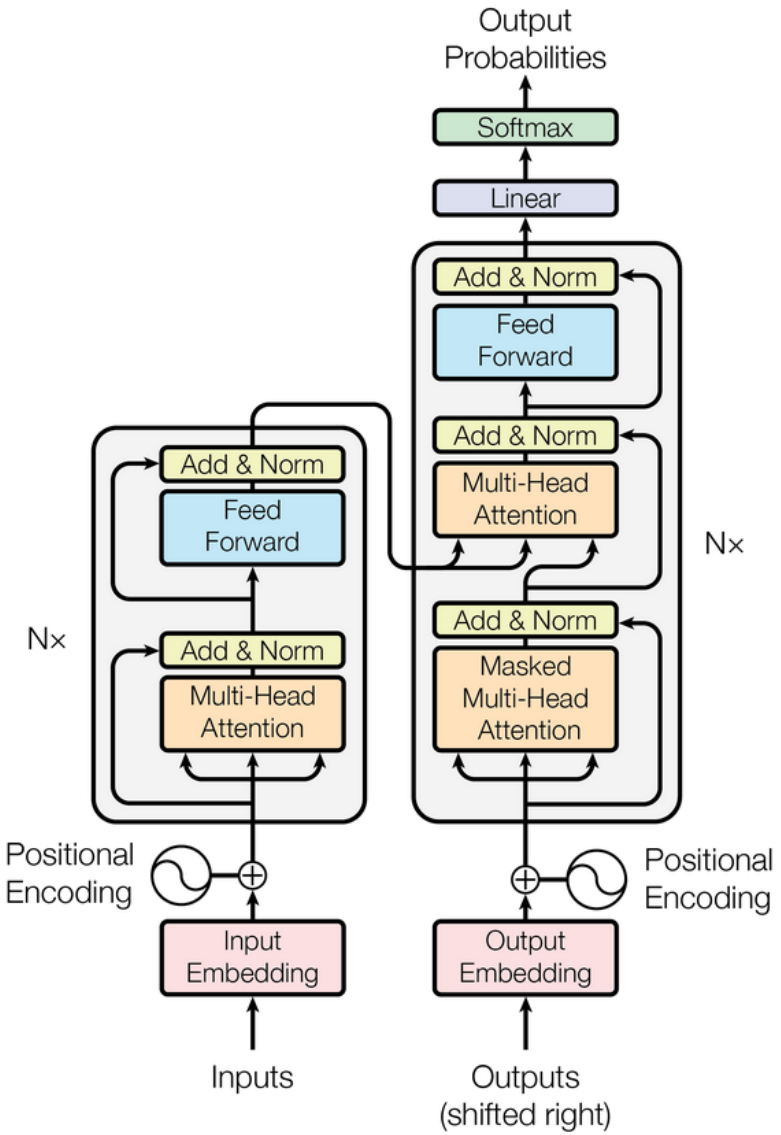
[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

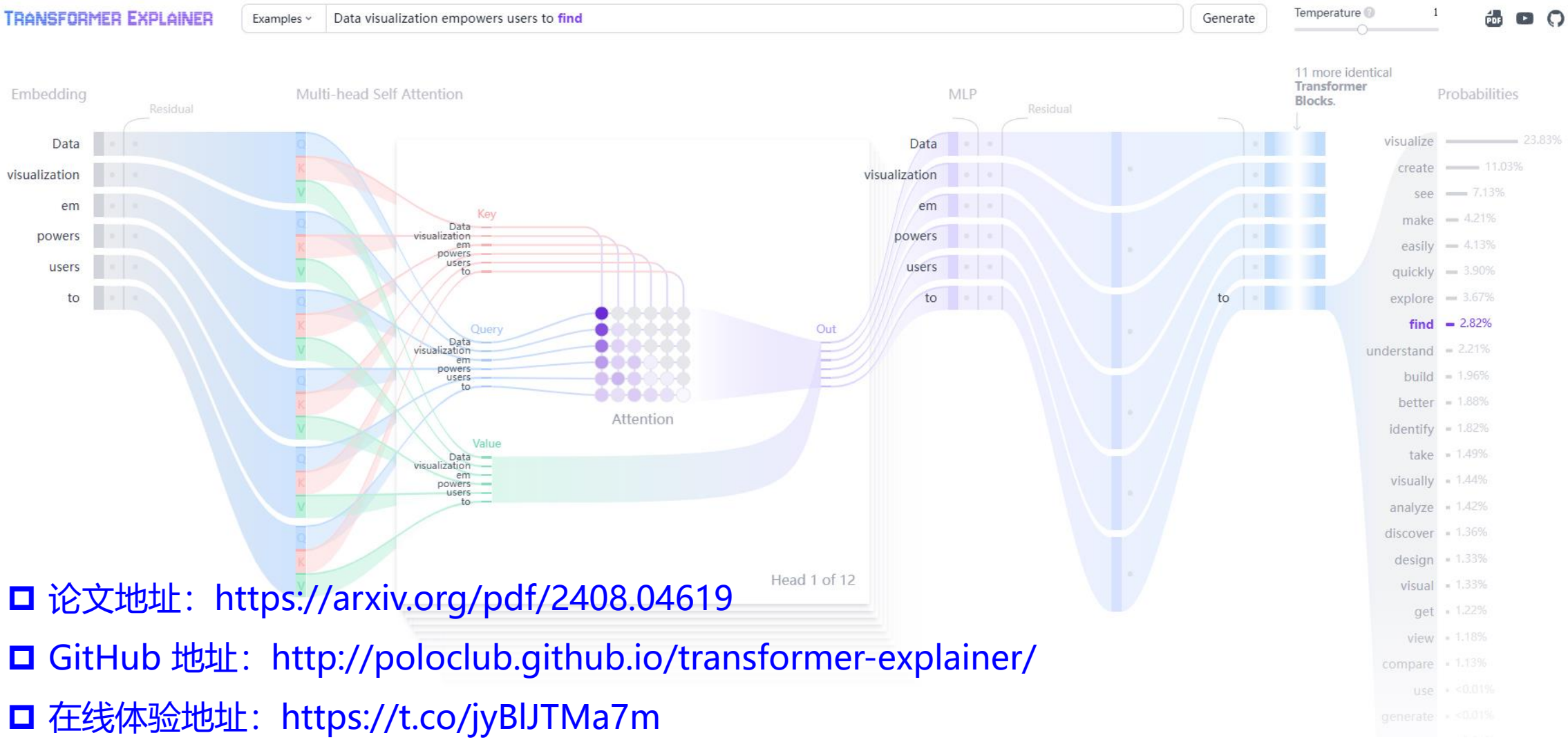
Transformer模型结构

- Transformer 包含四个部分：

1. 输入（Input）
2. 编码器（Encoder）
3. 解码器（Decoder）
4. 输出（Output）



Transformer可视化

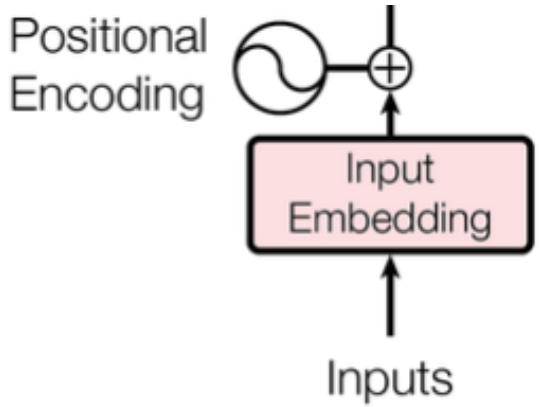


- 论文地址：https://arxiv.org/pdf/2408.04619
- GitHub 地址：http://poloclub.github.io/transformer-explainer/
- 在线体验地址：https://t.co/jyBlJTMa7m

输入，词嵌入

- 输入有两部分：

1. 输入嵌入（Input Embedding）：将单词转换为向量。
2. 位置编码（Positional Encoding）：编码单词在不同位置出现的事实。



- 给定一个语料库中的一句话：“Hello, how are you?”
- 第一步是进行分词：

“Hello, how are you?” → [“Hello”，“,”，“how”，“are”，“you”，“?”]

然后，我们将得到词向量矩阵。

$$\begin{array}{c} \\ Hello \\ , \\ how \\ are \\ you \\ ? \end{array} \begin{array}{c} \begin{array}{ccccc} < & - & d_{emb\_dim} & - & > \end{array} \\ \begin{pmatrix} 123.4 & 0.32 & \cdots & 94 & 32 \\ 83 & 34 & \cdots & 77 & 19 \\ 0.2 & 50 & \cdots & 33 & 30 \\ 289 & 432.98 & \cdots & 150 & 92 \\ 80 & 46 & \cdots & 23 & 32 \\ 41 & 21 & \cdots & 74 & 33 \end{pmatrix} \end{array}$$

位置编码

- Transformer 模型不使用 RNN 结构，因此无法捕捉序列的顺序信息，因此需要其他机制将位置信息传递到编码部分。

1.按文本长度计数

0	1	2	3	4	5	.....	497	498	499

2.文本长度在每个位置上进行归一化处理

短文本	0/99	1/99	2/99	3/99	4/99	.....	96/99	97/99	98/99	99/99
长文本	0/999	1/999	2/999	3/999	4/999	.....	996/999	997/999	998/499	999/999

位置编码

- 位置编码的要求：

1. 必须防止编码与文本长度无关。
2. 需要反映一定的顺序关系。
3. 需要反映相同单词在不同位置上的差异。

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

因此，引入了正弦和余弦函数。

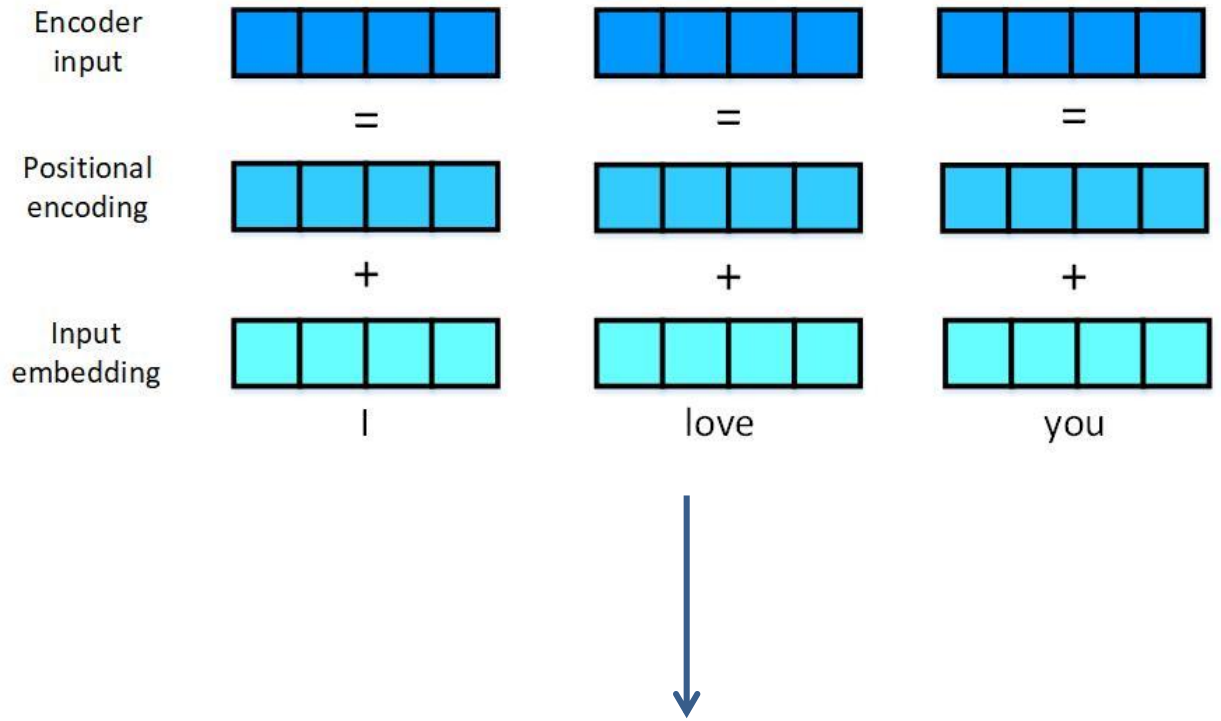
- 扩展上述公式后，得到以下形式：

$$\begin{cases} PE(pos+k,2i) = PE(pos,2i) * PE(k,2i+1) + PE(pos,2i+1) * PE(k,2i) \\ PE(pos+k,2i+1) = PE(Pos,2i) * PE(k,2i+1) - PE(pos,2i+1) * PE(k,2i) \end{cases}$$

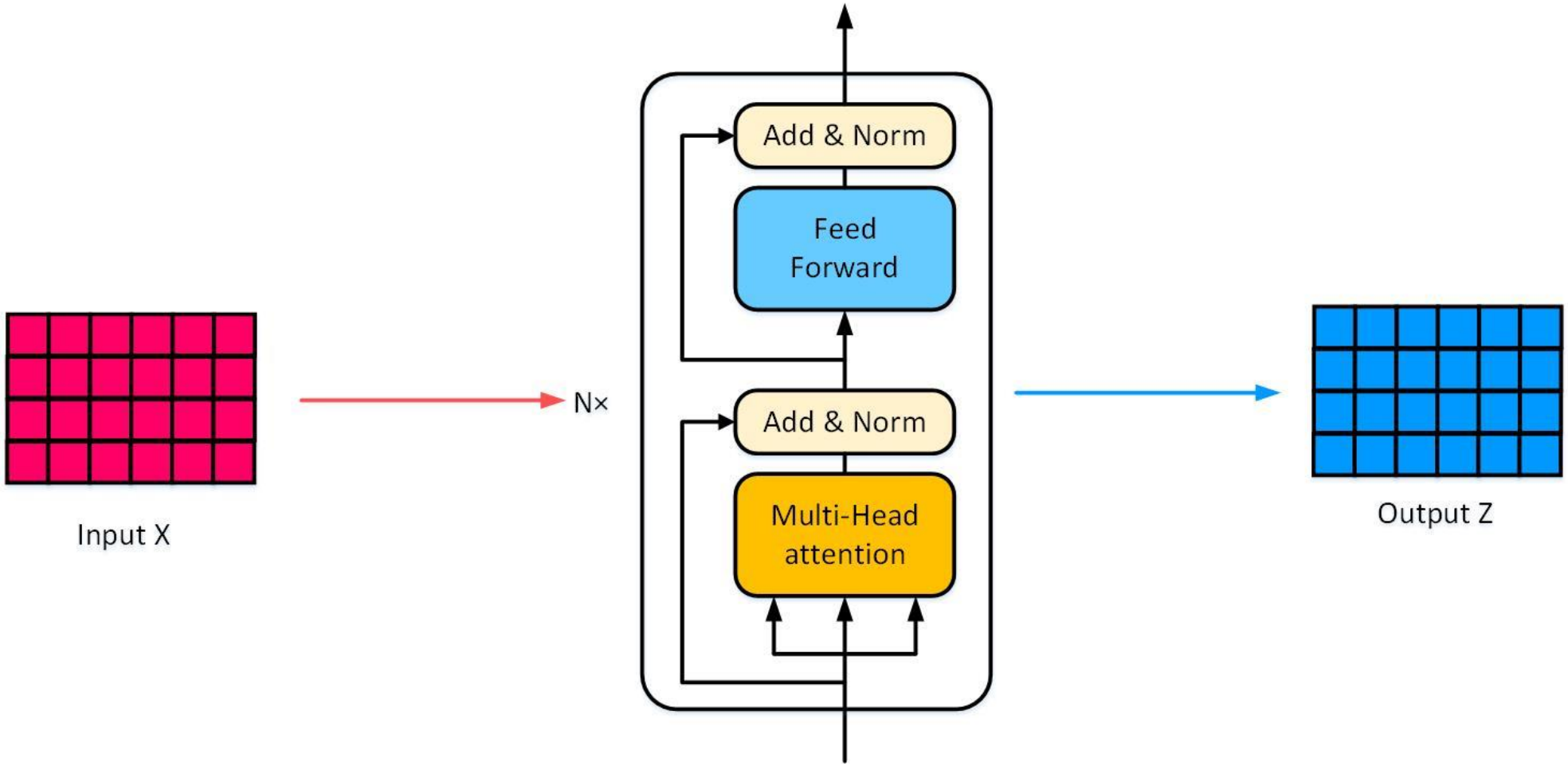
- 可以看出，对于位置向量中某一维度 $2i$ 或 $2i+1$ 的位置 $pos+k$，它可以表示为 $PE(pos)$ 和 $PE(k)$ 的组合，同时也可以表达相对位置的嵌入。

$$
\begin{array}{c}
 & < & - & d_{emb\_dim} & - & > \\
Hello & \sin\left(\frac{0}{10000^{\frac{0}{emb_{dim}}}}\right) & \cos\left(\frac{0}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{0}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{0}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots \\
, & \sin\left(\frac{1}{10000^{\frac{0}{emb_{dim}}}}\right) & \cos\left(\frac{1}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{1}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{1}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots \\
how & \sin\left(\frac{2}{10000^{\frac{0}{emb_{dim}}}}\right) & \cos\left(\frac{2}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{2}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{2}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots \\
are & \sin\left(\frac{3}{10000^{\frac{0}{emb_{dim}}}}\right) & \cos\left(\frac{3}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{3}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{3}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots \\
you & \sin\left(\frac{4}{10000^{\frac{0}{emb_{dim}}}}\right) & \cos\left(\frac{4}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{4}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{4}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots \\
? & \sin\left(\frac{5}{10000^{\frac{0}{d_{emb_{dim}}}}}\right) & \cos\left(\frac{5}{10000^{\frac{0}{emb_{dim}}}}\right) & \sin\left(\frac{5}{10000^{\frac{2}{emb_{dim}}}}\right) & \cos\left(\frac{5}{10000^{\frac{2}{emb_{dim}}}}\right) & \cdots
\end{array}
$$

Transformer的输入

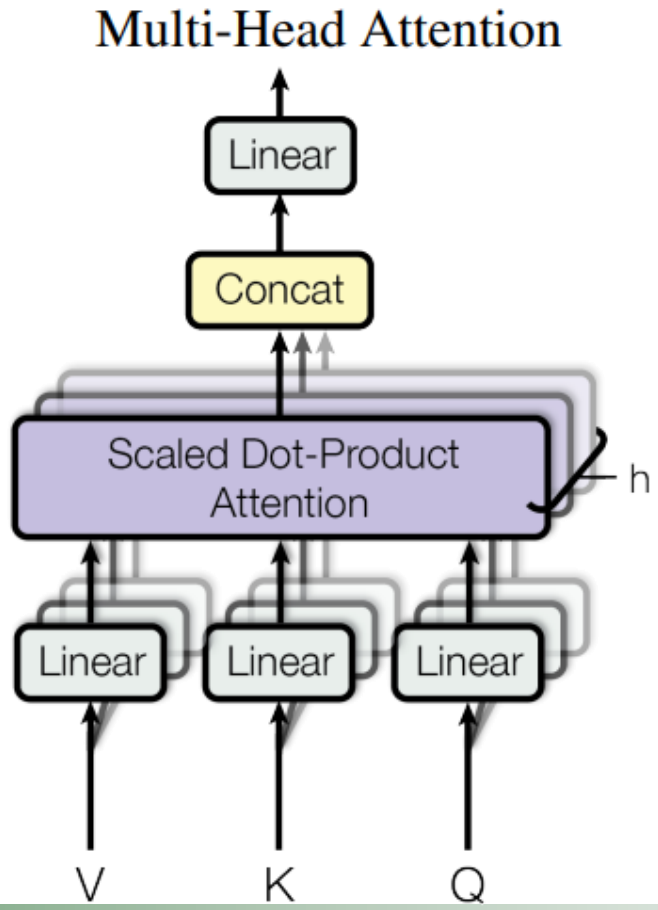


输入矩阵 X经过编码器处理后，会转化为一个隐藏状态矩阵。



多头注意力(multi-head attention)

- 在前一步中，我们知道如何通过自注意力（Self-Attention）计算得到输出矩阵 Z)，而多头注意力（Multi-Head Attention）是通过多个自注意力的组合形成的。



Scaled Dot-Product Attention

- 计算注意力包含五个步骤：

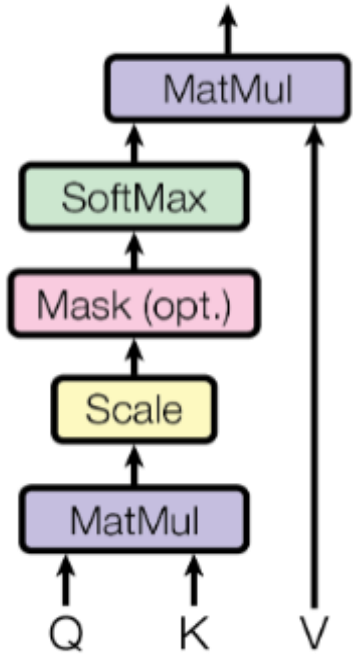
1.将两个矩阵 Q 和 K 相乘，得到 QK^T 矩阵；

2.将 QK^T 除以缩放因子；

3.对缩放后的矩阵进行掩蔽操作；

4.对矩阵的每一行执行 $softmax$ 操作，得到 SoftMax 矩阵；

5.最后，将 $softmax$ 矩阵与 V 矩阵相乘，得到注意力矩阵。



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

自注意力

- 在得到由每个单词的词嵌入（Word Embedding）和位置编码（Positional Encoding）相加形成的$X_{Embedding}$后，我们需要使用这个矩阵来获得三个矩阵：

1.Q（Query）矩阵

2.K（Key）矩阵

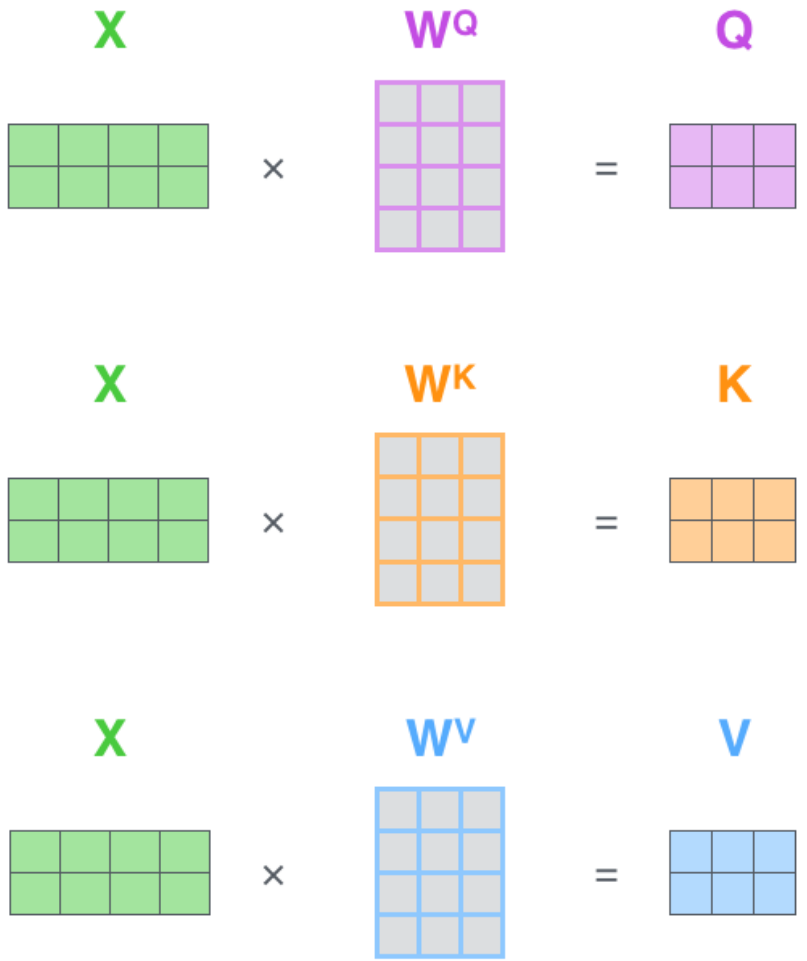
3.V（Value）矩阵

这些矩阵将用于后续的自注意力（Self-Attention）计算。

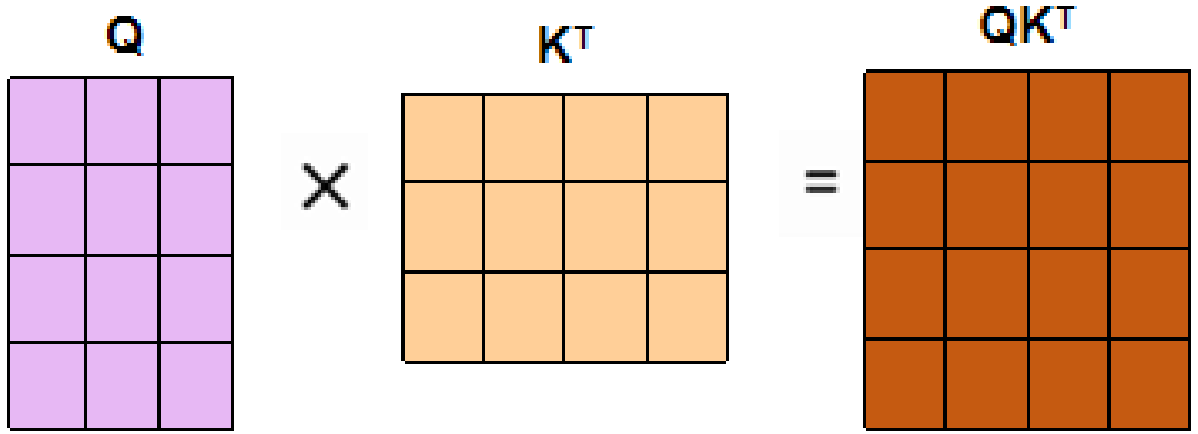
$$X_{Embedding} * W_Q = Q$$

$$X_{Embedding} * W_k = K$$

$$X_{Embedding} * W_V = V$$



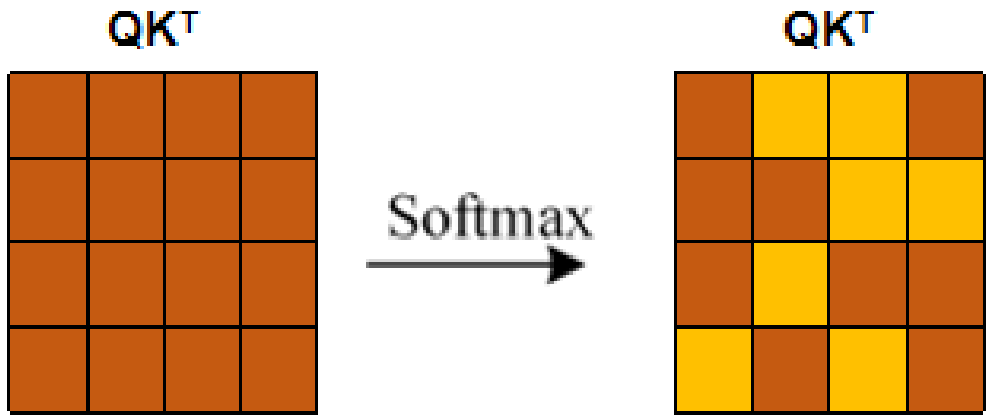
- QK^T矩阵表示句子中每个单词与其他单词之间的关联信息。它通过计算查询（Query）与键（Key）之间的相似性，揭示了每个单词在句子中与其他单词的关系。



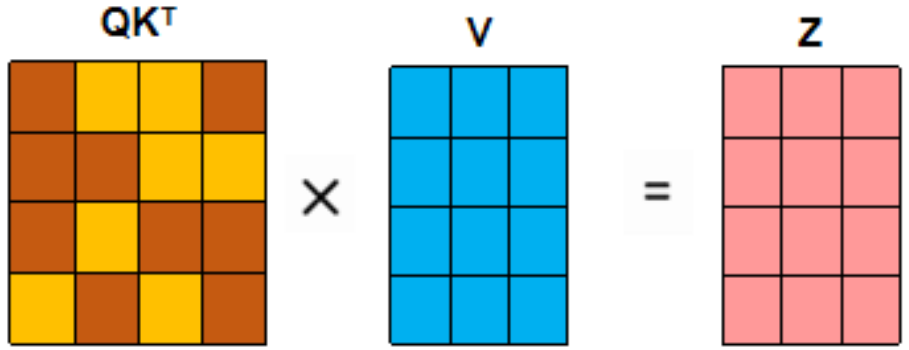
- 将 QK^T除以缩放因子在文章中解释为：我们怀疑对于较大d_k值，点积的数值会变得非常大，这会将 $softmax$ 函数推向梯度极小的区域。为了避免这种情况，缩放因子 $\sqrt{d_k}$ 被引入，以防止点积过大，确保将 $softmax$ 函数的梯度保持在合适的范围内。

自注意力

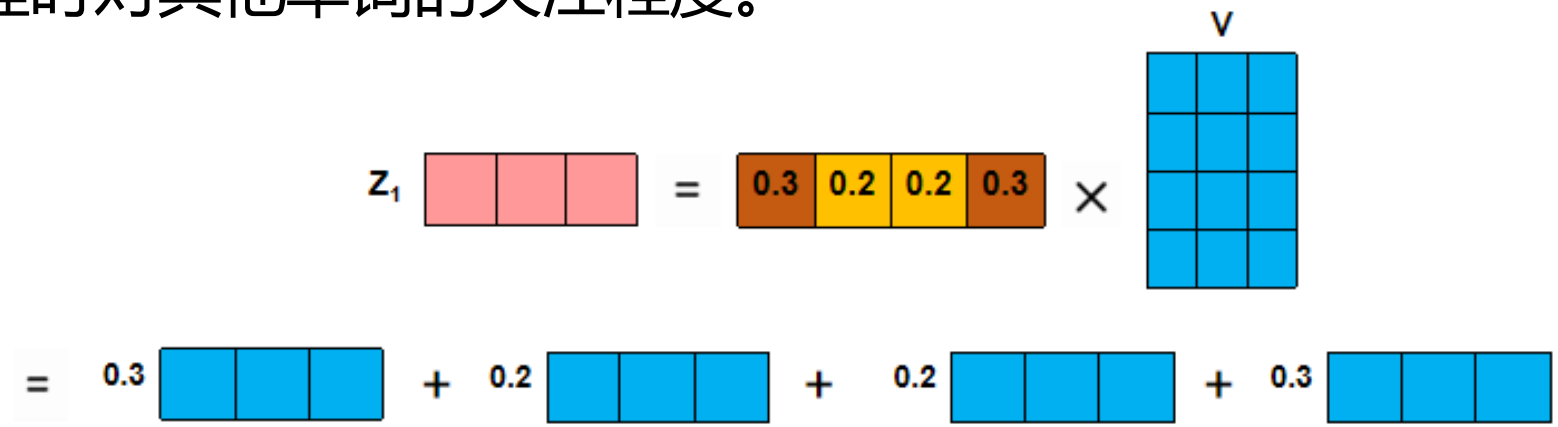
- 这完成了缩放操作，接下来进行掩蔽操作。
- 这里的掩蔽操作是 Padding Mask，用于遮盖掉填充的部分，避免它们在计算注意力时产生影响。
- 然后，使用 $softmax$来计算每个单词对其他单词的注意力系数。



在获得 $softmax$ 矩阵后，可以将其与 V 矩阵相乘，从而得到最终的输出矩阵 Z。



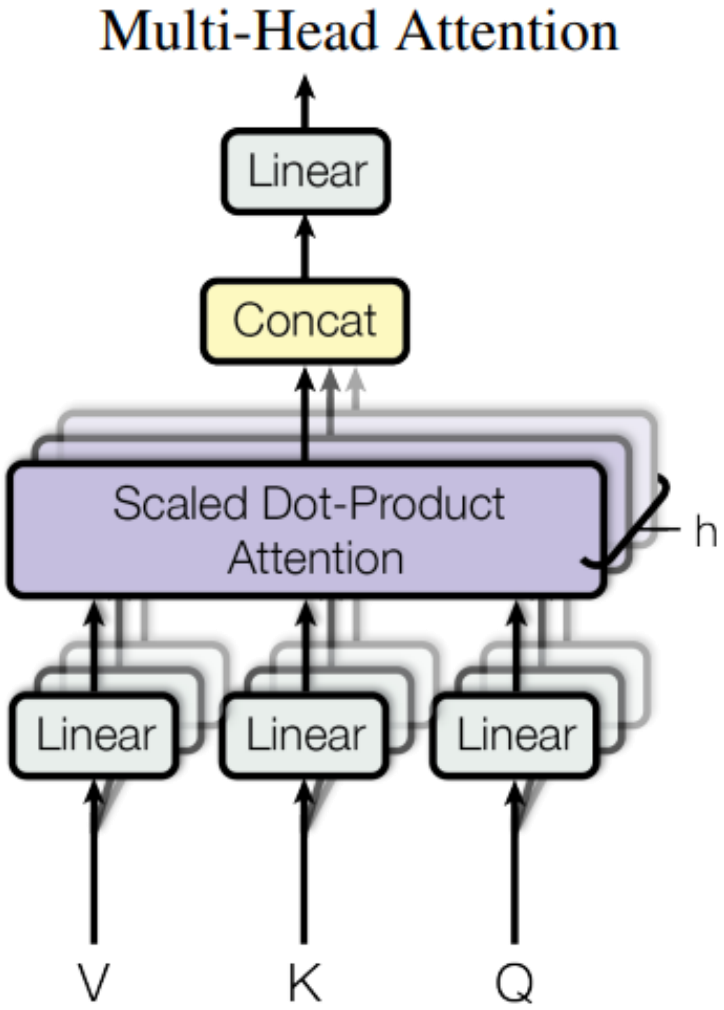
矩阵 Z 中的某一行表示某个单词与所有其他单词之间的注意力系数关系，即该单词在处理时对其他单词的关注程度。



多头注意力

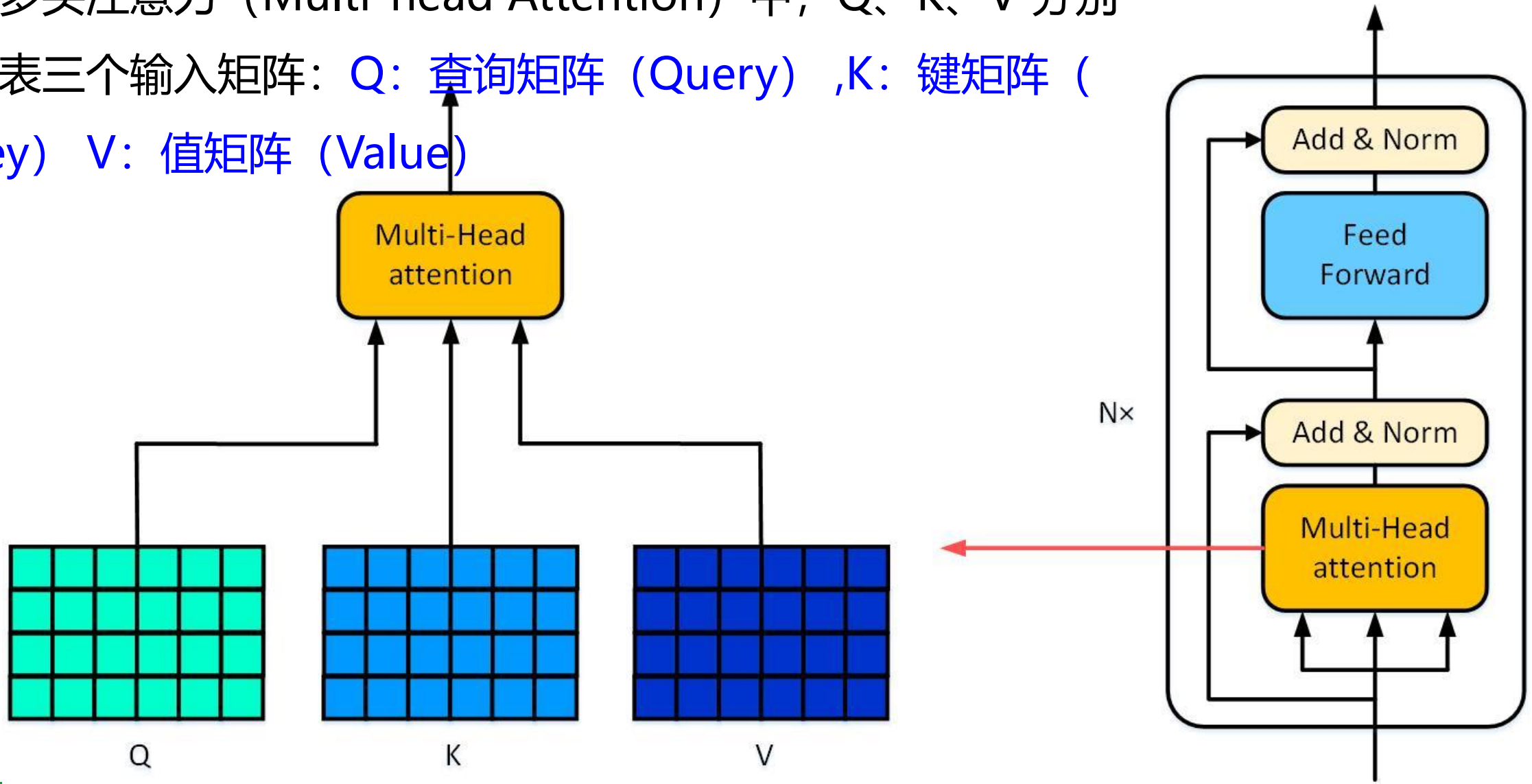
$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^O$$
$$where\ head_i = Attention\left(QW_i^Q, kW_i^K, VW_i^V\right)$$

多头注意力（Multi-head Attention）允许模型在不同的位置同时关注来自不同表示子空间的信息。通过将注意力机制应用于多个不同的子空间，模型能够捕捉到更多的特征和复杂的依赖关系。



多头注意力

在多头注意力（Multi-head Attention）中，Q、K、V 分别代表三个输入矩阵：Q：查询矩阵（Query），K：键矩阵（Key） V：值矩阵（Value）

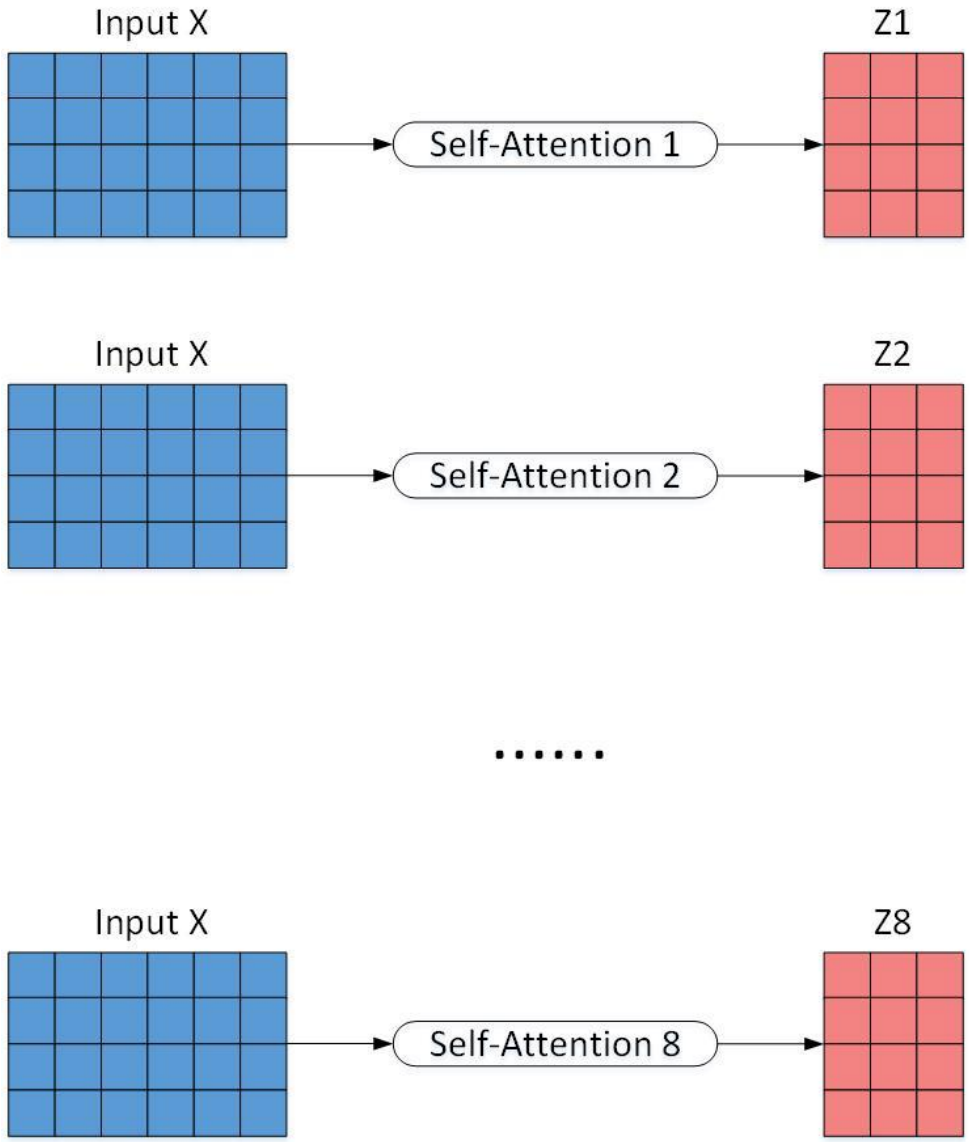


多头注意力

在这些查询（Query）、键（Key）和值（Value）的投影版本上，我们然后并行执行注意力函数，从而得到 dv 维的输出值。

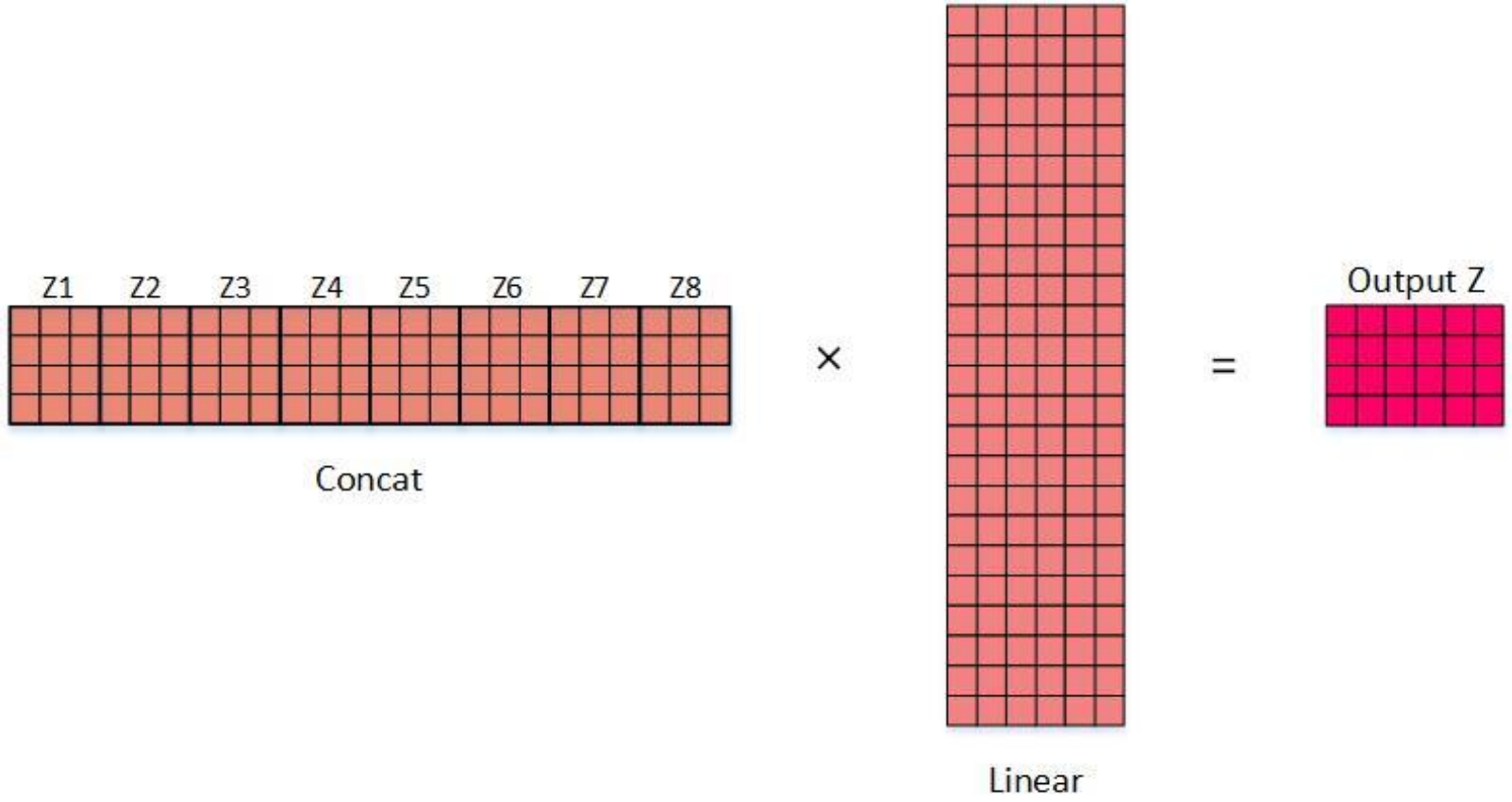
首先，输入 X 被传递给 h 个不同的自注意力（$Self-Attention$），并计算出 h 个输出矩阵 Z。

下图展示了 $h=8$ 的情况，得到 8 个输出矩阵 Z。



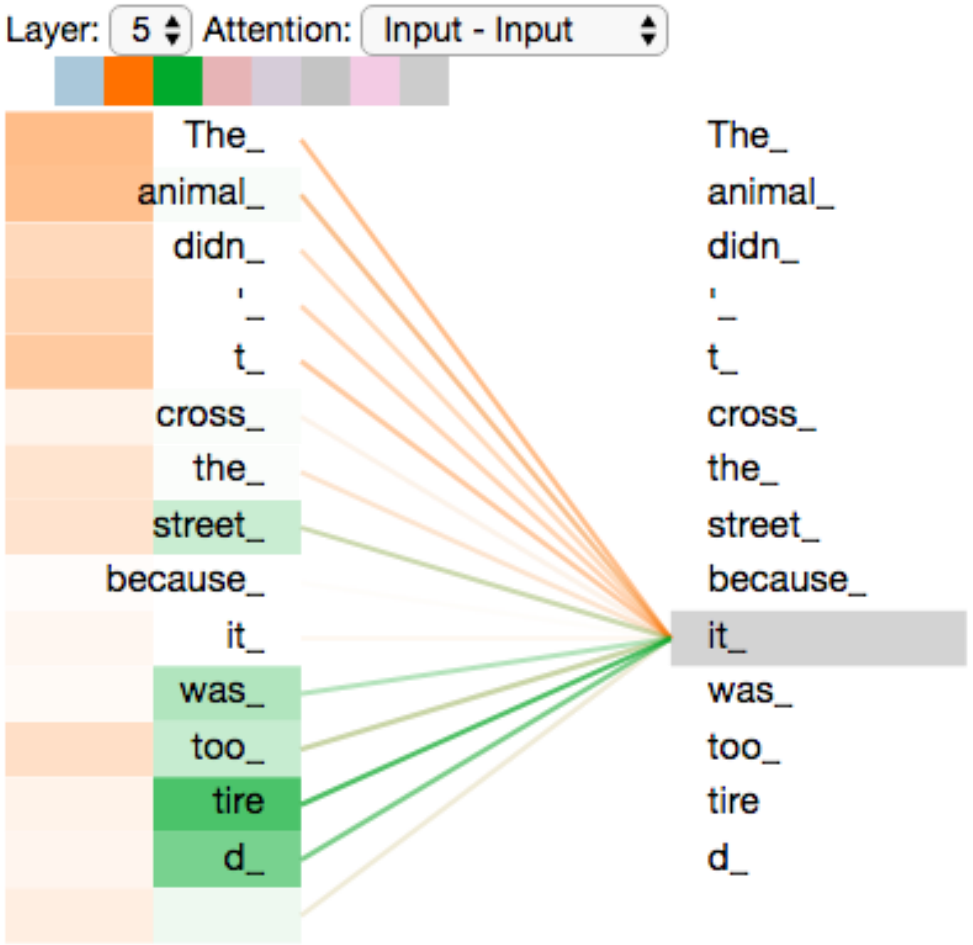
多头注意力

在得到 8 个输出矩阵 Z_1 到 Z_8 后，多头注意力（Multi-Head Attention） 将它们连接在**一起（Concat）**，然后通过一个线性层（Linear layer）得到最终的输出矩阵 Z。



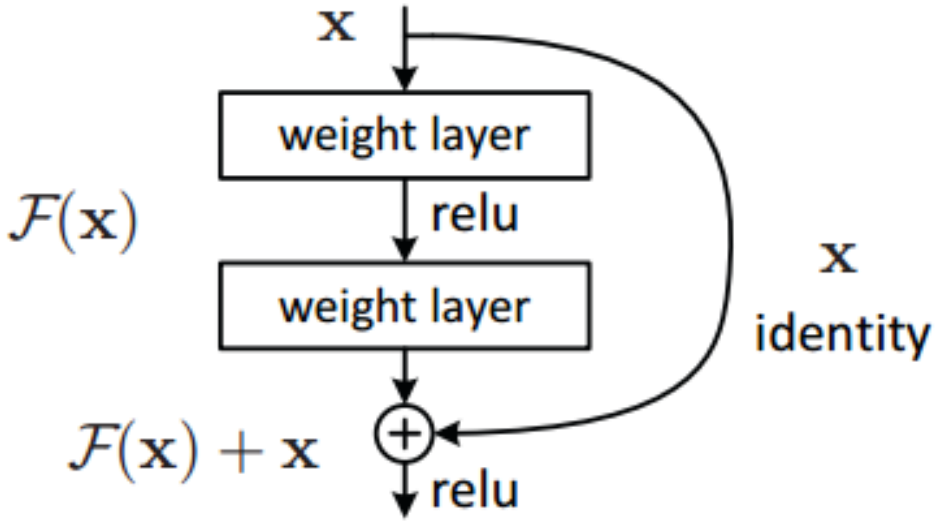
表达子空间

当我们翻译一句话，如“The animal didn’ t cross the street because it was too tired.”，我们希望知道“it”指的是哪个词。



当我们对单词“**it**”进行编码时，一个注意力头（attention head）主要关注“**the animal**”，而另一个则专注于“**tired**”。

残差连接：



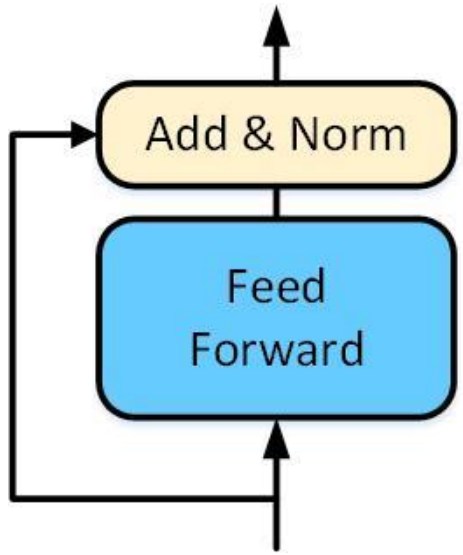
假设网络中一层在作用于输入 x 后的输出为 $F(x)$，那么在添加残差连接后，变为：$F(x) + x$，这个“$+ x$”操作就是快捷连接（shortcut）。

在 Transformer 架构图中的 Add & Norm 就是指这个操作。

- 在每个子层中，多头注意力（$Multi-Head\ Attention$） 层连接到一个 前馈网络（FFN） 层，公式如下：

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

$$W_1 \in R^{d_{model} \times d_{ff}} \quad W_2 \in R^{d_{ff} \times d_{model}}$$



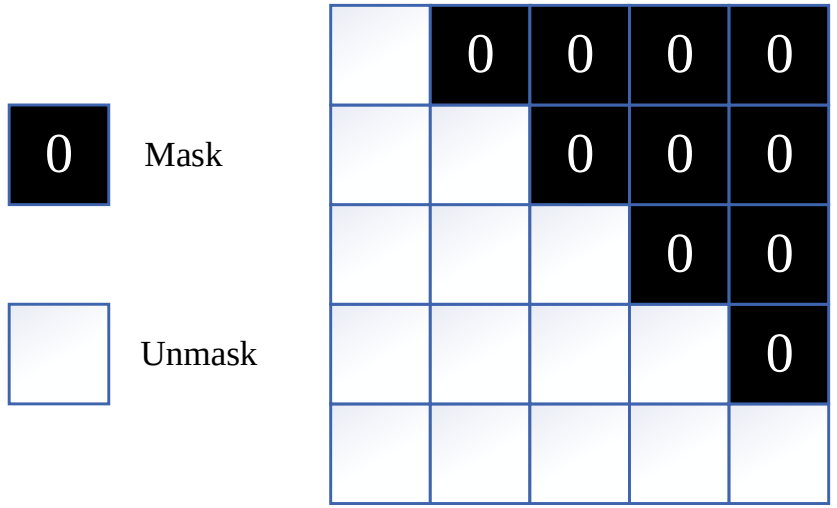
- FFN 等价于将每个位置的注意力结果映射到更大维度的特征空间，然后使用 $ReLU$ 引入非线性进行过滤，最后恢复到原始维度。

多头掩码注意力

- 与编码器的多头注意力不同，为了防止当前单词的生成依赖于未来的单词，位置 i 的预测只依赖于位置 i 前面已知的输出。

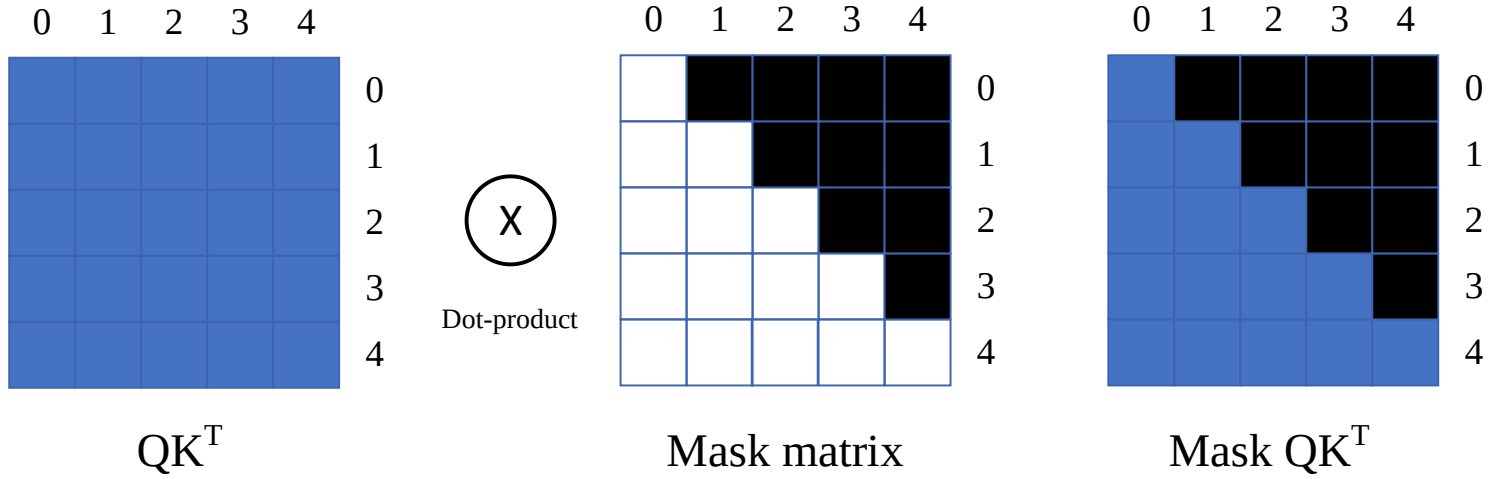
- 如何掩蔽（mask）：

如图所示，生成一个掩蔽矩阵。

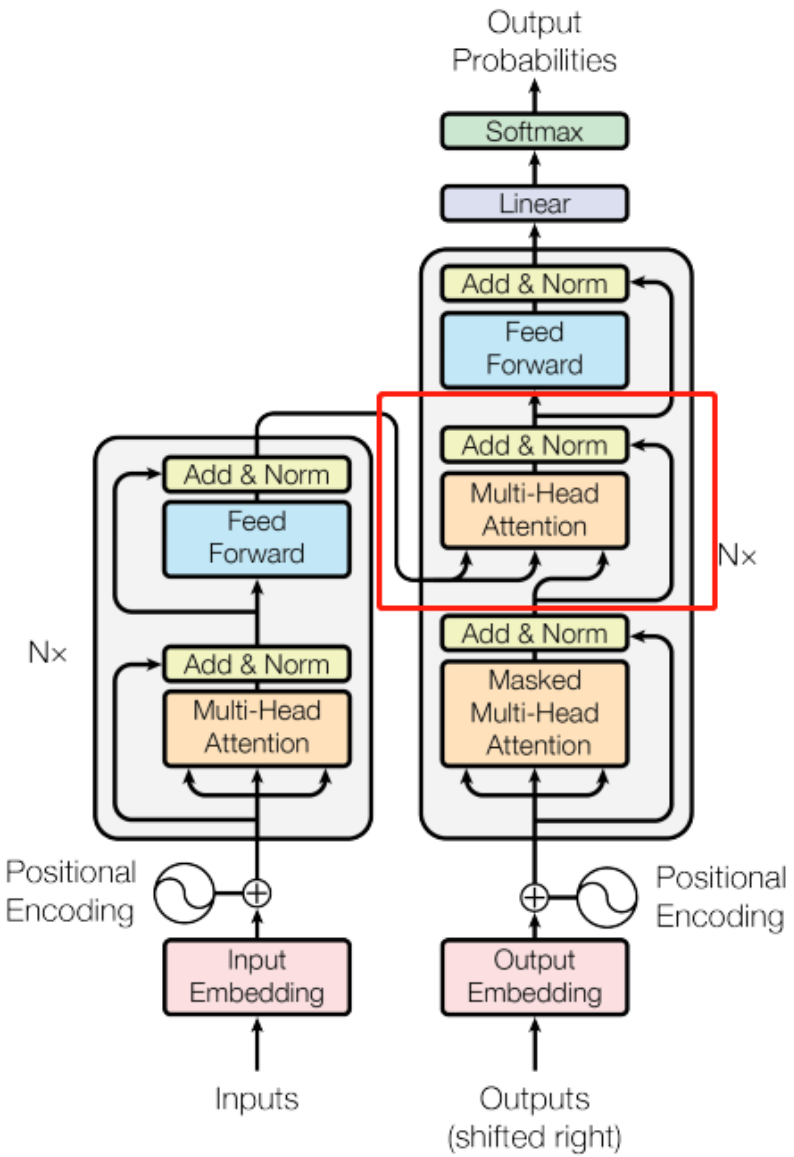


- 使用与多头注意力相同的方法来获取 QK^T 矩阵。

- 我们将 QK^T 矩阵与掩蔽矩阵进行点积操作。



编码-解码注意力

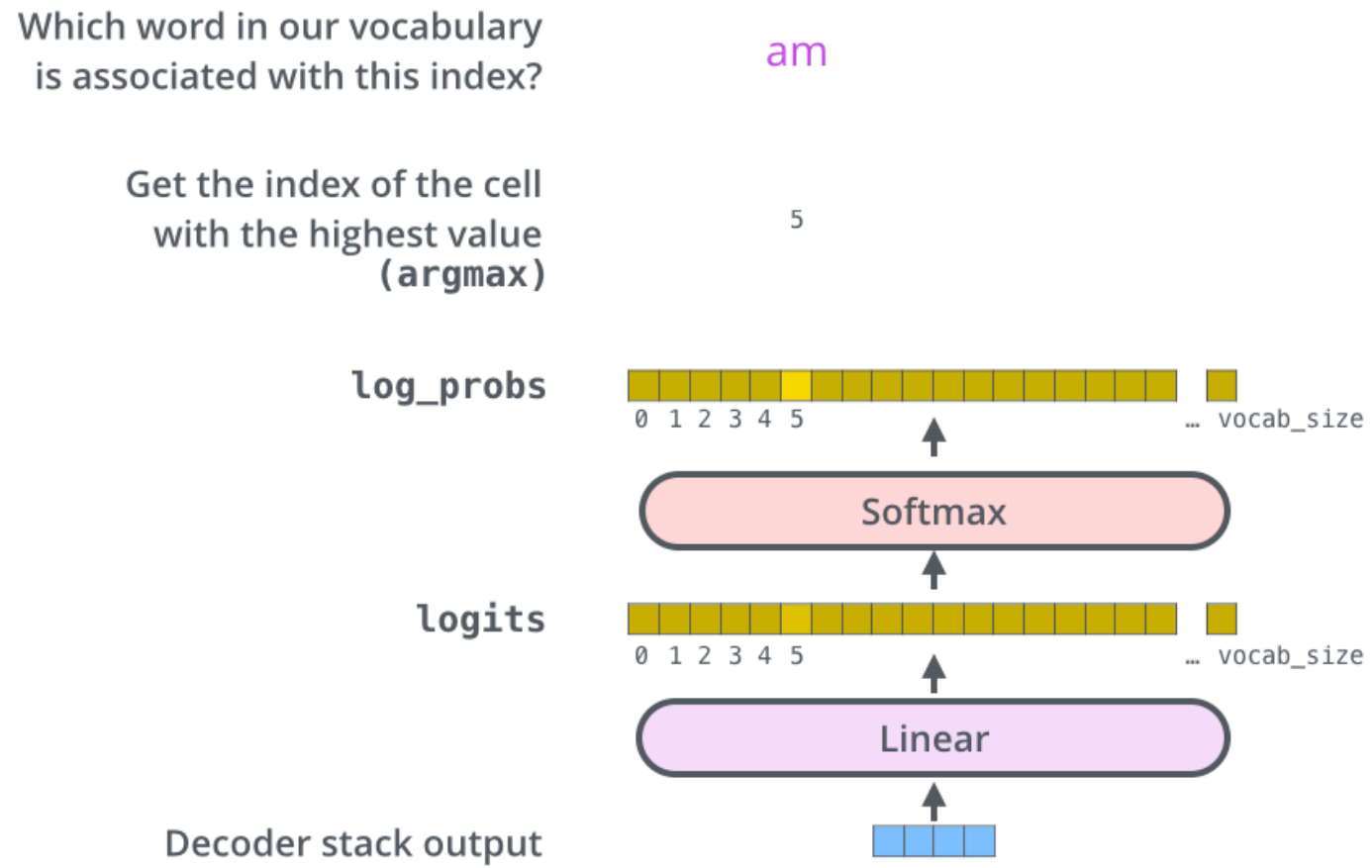


在编码器的多头注意力层中，三个输入矩阵（Q、K、V）都来自前一个编码器的输出。

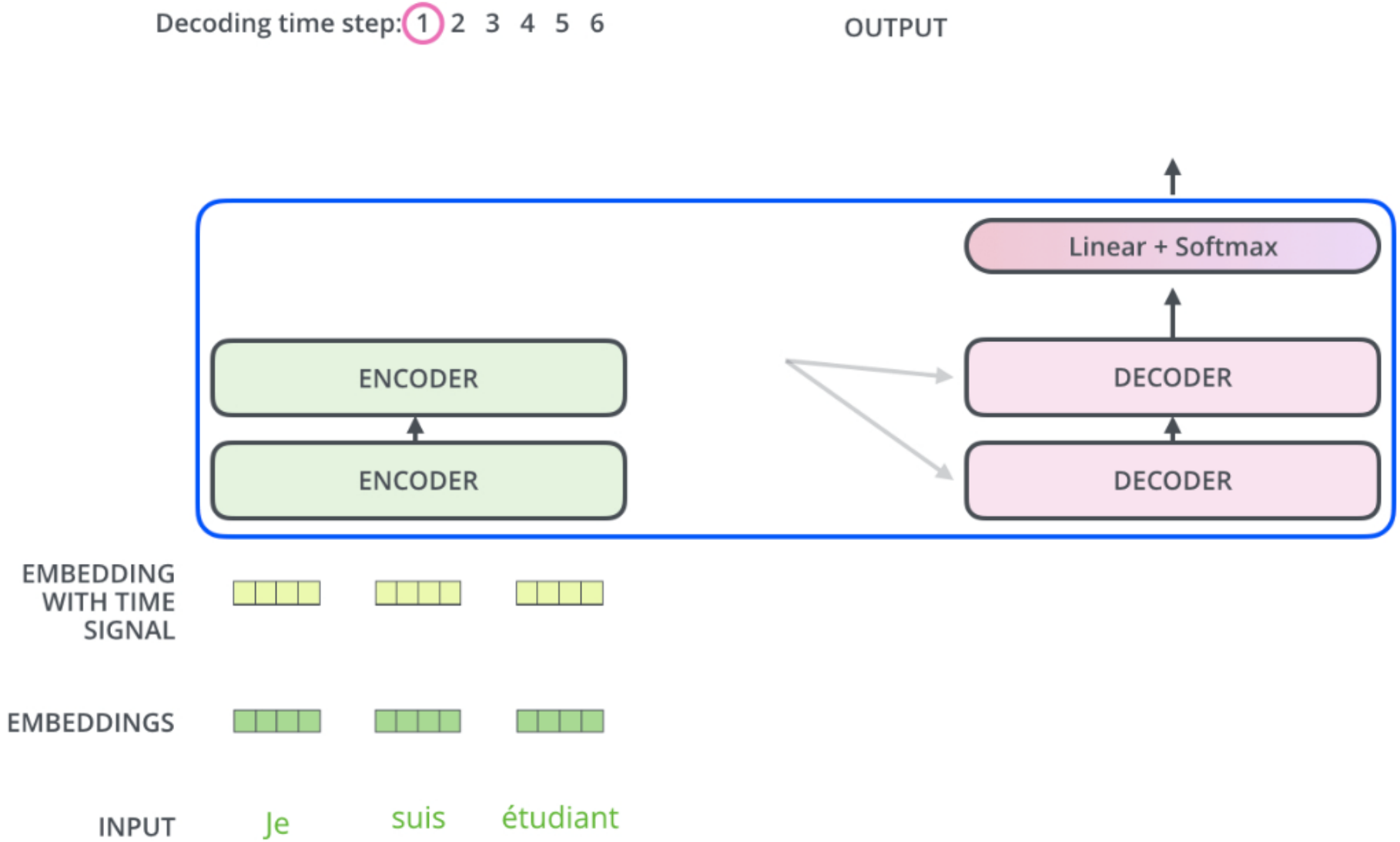
在解码器中，输入矩阵（K、V）来自第六个编码器的输出，输入矩阵（Q）来自掩蔽多头注意力的输出。

Linear+Softmax

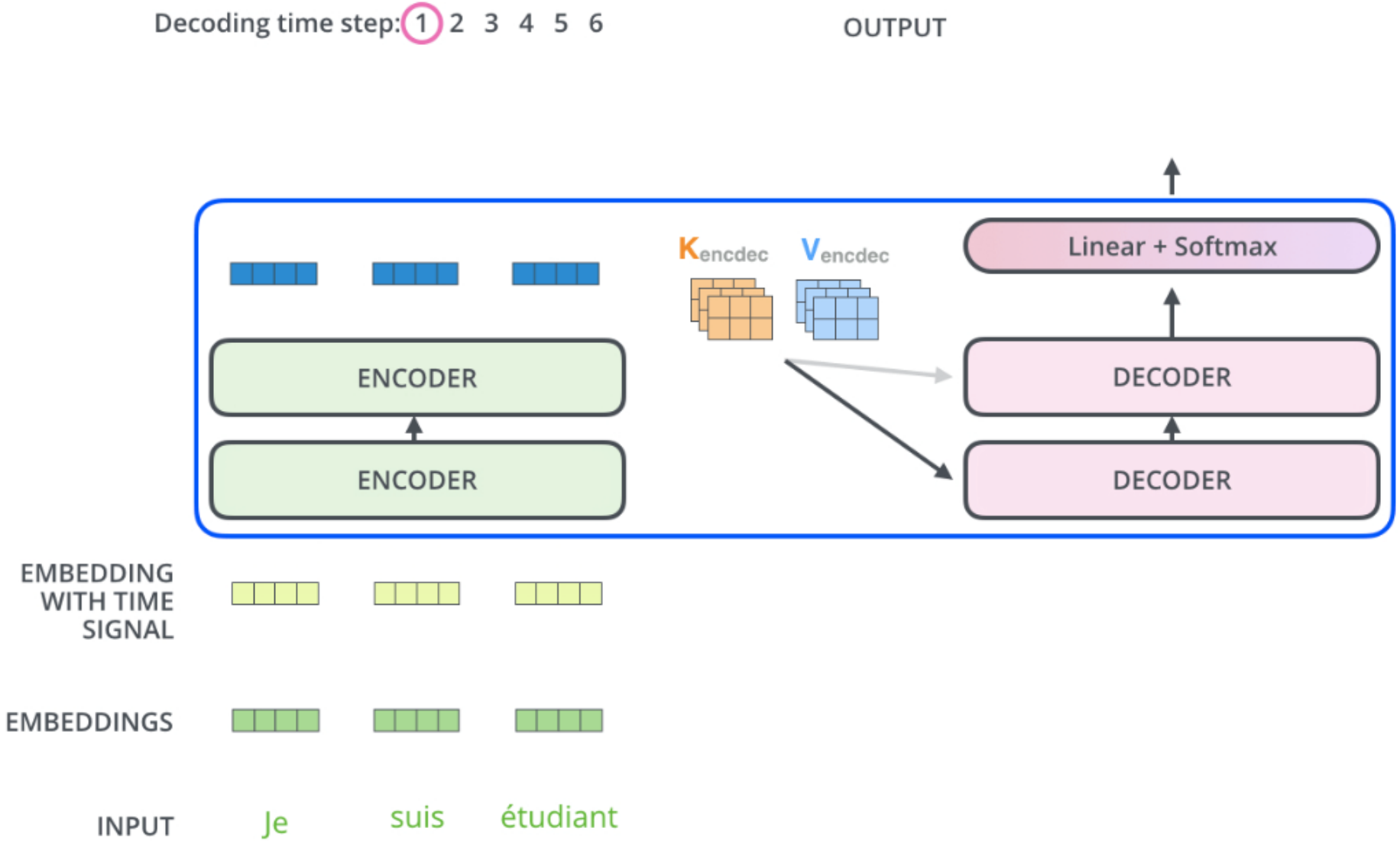
在第六个解码器输出一个向量后，该向量将被送入线性层。然后，这些分数将输入到 $softmax$ 层，以获得下一个单词的概率。



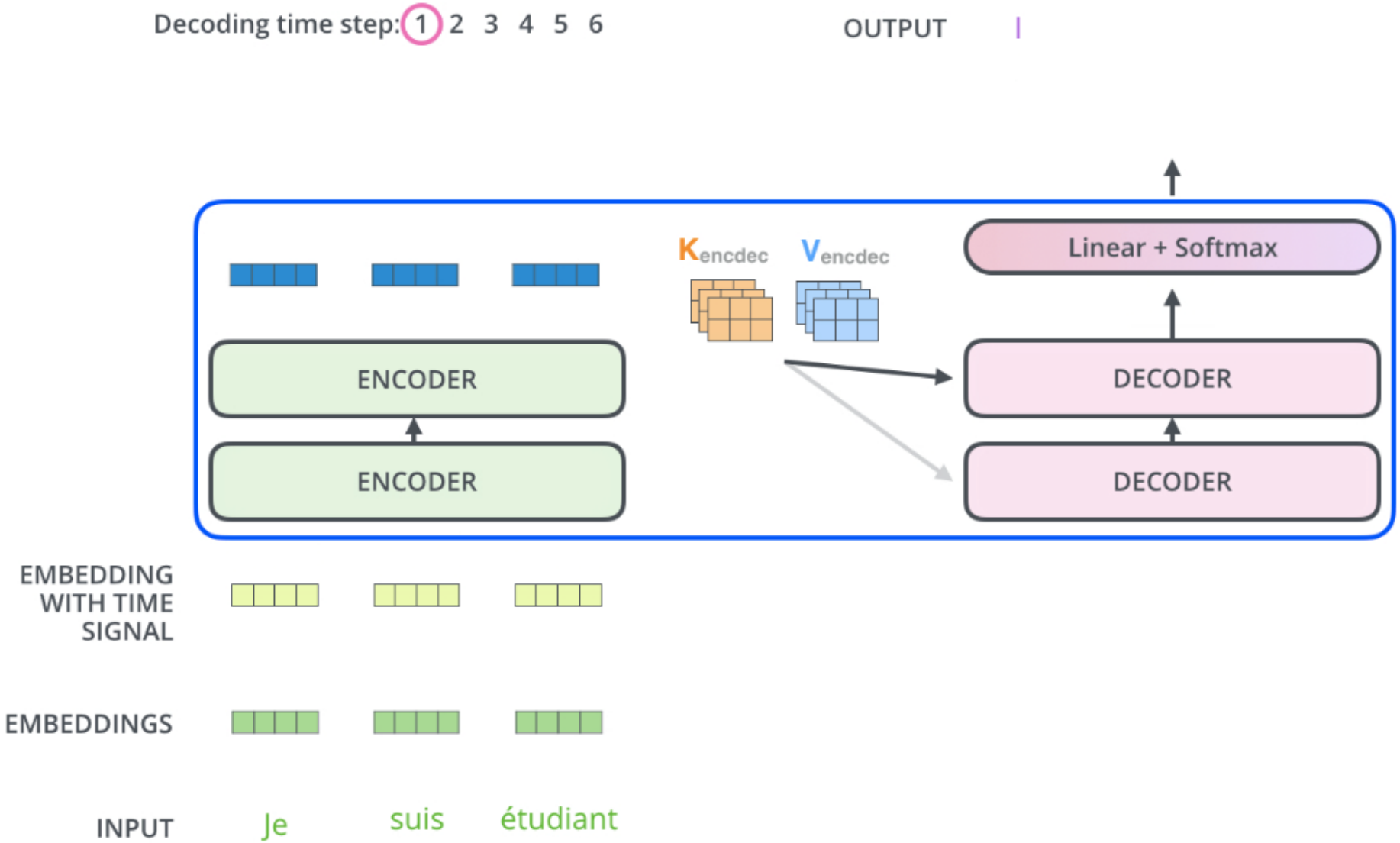
动态过程


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

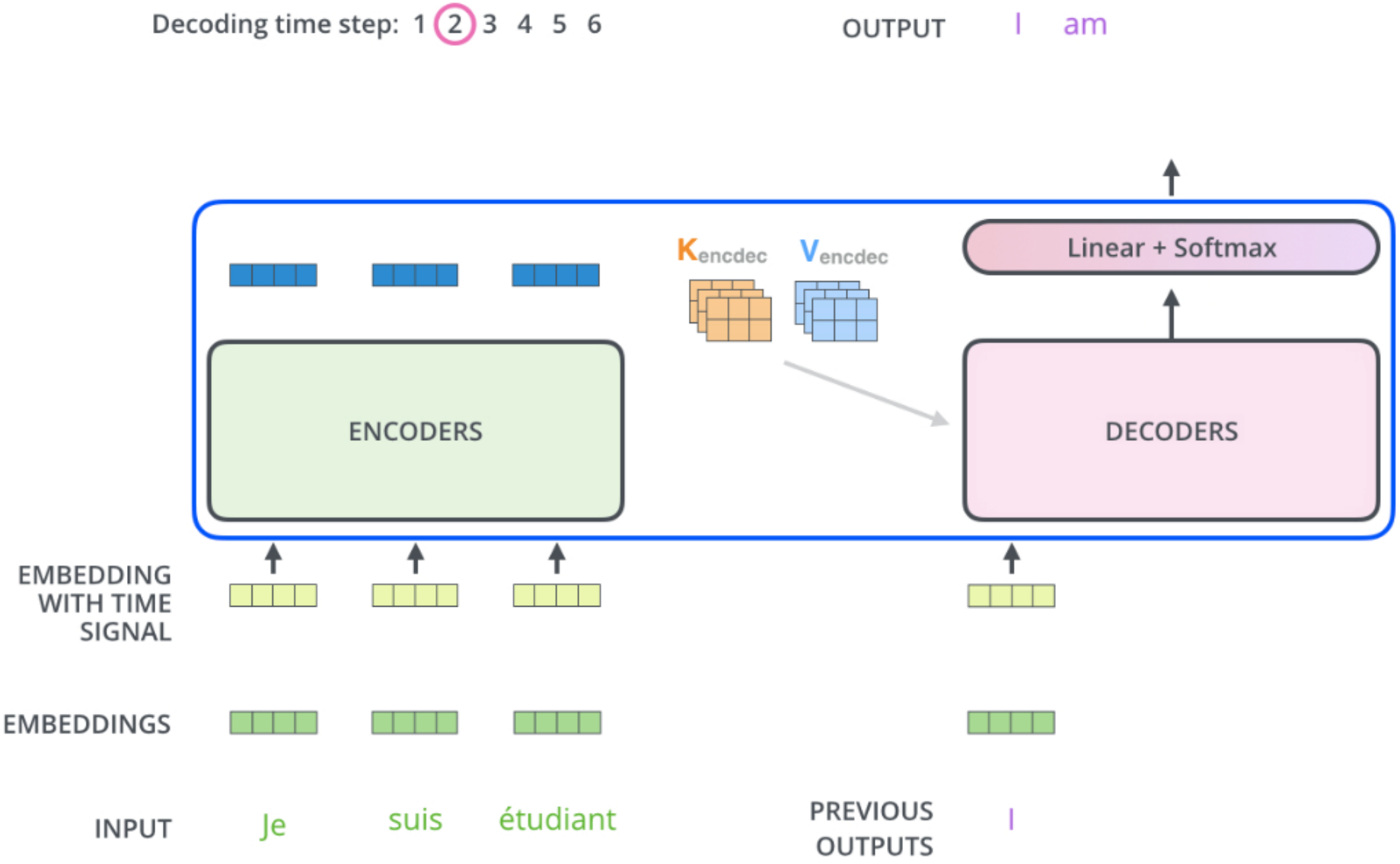
动态过程


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
K encdec
V encdec
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

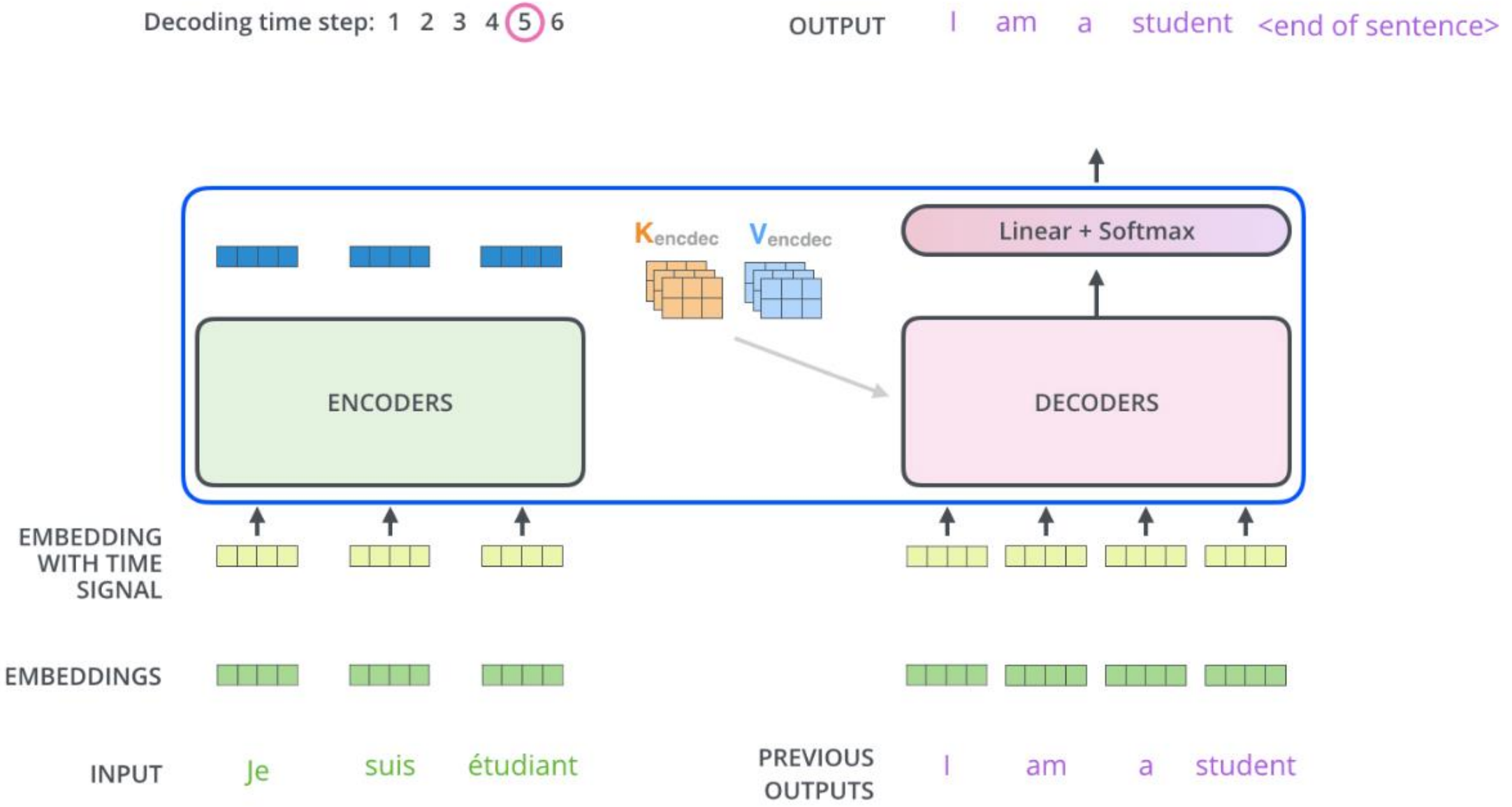
动态过程


Decoding time step: 1 2 3 4 5 6
OUTPUT I
Linear + Softmax
K encdec
V encdec
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

动态过程


Decoding time step: 1 2 3 4 5 6
OUTPUT I am
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT Je suis étudiant
PREVIOUS OUTPUTS I

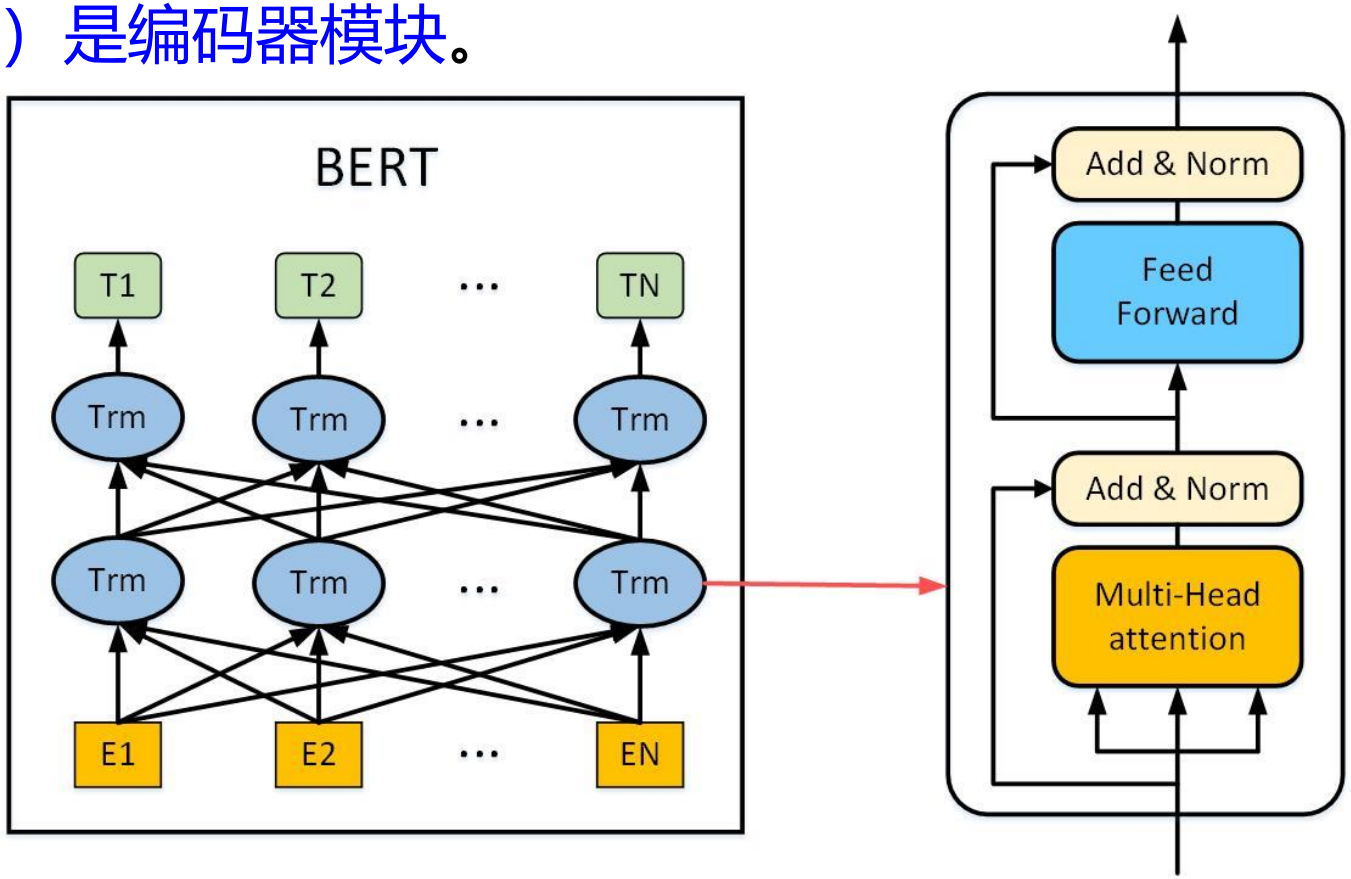
动态过程


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a student <end of sentence>
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am a student

目录
Contents

Transformer的应用——BERT

BERT（Bidirectional Encoder Representations from Transformers）是一个预训练模型，使用的是 Transformer 的编码器部分。正如我们所看到的，Trm（Transformer）是编码器模块。

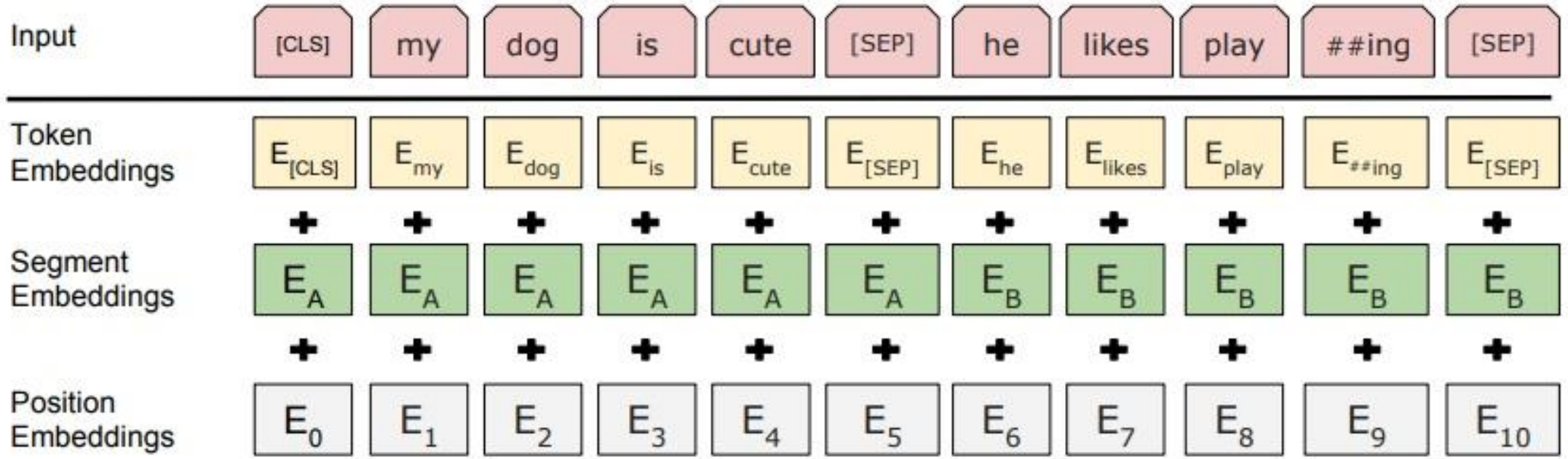


为了使模型输出的文本语义表示能够描述语言的本质，BERT 使用了两种创新的方法：Masked LM 和 Next Sentence Prediction。

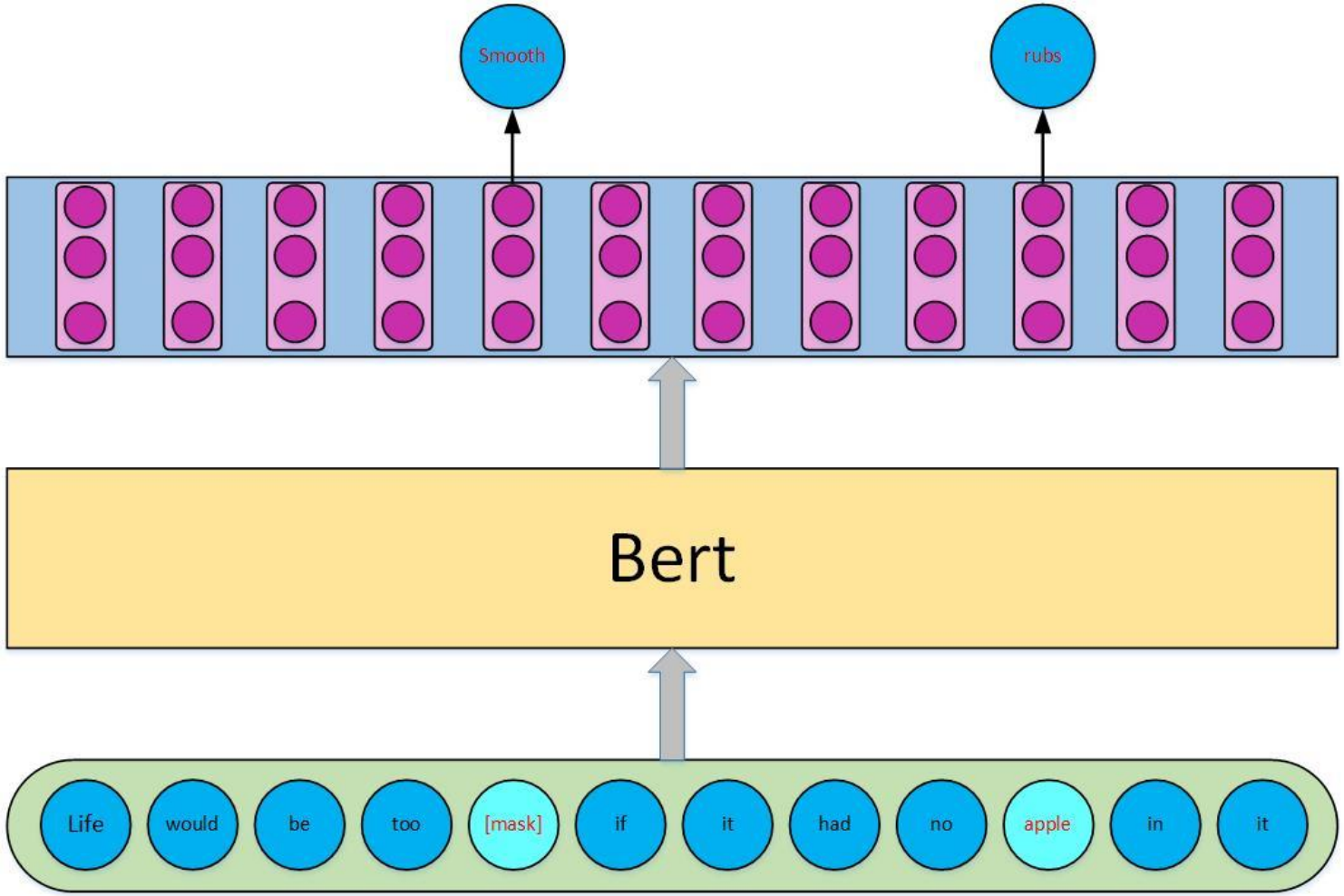
Input	[CLS] the man went to [mask] store [SEP] he bought a gallon [mask] milk [SEP]
Label	IsNext

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

BERT 的输入由三部分组成：Token embeddings、Segment embeddings 和 Position embeddings。例如，对于句子 "my dog is cute. He likes playing." ，其输入表示方式如下：



给定一句话，随机删除其中一个或多个单词，剩下的单词需要用来预测被删除的单词，如下图所示。



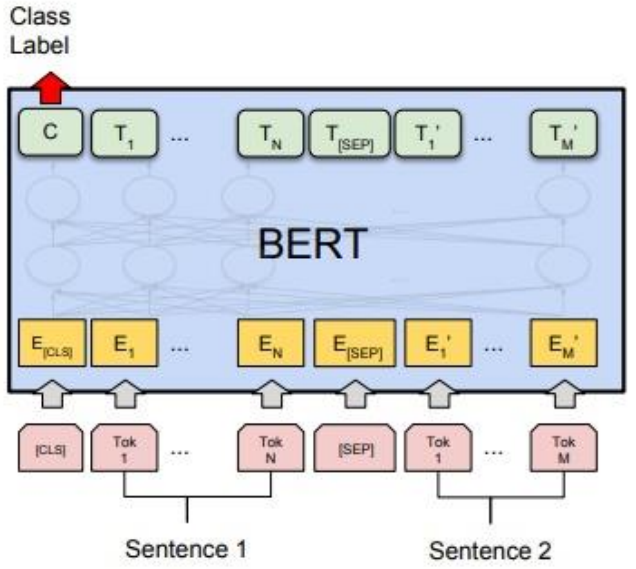
在原句中被删除的词汇，80%的情况下会被替换为一个特殊符号 [MASK]，10%的情况下会被替换为一个任意的词汇，其余10%的情况下原词汇保持不变。

percent	Former	later
80%	My dog is hairy	My dog is [mask]
10%	My dog is hairy	My dog is apple
10%	My dog is hairy	My dog is hairy

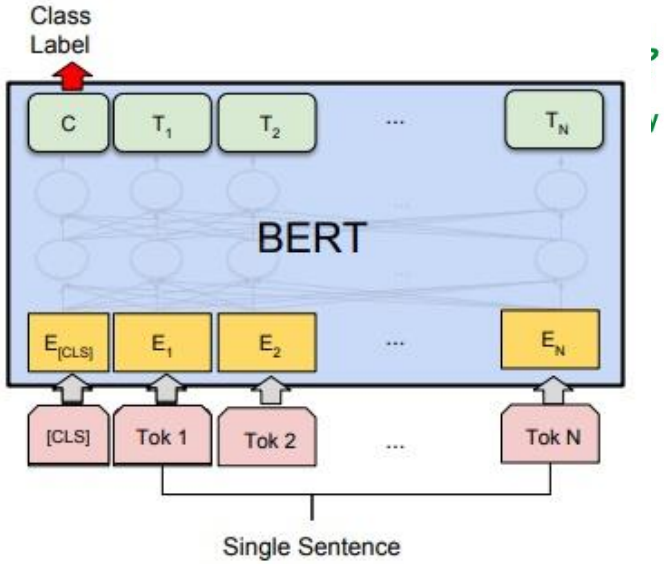
Fine-Tuning

对于序列级别的分类任务，BERT 直接使用第一个 [CLS] 标记的最终隐藏状态，然后加上一层权重来预测 Softmax 标签。

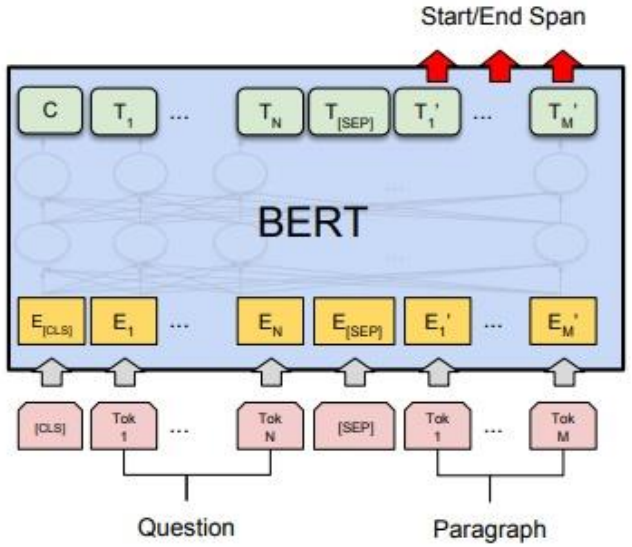
其他预测任务需要进行一些调整，如图所示：



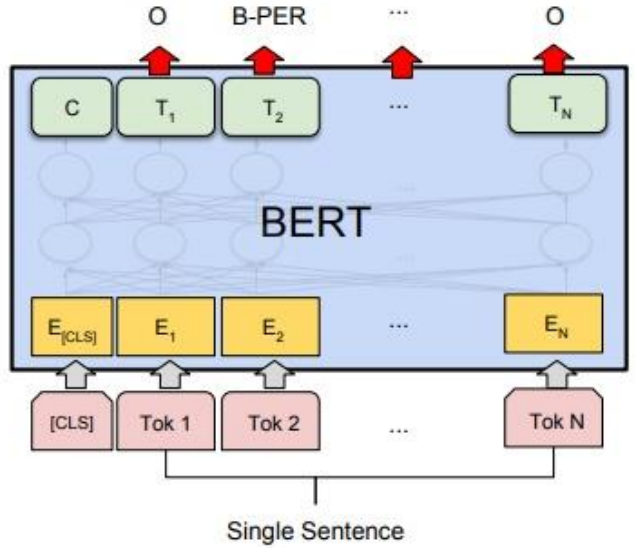
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



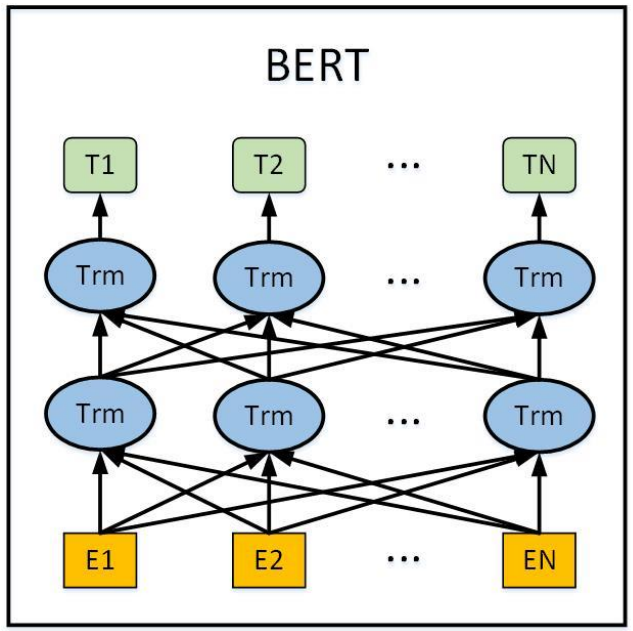
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

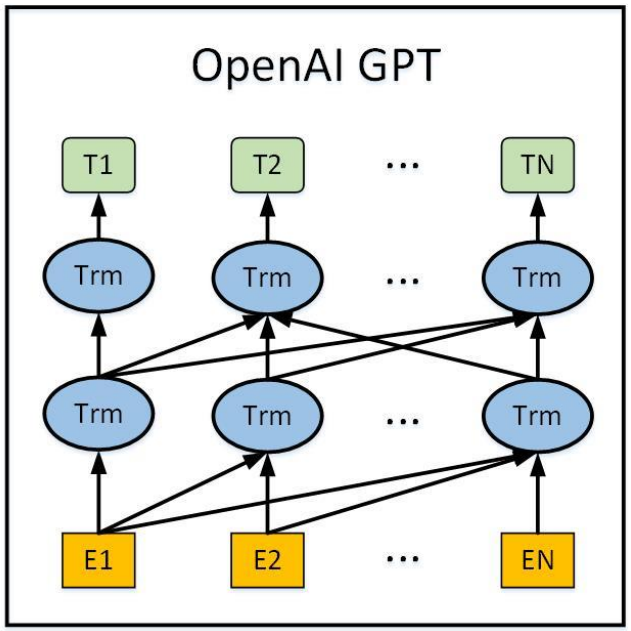
对比

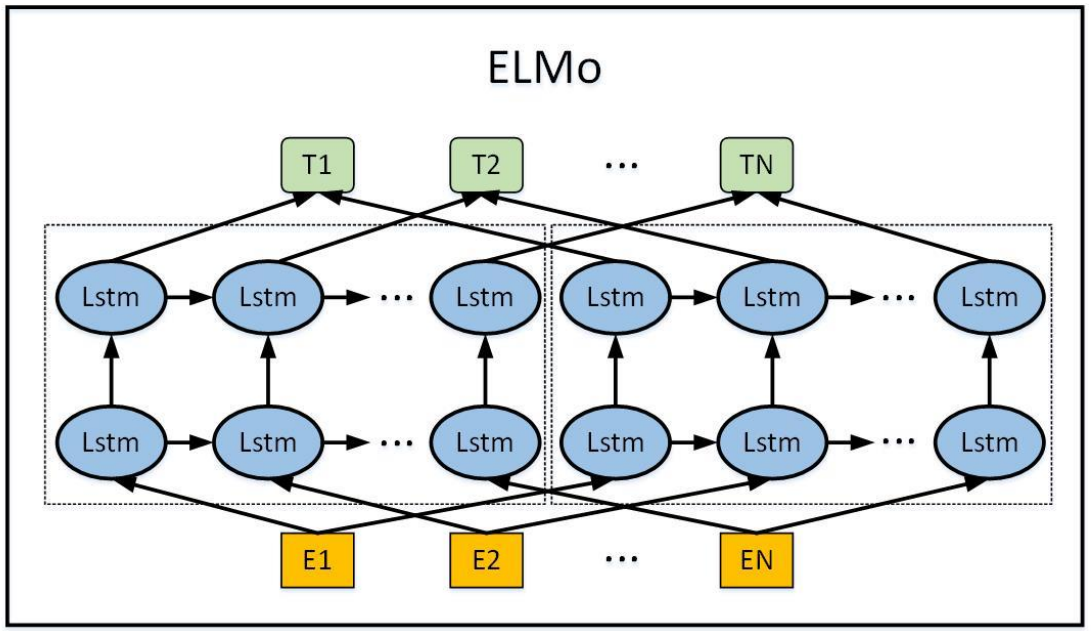
BERT 使用的是双向 Transformer。

OpenAI GPT 使用的是从左到右的 Transformer。

ELMo 使用独立训练的从左到右和从右到左的 LSTM 的连接来生成下游任务的特征。







GPT & GPT-2

与 GPT 相同，GPT-2 仍然是一个单向语言模型。

1. GPT 和 GPT-2 使用的是 Transformer 的解码器，并删除了编码器-解码器的注意力层。

2. GPT 和 GPT-2 使用的是 Transformer 的编码器，并将多头注意力层（Multi-head Attention）更改为掩蔽多头注意力层（Masked Multi-head Attention）。

模型	GPT	BERT	GPT2
基础模块	解码	编码	解码
方向	单程	双程	单程
层数	12	24	48
参数	1亿	3亿	15亿
过程	预训练+精调	预训练+精调	无监督多任务学习

- 优点：

Transformer 摒弃了 NLP 中最基本的 RNN 或 CNN。多头注意力的使用具有足够的创新性，并取得了非常好的效果。它不仅可以应用于机器翻译领域，还可以应用于其他 NLP 领域。

- 缺点：

Transformer 丢失的位置位置信息在 NLP 中实际上非常重要。将位置编码（Position Embedding）添加到特征向量中仅仅是一个补救措施，并没有改变 Transformer 结构中固有的缺陷。

语料万千藏知识，模型千变解语意。
注意焦点连长忆，Transformer绘新篇章。