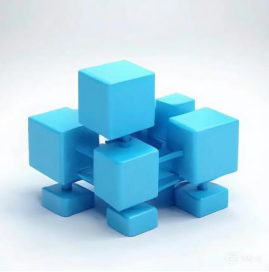
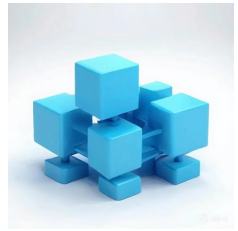


大模型基础: Transformer原理

主讲：东东东

B站视频链接:
https://www.bilibili.com/video/BV1BZTszKEbv/?share_source=copy_web&vd_source=f23fdab1cf57871b257305ebe143b9c2

大模型基础：Transformer原理



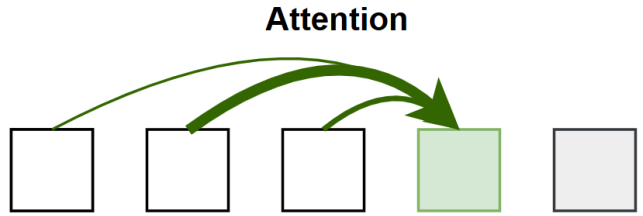
1. 大模型是如何生成文本的？

2. 大模型的基础 – Transformer结构

- Transformer: 让语言模型拥有上下文理解能力，**“注意”**到每一个前文词汇
- **模块化结构：**
  - **1.注意力模块（Attention Module)**
  - 2.全连接神经网络模块 (Feed-Forward Network Module/Multilayer Perceptron)

3. Transformer中的注意力（Attention）机制

- 目的：让大模型拥有上下文理解能力 < --- > 让大模型“注意”到上下文词汇 < --- >让文本里的每一个词（Token）包含前文有效信息
  - 输入：初始词向量 = E →注意力层 → 输出：包含前文信息的词向量=E’
- 让当前词“注意”到每一个前文词汇，“注意力”通过**Q,K,V**计算
  - Q- Query 查询
  - K - Keys 键
  - V – Values 值
- 根据“注意力”程度，把前文词汇语义信息加入到当前词汇中使大模型能理解文本内容

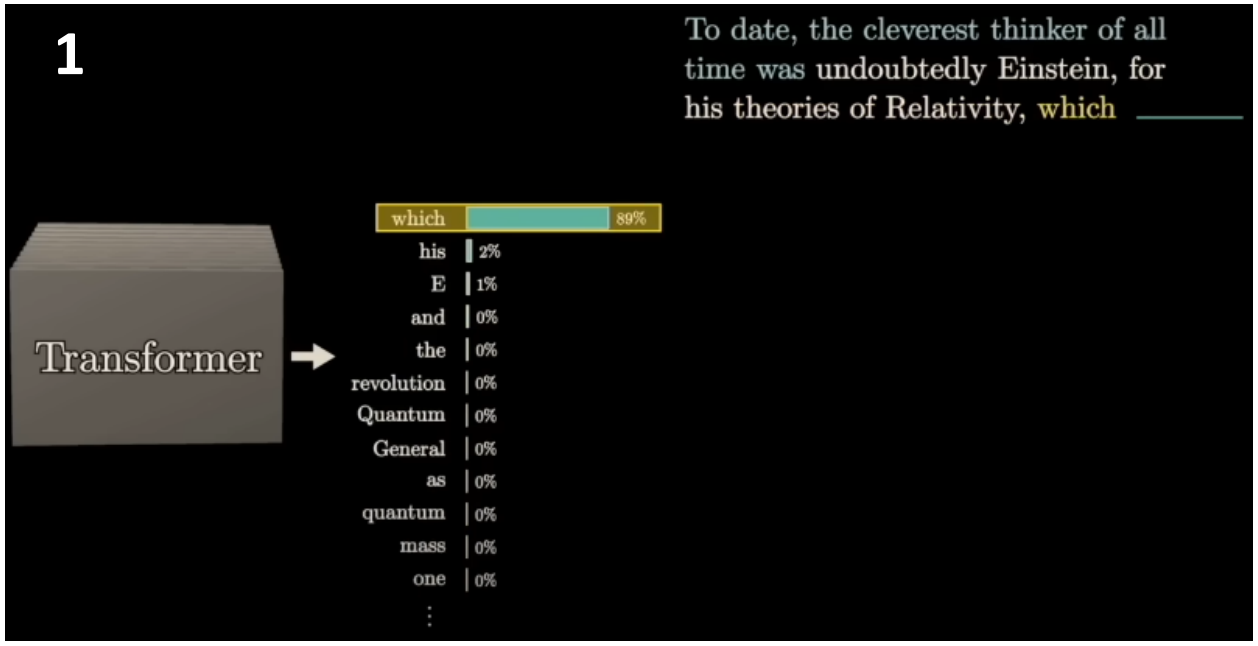


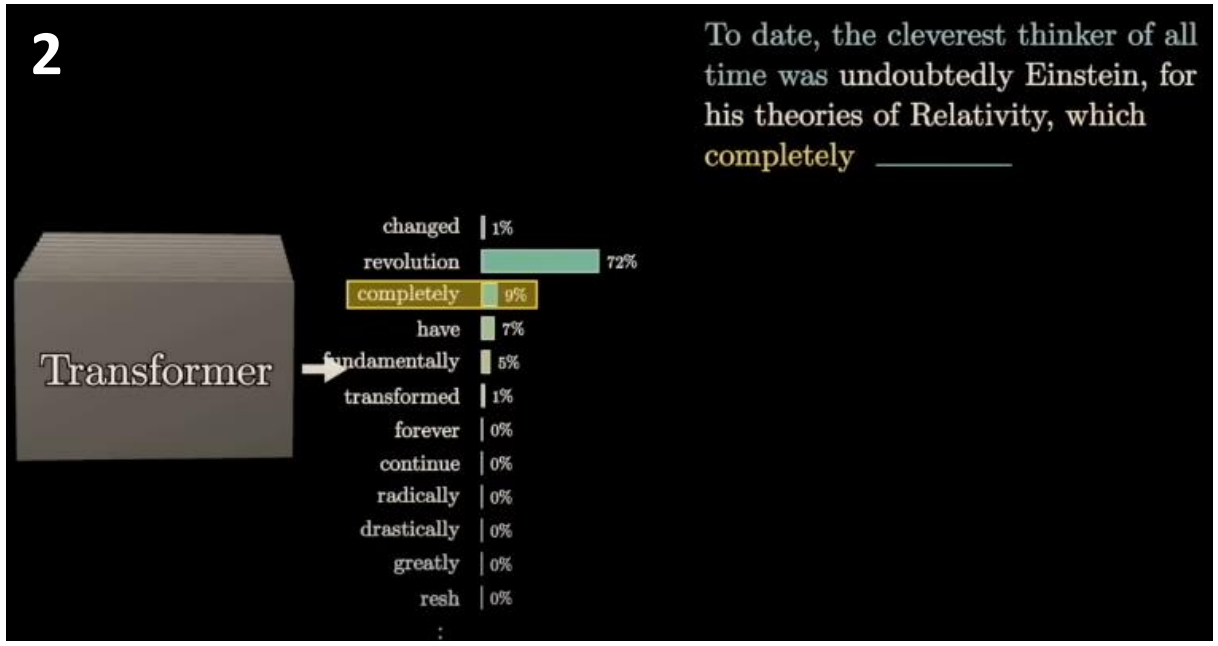
The cat

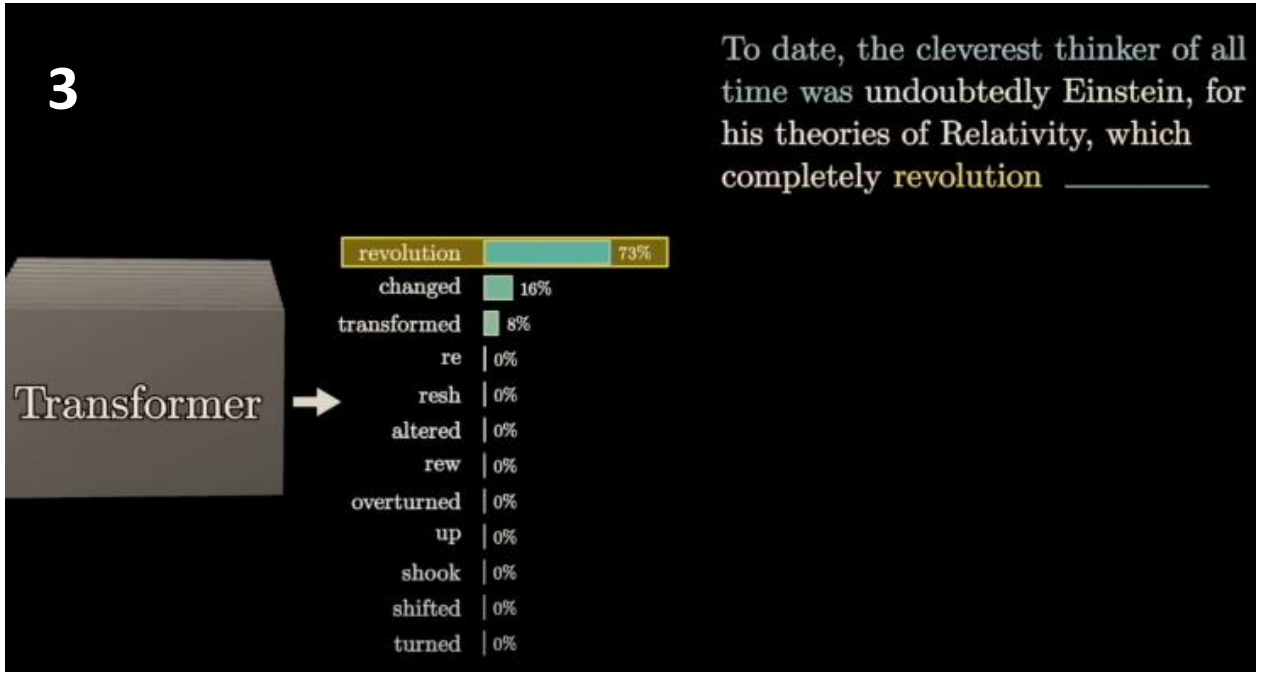
1.大模型是如何生成文本/产生对话的?

大模型每次预测下一个“最有可能”的词汇，循环往复

大模型是如何生成文本的?







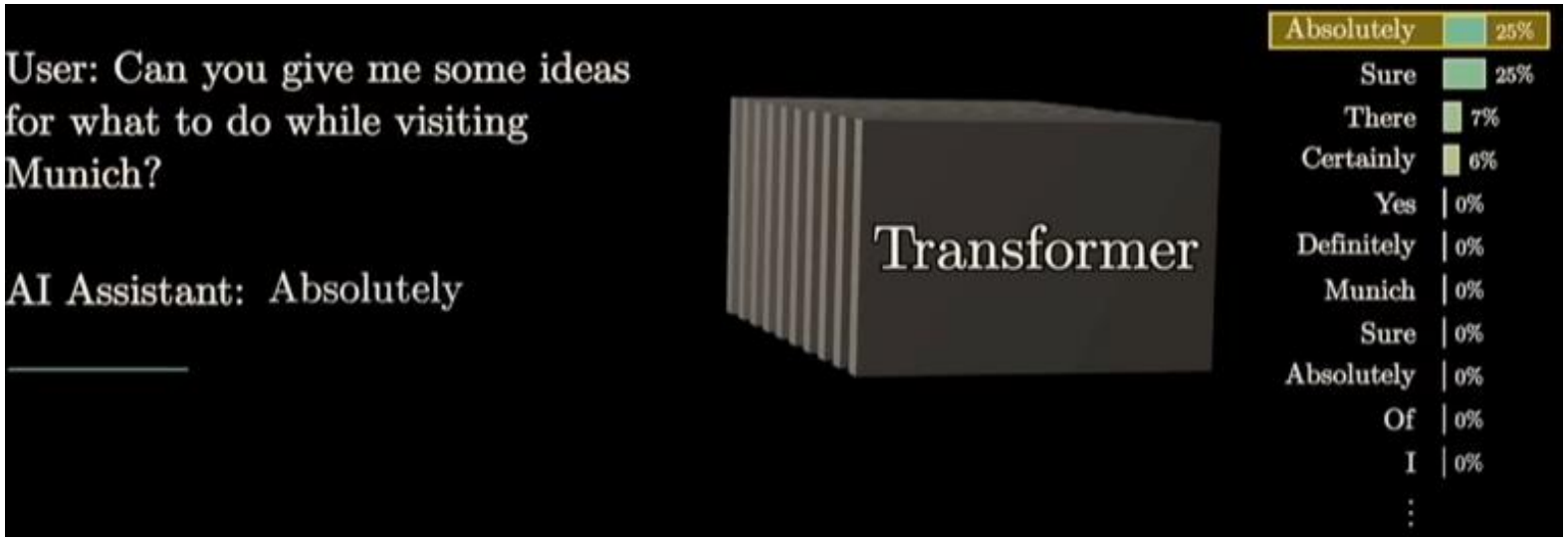
Transformer (2017)是大模型的基础，现有的各种大模型都基于Transformer中的核心概念-**注意力（Attention）**机制

大模型是如何生成文本的?

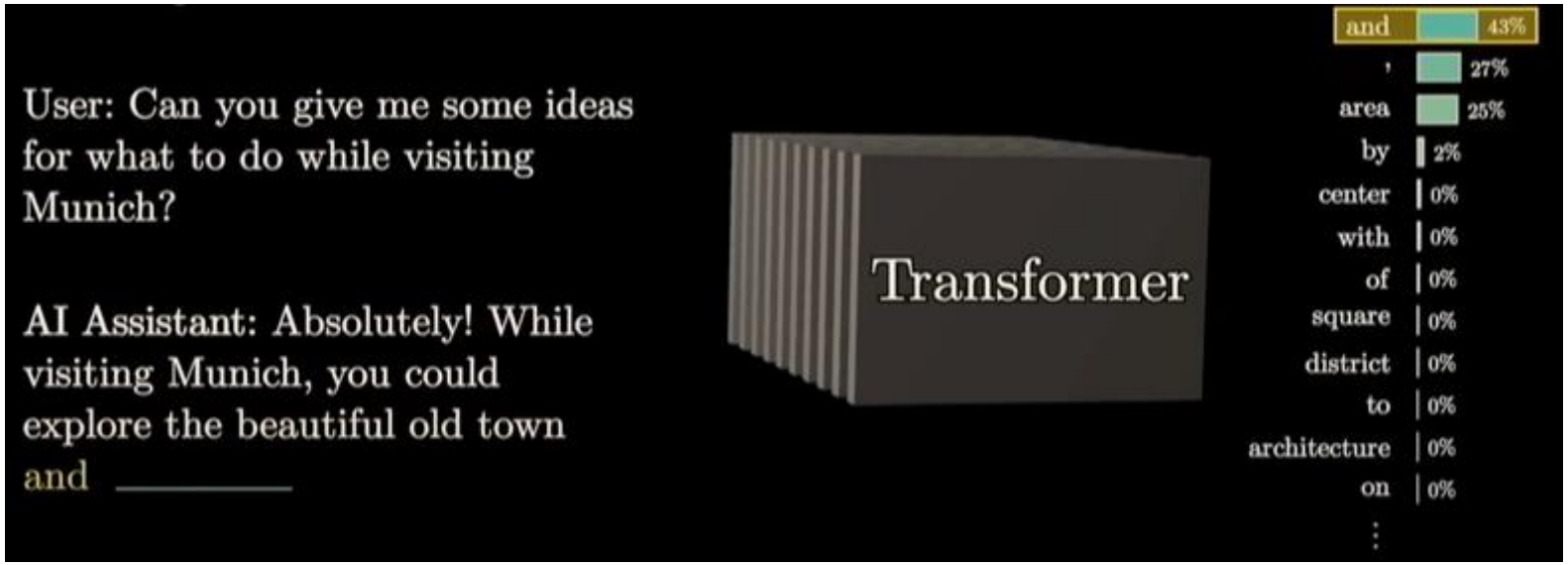
大模型每次预测***基于已有文本***的***下一个"最有可能"***的词汇，循环往复，最终生成大段文字 （见图1，2，3）

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

大模型是如何回答问题的?



循环往复

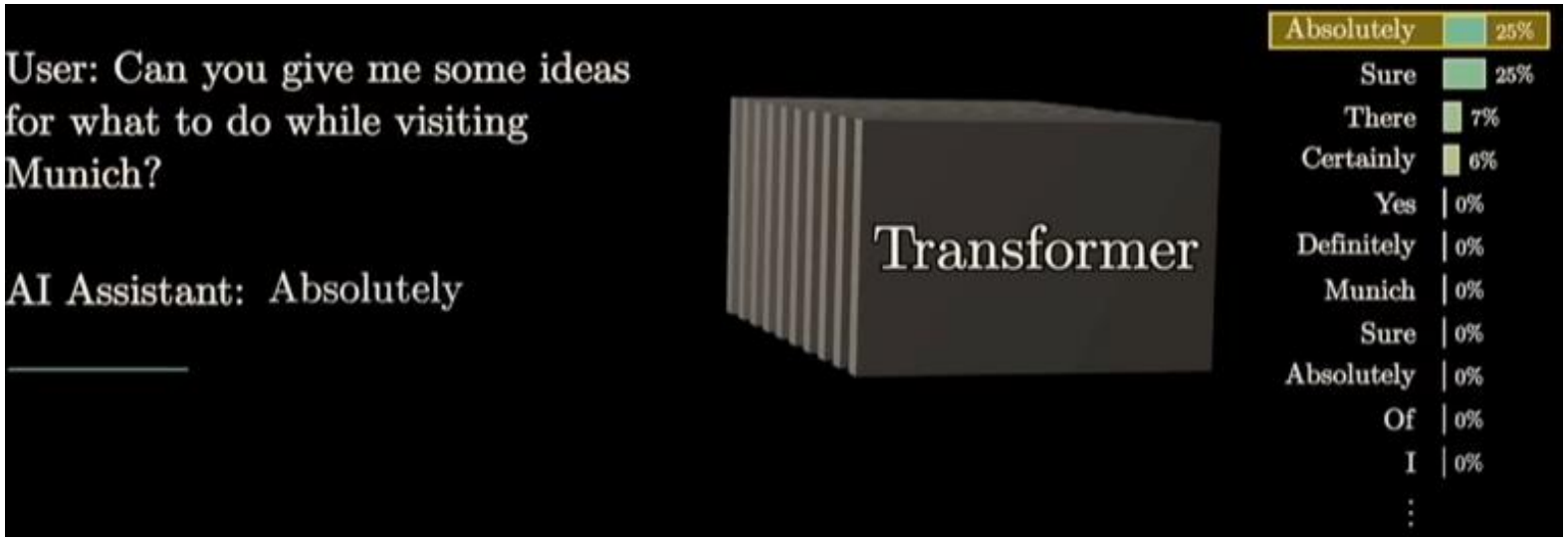


图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

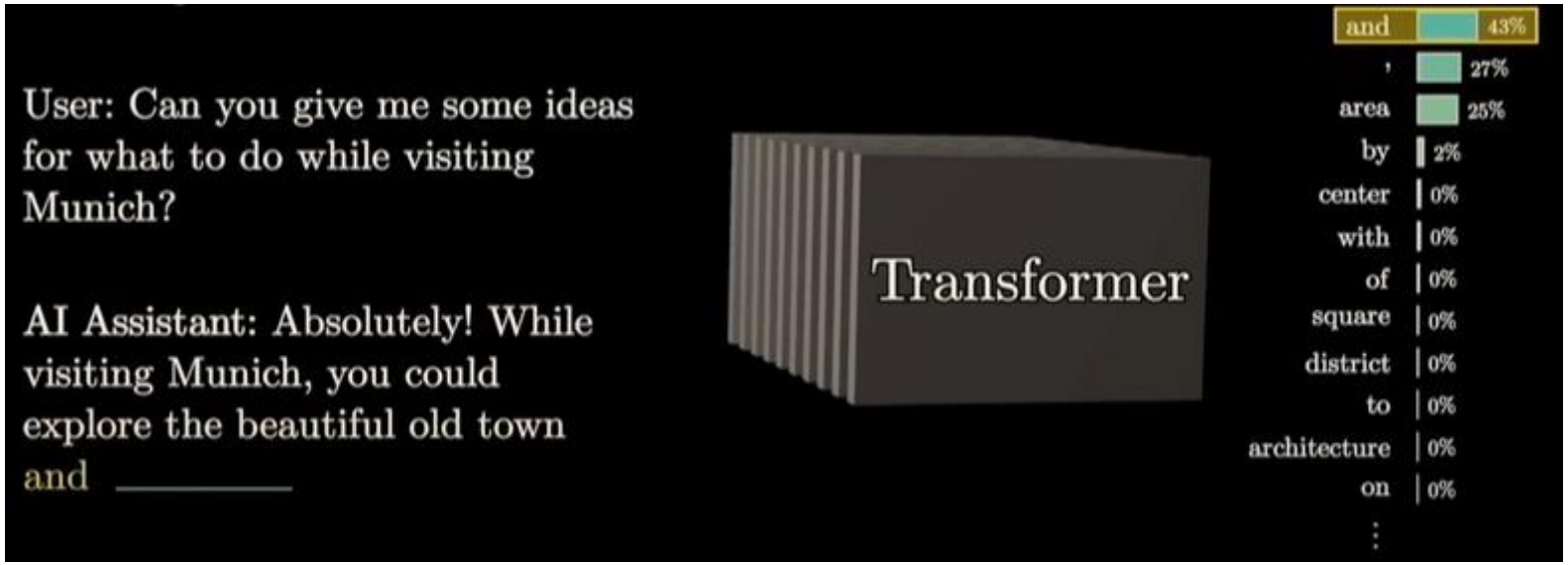
大模型是如何回答问题的?

- 和生成文本的原理一样，每次输出**下一个“最有可能”的词**，**循环往复**，最终生成回答

大模型是如何生成文本/回答问题的?



循环往复



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

大模型是如何回答问题的?

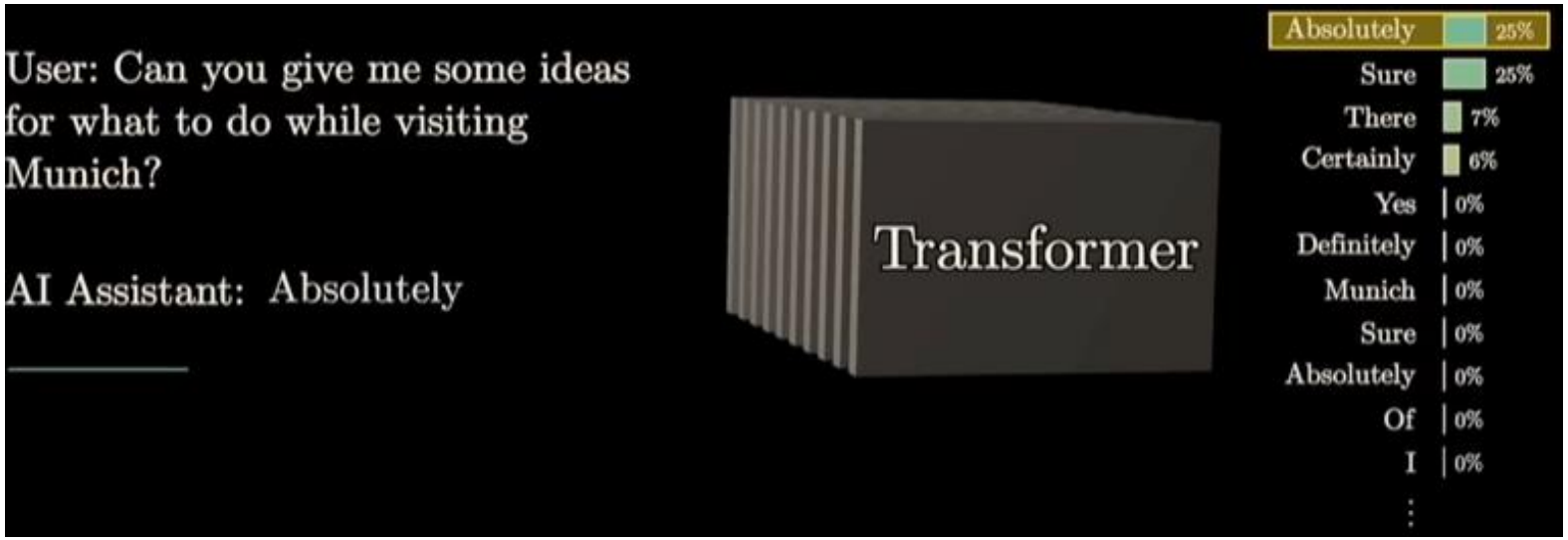
- 和生成文本的原理一样，每次输出**下一个"最有可能"的词**，**循环往复**，最终生成回答

所有主流大模型的设计基础 – Transformer

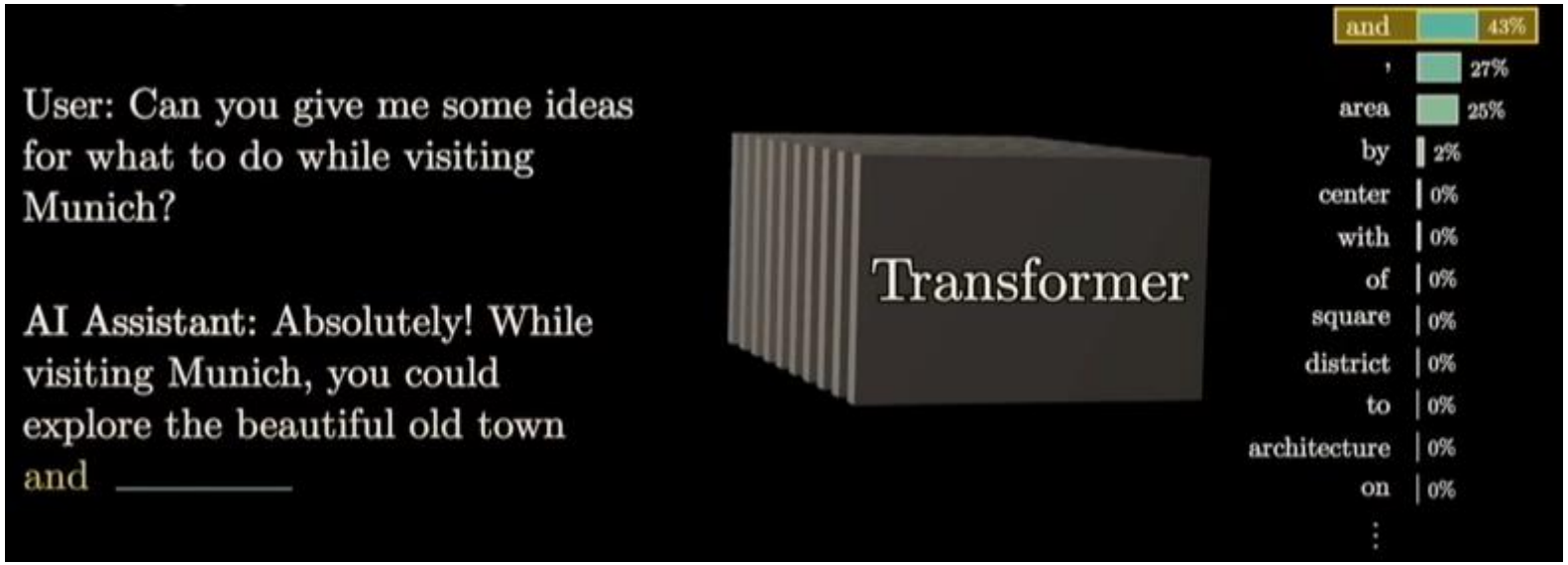
Qwen

为什么是Transformer？Transformer 相比其他自然语言模型的优势
-支持并行输入（充分利用GPU的并行计算能力）

大模型是如何生成文本/回答问题的?



循环往复



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

大模型是如何回答问题的?

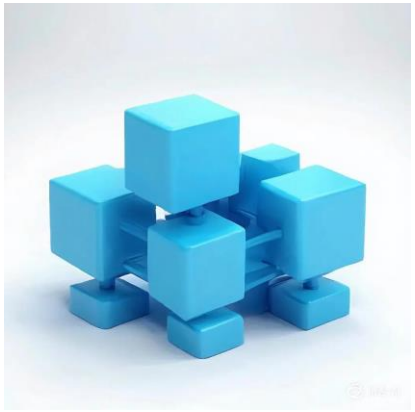
- 和生成文本的原理一样，每次输出**下一个“最有可能”的词**，**循环往复**，最终生成回答

所有主流大模型的设计基础 – Transformer

Qwen

为什么是Transformer? Transformer 相比其他自然语言模型的优势

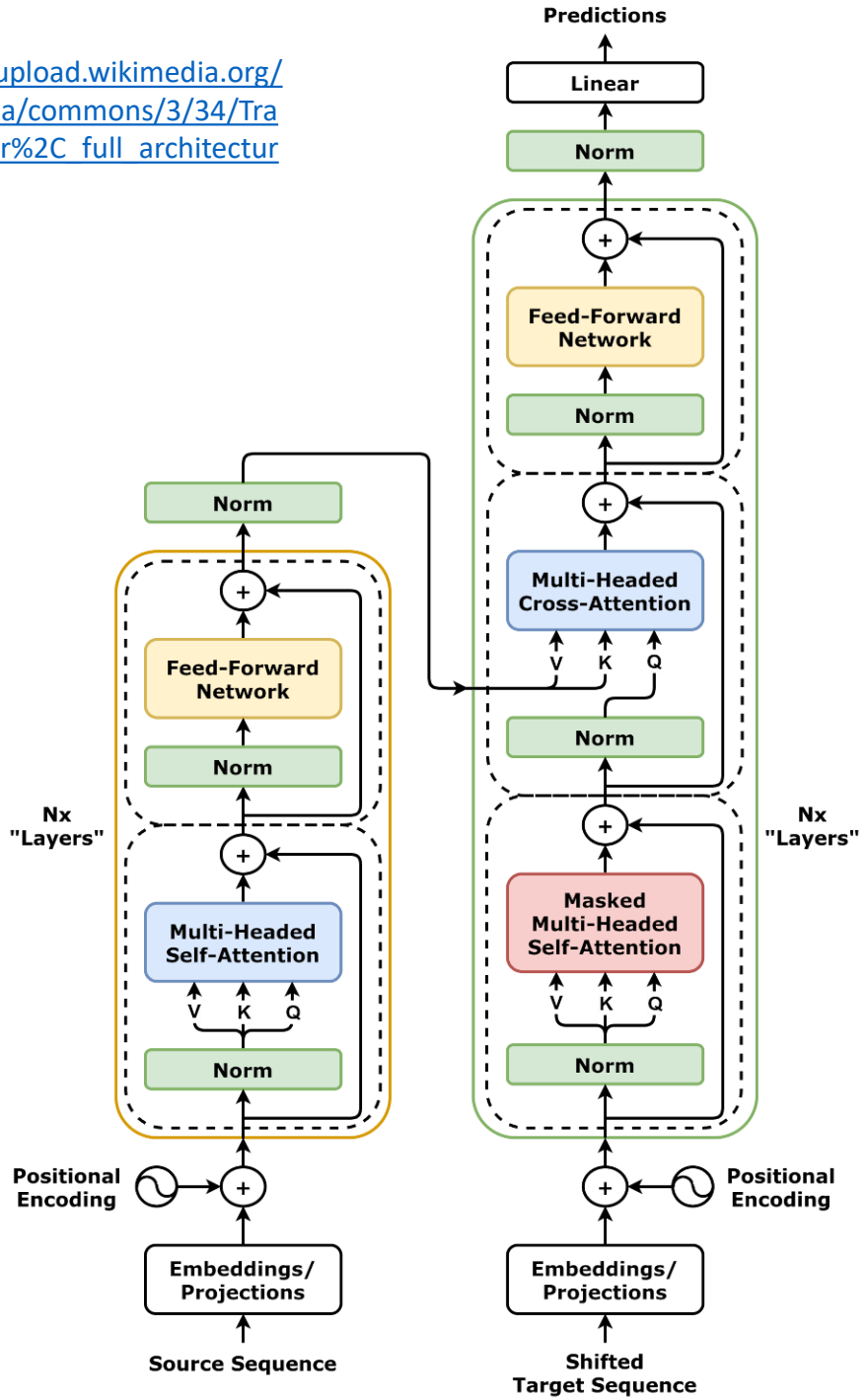
-支持并行输入（充分利用GPU的并行计算能力）



2. 大模型基础 - Transformer结构

注意力模块（Attention Module)，全连接神经网络模块 (Feed-Forward Network Module/Multilayer Perceptron)

图片
https://upload.wikimedia.org/wikipedia/commons/3/34/Transformer%2C_full_architecture.png



Transformer的提出

- “Attention Is All You Need“， 2017， Google

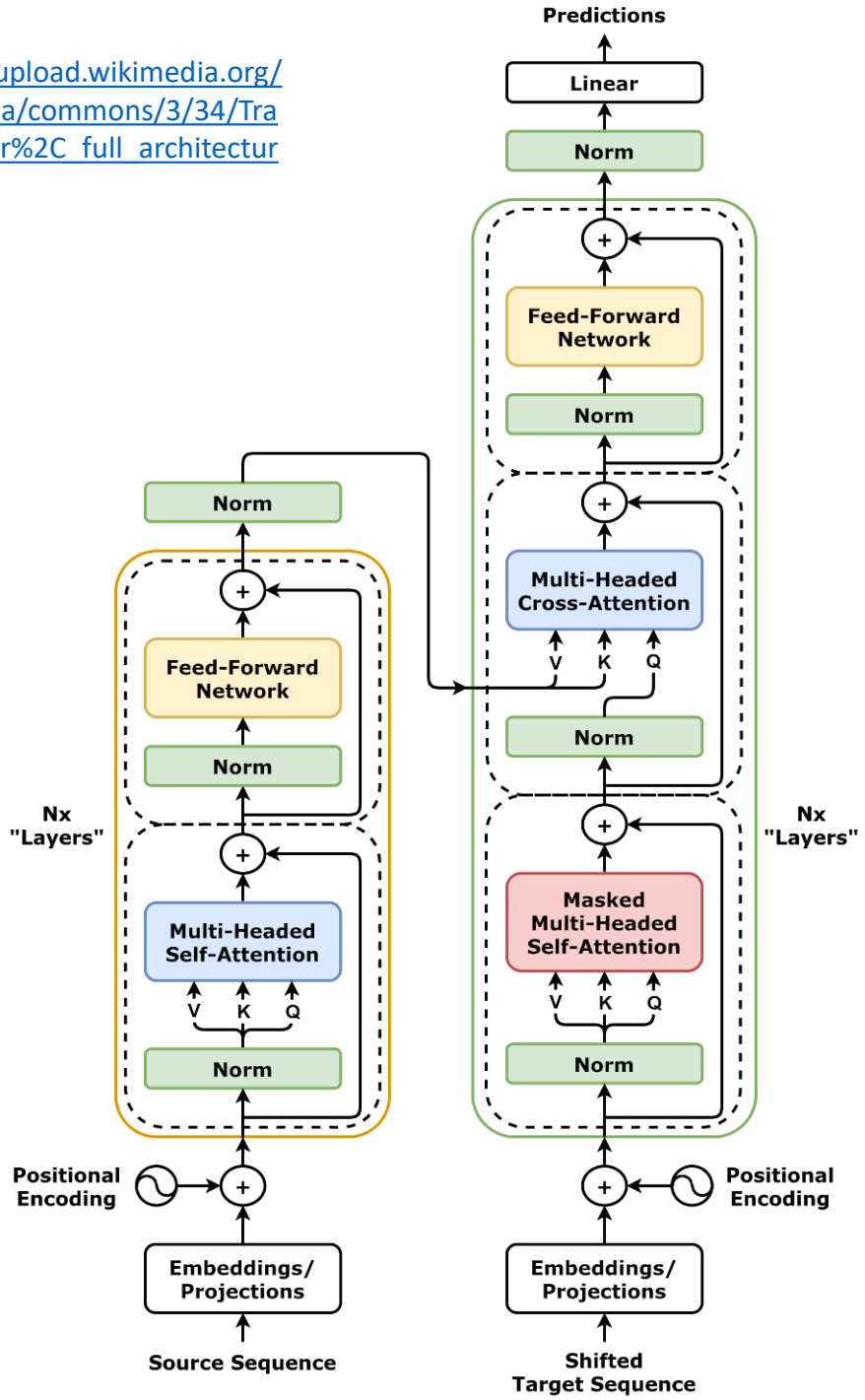
Transformer结构组成

- 编码器(Encoder), 解码器（Decoder）
  - **注意力模块（Attention Module)**
  - **全连接神经网络模块 （Feed-Forward Network Module/Multilayer Perceptron）**

语言模型的发展

- 早期模型
  - 编码器(Encoder) +解码器（Decoder)
    - Transformer, 2017
    - T5, 2019
  - 编码器（Encoder Only)
    - BERT, 2018
- *主流模型：解码器（Decoder Only）*

图片
https://upload.wikimedia.org/wikipedia/commons/3/34/Transformer%2C_full_architecture.png



Transformer的提出

- “Attention Is All You Need“， 2017， Google

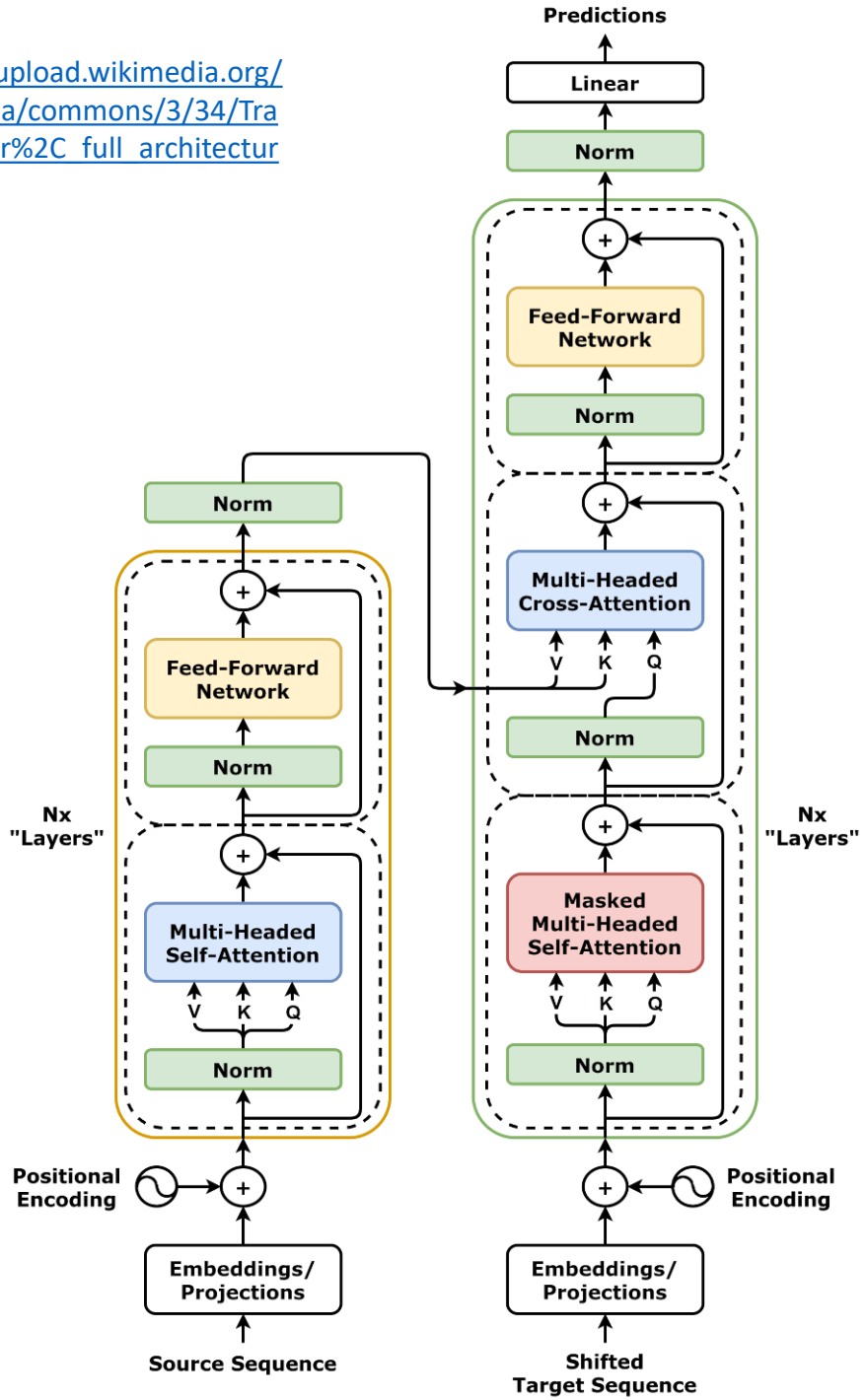
Transformer结构组成

- 编码器(Encoder), 解码器（Decoder）
  - **注意力模块（Attention Module)**
  - **全连接神经网络模块 （Feed-Forward Network Module/Multilayer Perceptron）**

语言模型的发展

- 早期模型
  - 编码器(Encoder) +解码器（Decoder)
    - Transformer, 2017
    - T5, 2019
  - 编码器（Encoder Only)
    - BERT, 2018
- ***主流模型：解码器（Decoder Only）***
  - *ChatGPT, Gemini, Llama, Qwen, ChatGLM, 文心, DeepSeek …*

图片
https://upload.wikimedia.org/wikipedia/commons/3/34/Transformer%2C_full_architecture.png

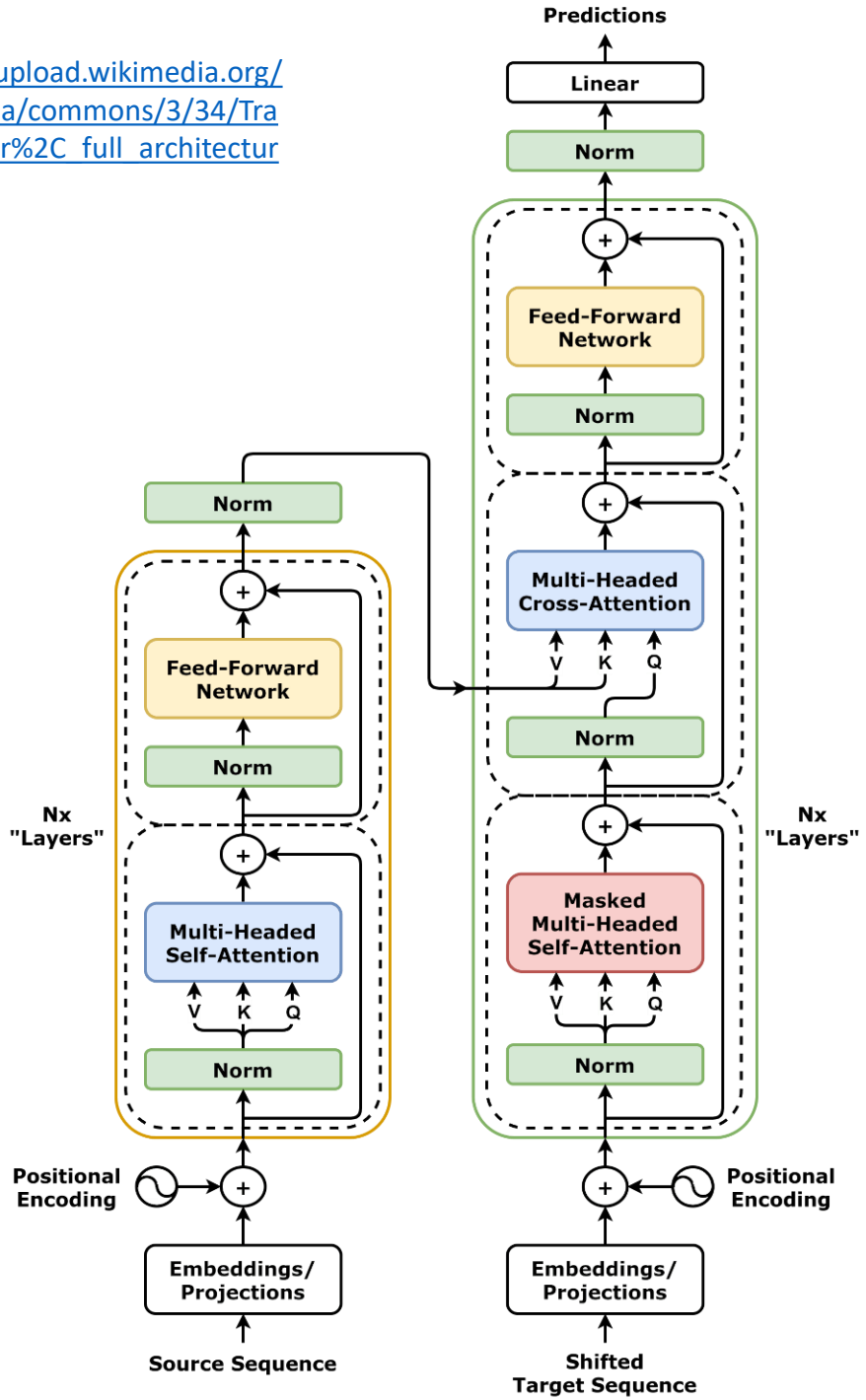


Transformer结构

- 编码器(Encoder), 解码器（Decoder）
  - **注意力模块（Attention Module)**
    - 主要作用：上下文理解
    - 核心概念：注意力机制
  - **前馈神经网络模块 （Feed-Forward Network Module）**
    - ~2/3参数
    - 主要作用：知识/记忆 存储

- **注意力模块（Attention Module)**
  - 主要作用：上下文理解
  - 核心组成：
    - **注意力层**
      - **Q**uery，**K**ey，**V**alue

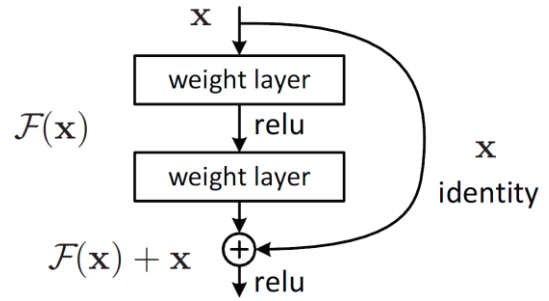
图片
https://upload.wikimedia.org/wikipedia/commons/3/34/Transformer%2C_full_architecture.png



Transformer结构

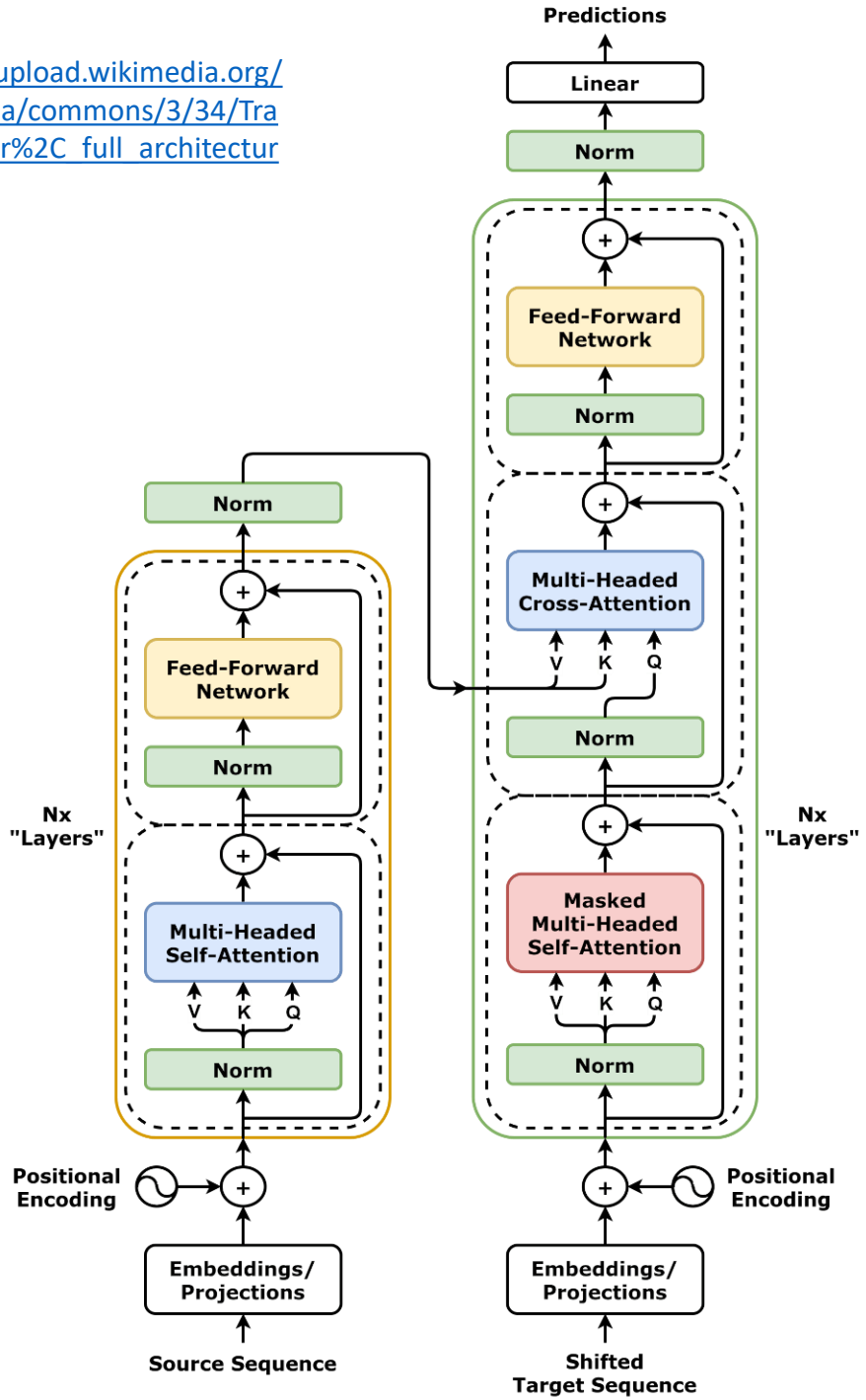
- 编码器(Encoder), 解码器（Decoder）
  - **注意力模块（Attention Module)**
    - 主要作用：上下文理解
    - 核心概念：注意力机制
  - **前馈神经网络模块 （Feed-Forward Network Module）**
    - ~2/3参数
    - 主要作用：知识/记忆 存储

- **注意力模块（Attention Module)**
  - 主要作用：上下文理解
  - 核心组成：
    - **注意力层**
      - **Q**uery，**K**ey，**V**alue
    - **残差连接 ($\oplus$)**
      - 避免深层网络梯度消失
        - Y= x + F(x)
        - dY/dx = 1 + dF/dx



图片：
https://paperswithcode.com/method/residual-connection

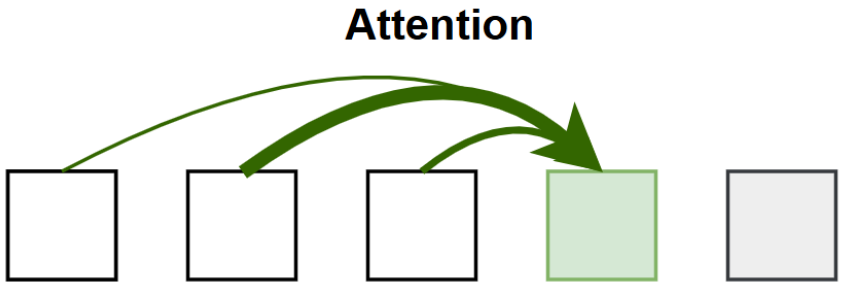
图片
https://upload.wikimedia.org/wikipedia/commons/3/34/Transformer%2C_full_architecture.png



Transformer结构

- 编码器(Encoder), 解码器（Decoder）
  - **注意力模块（Attention Module)**
    - 主要作用：上下文理解
    - 核心概念：注意力机制
  - **前馈神经网络模块 （Feed-Forward Network Module）**
    - ~2/3参数
    - 主要作用：知识/记忆 存储

- **注意力模块（Attention Module)**
  - 主要作用：上下文理解
  - 核心组成：
    - **注意力层**
      - **Q**uery，**K**ey，**V**alue
    - 残差连接 （⊕）
      - 避免深层网络梯度消失
    - **层正则化（Norm）**
      - 稳定&加速训练，增强泛化



3. Transformer中的注意力（Attention）机制

目的：让文本里的每一个词（Token）包含前文信息，模型更好理解文本内容

*词汇信息 --- 词向量表示 （E）

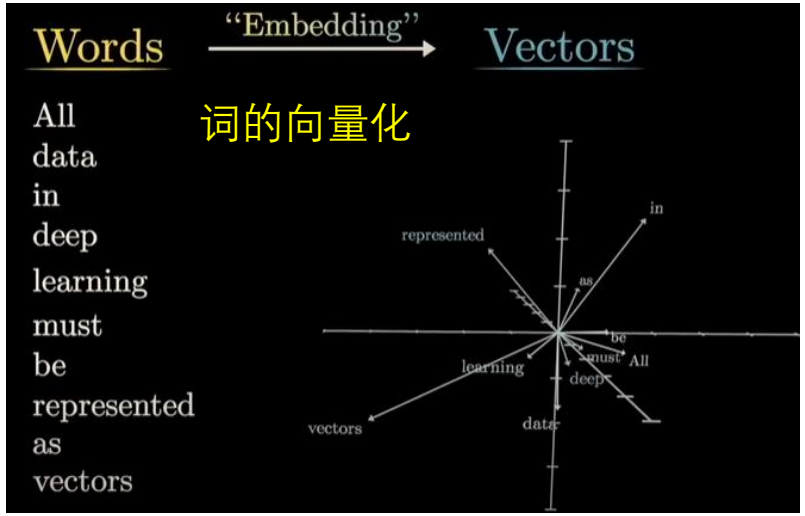
初始词向量 = E →**注意力层** → **包含前文信息的词向量=E’**

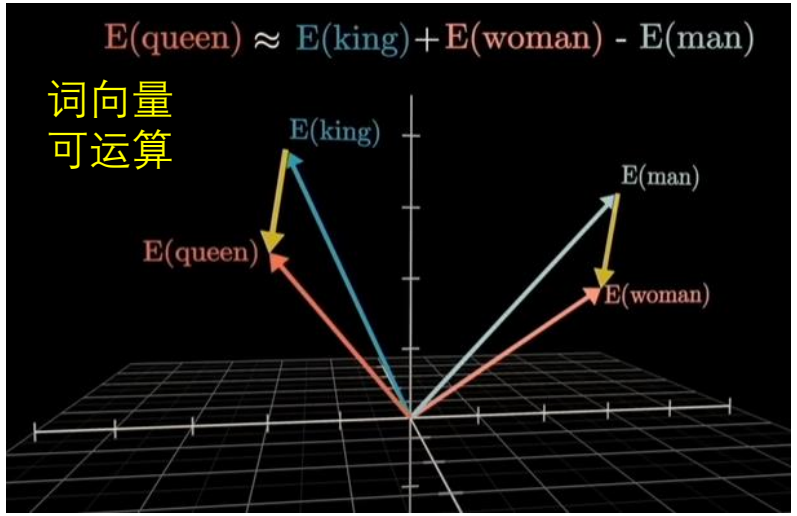
E’ = E + ΔE

ΔE = f (Q,K,V) = Attention (Q,K,V)

Q , K, V 分别是 查询 （Queries），键（Keys)和 值（Values）

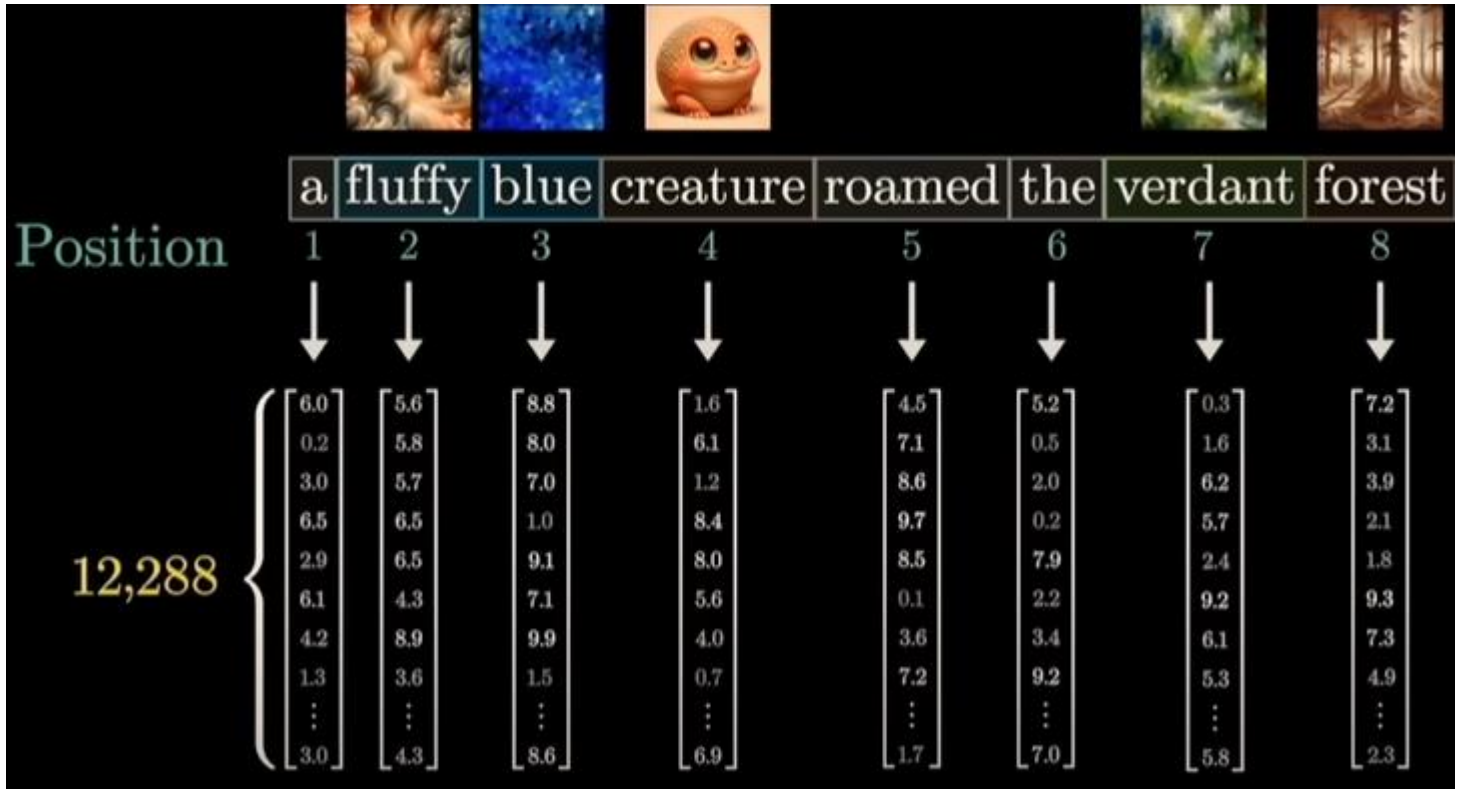
词（Token）的向量表示 – 词向量E





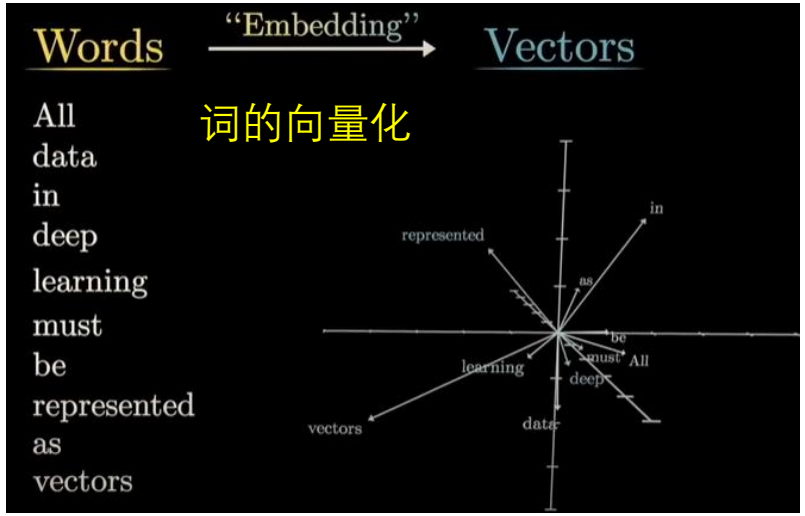
大模型理解语言的方式

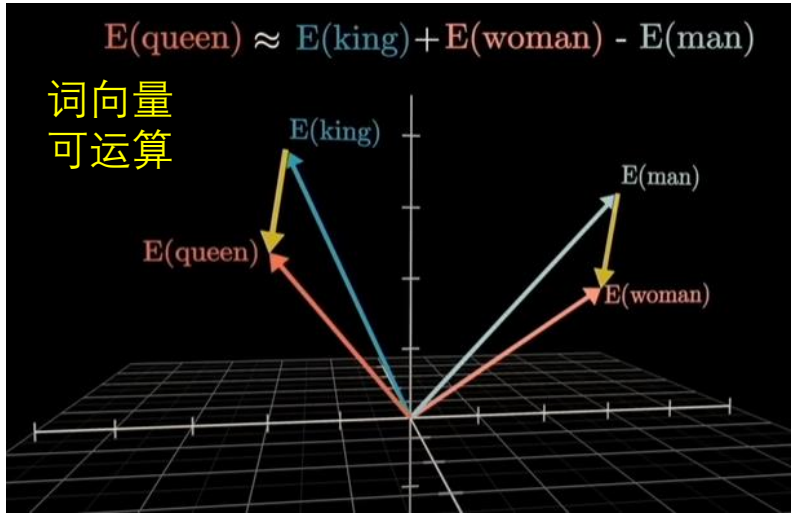
- 文本基础构建： 词（Token)
- 词的向量化 （Word Embedding）
  - 词向量 E
  - 包含词的含义
  - 可运算
- 词向量的不足之处
  - 仅反应当前词（单个词）的含义，缺乏上下文关联



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

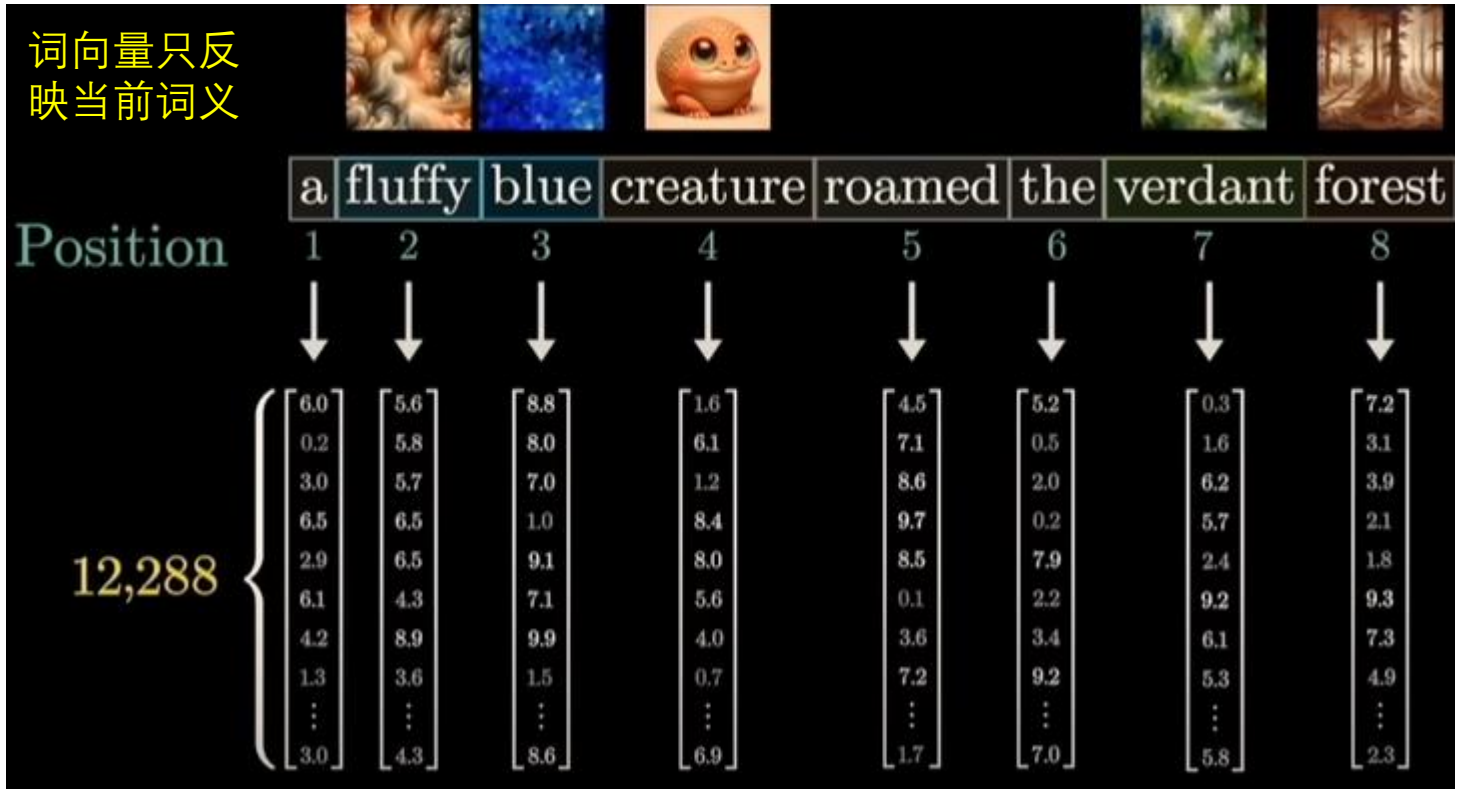
词（Token）的向量表示 – 词向量E





大模型理解语言的方式

- 文本基础构建： 词（Token)
- 词的向量化 （Word Embedding）
  - 词向量 E
  - 包含词的含义
  - 可运算
- **词向量的不足之处**
  - 仅反应当前词（单个词）的含义，缺乏上下文反应的信息

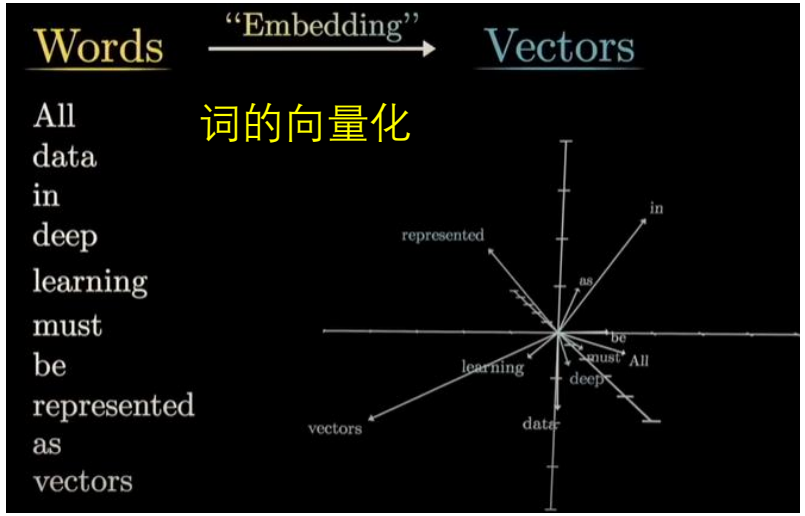


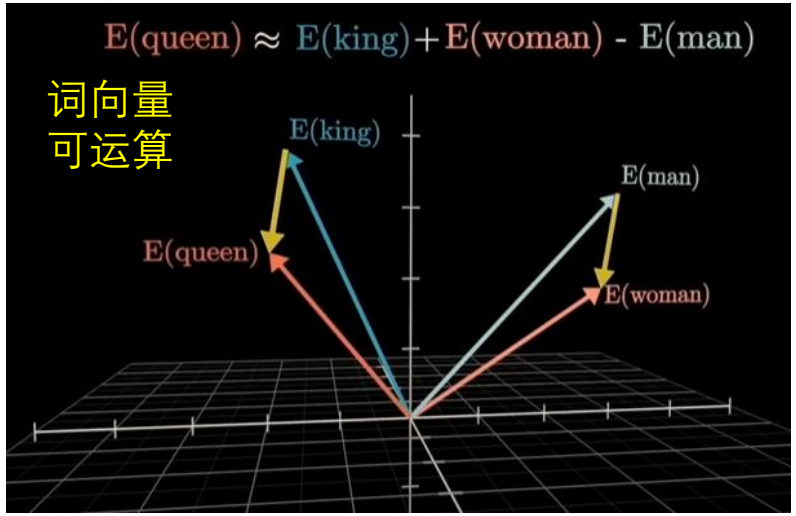
1."苹果公司发布了新款iPhone，股价应声上涨。"

2."她每天吃一个苹果，医生夸她饮食健康。"

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

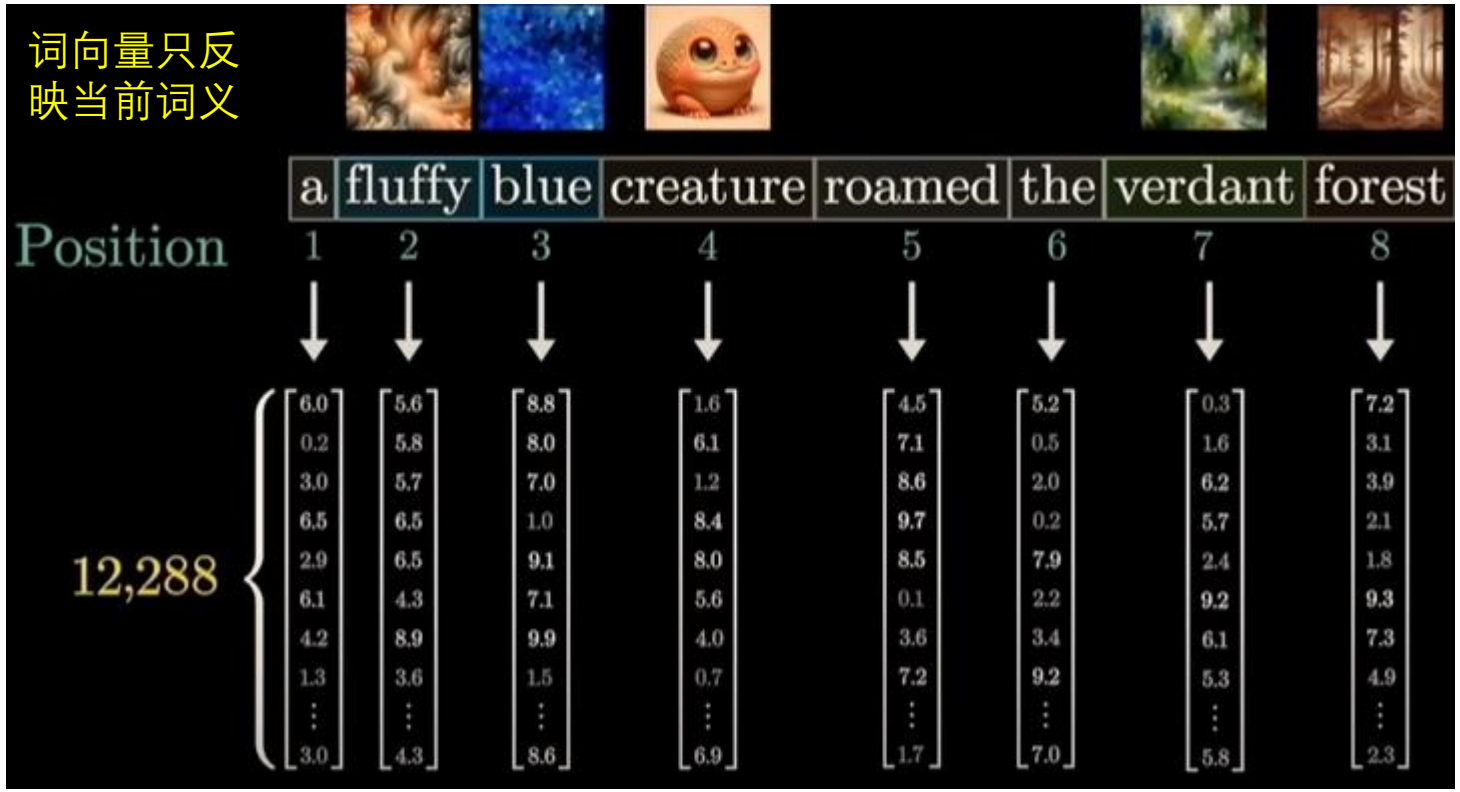
词（Token）的向量表示 – 词向量E





大模型理解语言的方式

- 文本基础构建： 词（Token)
- 词的向量化 （Word Embedding）
  - 词向量 E
  - 包含词的含义
  - 可运算
- **词向量的不足之处**
  - 仅反应当前词（单个词）的含义，缺乏上下文反应的信息



上下文对语义的影响

1."苹果公司发布了新款iPhone，股价应声上涨。"

2."她每天吃一个苹果，医生夸她饮食健康。"

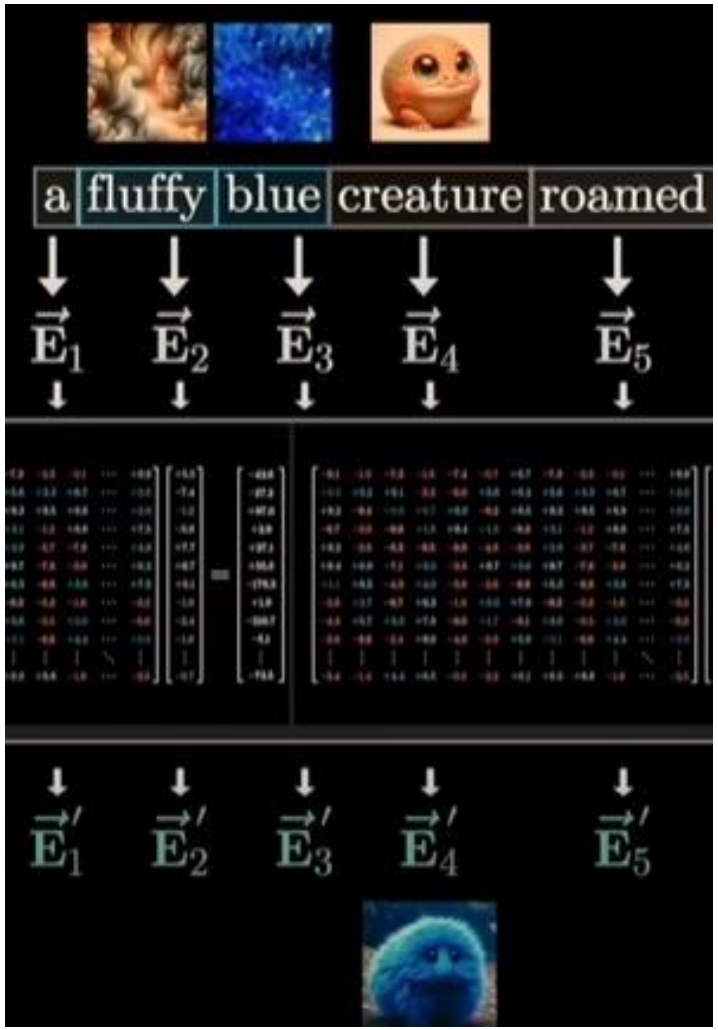
如何让每一个词向量包含前文语义?

让当前词汇"注意"到前文词汇的语义信息，根据注意力程度将前文词汇语义融入到当前词汇中 – 注意力机制

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

注意力机制 Attention： 让每一个词（向量）“注意”到前文词汇

目的：让文本里的每一个词向量包含前文有效信息，让模型更好理解结合上下文的语义内容



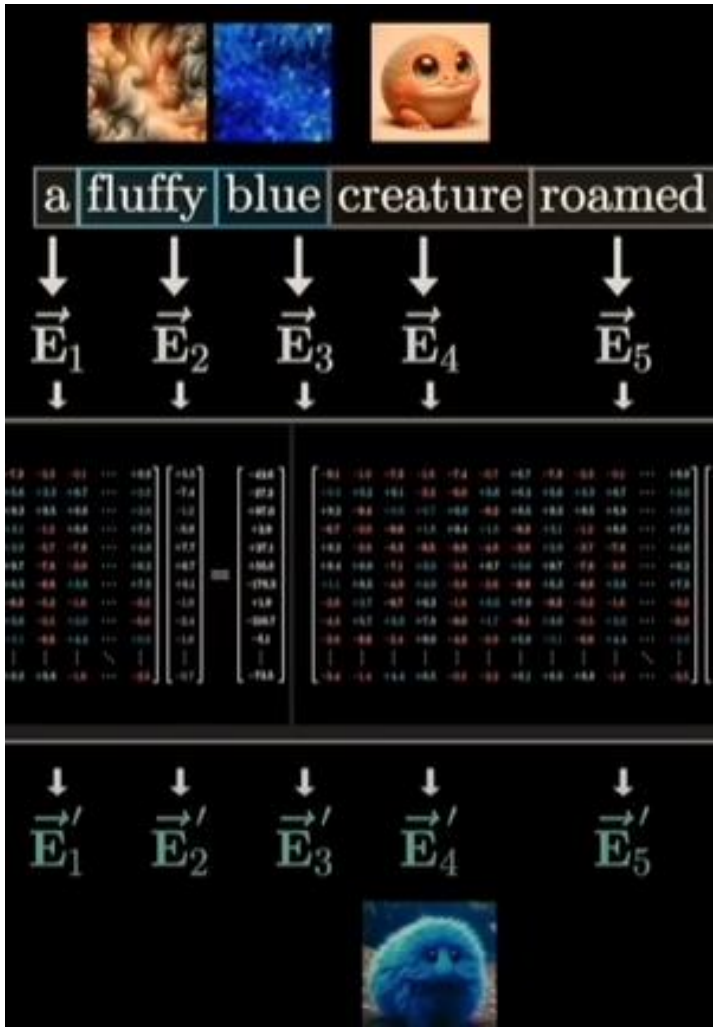
如何让文本里的每一个词向量包含前文有效信息？

- 原始词向量E （不包含前文信息，只包含词本身信息）
- 目标词向量E‘ （词本身信息 + 前文信息）
- 如何从E到E’？ - 注意力机制（Attention)
  - E’ = E + ΔE
  - ΔE = f (Q,K,V) = Attention (Q,K, V)

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

注意力机制 Attention： 让每一个词（向量）“注意”到前文词汇

目的：让文本里的每一个词向量包含前文有效信息，模型更好理解文本内容



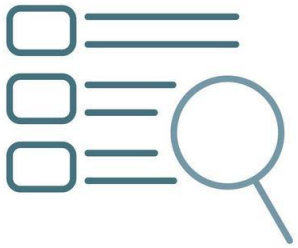
如何让文本里的每一个词向量包含前文有效信息？

- 原始词向量E （只包含词本身信息）
- 目标词向量E‘ （词本身信息 + 前文信息）
- 如何从E到E’？ - 注意力机制（Attention)
  - E’ = E + ΔE
  - ΔE = f (Q,K,V) = Attention (Q,K,V)

注意力机制的核心：**Q, K, V**

Q , K, V 分别是 查询 （Queries)， 键 （Keys) 和 值 （Values）

查询 （**Q**ueries)



键 （**K**eys)

值 （**V**alues)

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

注意力机制 Attention – Q, K ,V

目的：让文本里的每一个词向量包含前文有效信息，模型更好理解文本内容

如何让文本里的每一个词向量包含前文有效信息？

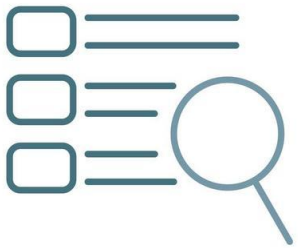
- 原始词向量E （不包含前文信息，只包含词本身信息）
- 目标词向量E‘ （词本身信息 + 前文信息）
- 如何从E到E’？ - 注意力机制 （Attention)
  - E’ = E + ΔE （ΔE = Attention (Q,K, V) ）

概念	概念类比（网络搜索）
Query	查询问题：“**适合孩子的北京景点**”
Key	网页标签：“亲子游”,“博物馆” ,“北京”,…
Value	网页内容

注意力机制的核心：**Q, K, V**

Q , K, V 分别是 查询 （Queries)，键 （Keys) 和 值 （Values）

查询 （**Q**ueries)



键 （**K**eys)

值 （**V**alues)

注意力机制 Attention – Q, K ,V

目的：让文本里的每一个词向量包含前文有效信息，模型更好理解文本内容

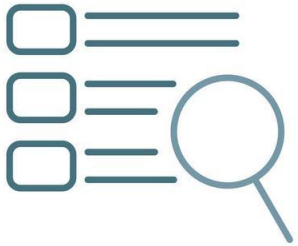
如何让文本里的每一个词向量包含前文有效信息？

- 原始词向量E （不包含前文信息，只包含词本身信息）
- 目标词向量E‘ （词本身信息(E) + **前文信息(ΔE)**）
- **前文信息(ΔE) ~ 注意力机制 （Attention)**
  - ΔE = Attention (**Q,K, V**) ）
  - E’ = E + ΔE

注意力机制的核心：**Q, K, V**

Q , K, V 分别是 查询 （Queries)，键 （Keys) 和 值 （Values）

查询 （**Q**ueries)

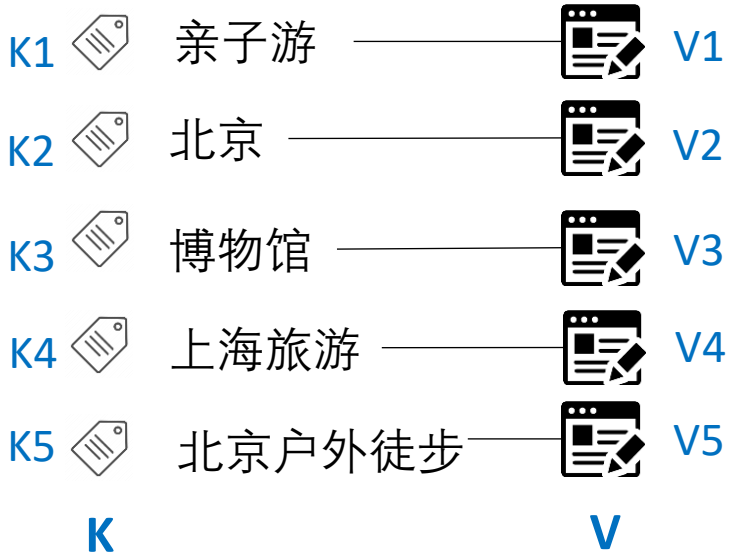


键 （**K**eys)

值 （**V**alues)

概念	概念类比（网络搜索）
Query	查询问题：“**适合孩子的北京景点**”
Key	网页标签：“亲子游”,“博物馆” ,“北京”,…
Value	网页内容

Q



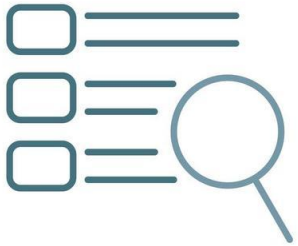
注意力机制 Attention – Q, K ,V

目的：让文本里的每一个词向量包含前文有效信息，模型更好理解文本内容

如何让文本里的每一个词向量包含前文有效信息？
- 原始词向量E （不包含前文信息，只包含词本身信息）
- 目标词向量E‘ （词本身信息(E) + **前文信息(ΔE)**）
- **前文信息(ΔE) ~ 注意力机制 （Attention)**
  - ΔE = Attention (**Q,K, V**) ）
  - E’ = E + ΔE

注意力机制的核心：**Q, K, V**
Q , K, V 分别是 查询 （Queries)，键 （Keys) 和 值 （Values）

查询 （**Q**ueries)



键 （**K**eys)

值 （**V**alues)

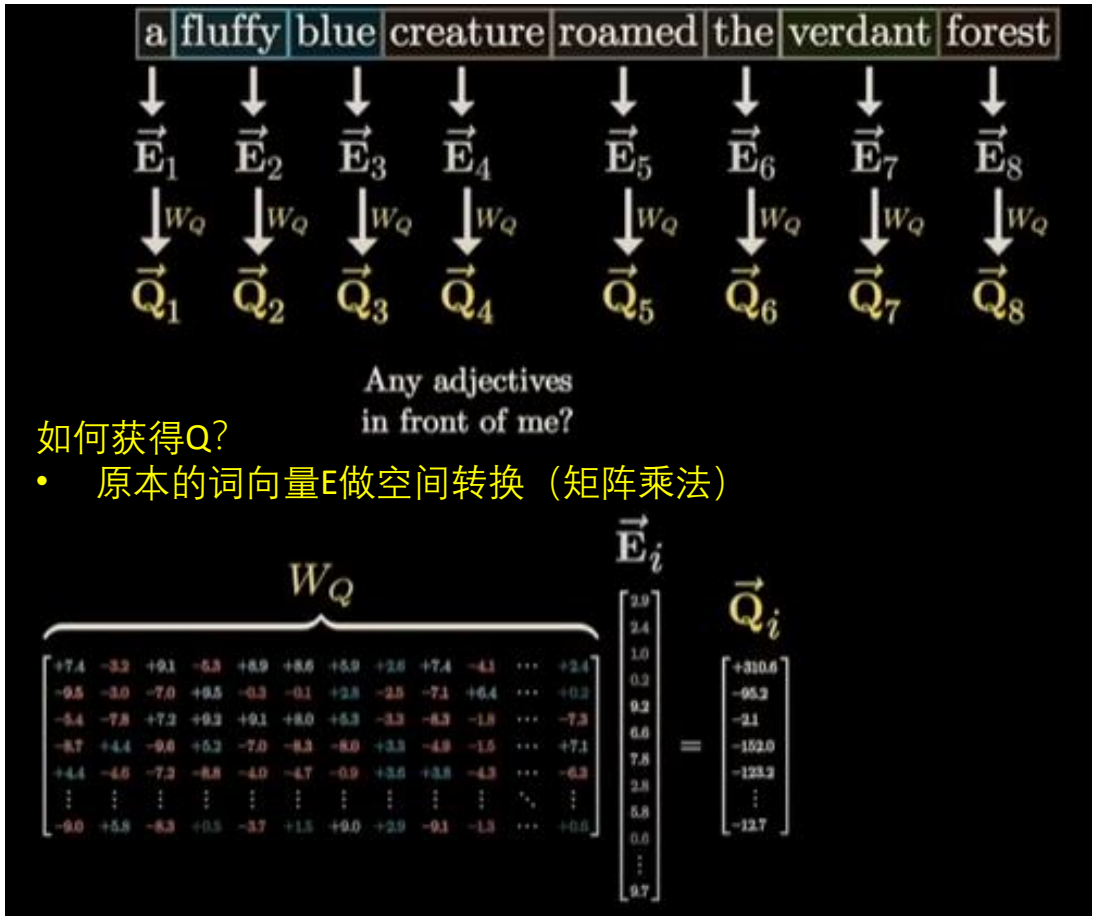
让当前词向量增加前文词汇信息
-检索前文词汇语义信息，让当前词汇分配“注意力”在前文词汇上
- 更相关的前文词汇--- 分配更多“注意力”
- 不相关的前文词汇--- 分配更少“注意力”

注意力程度 – Q，K 决定 (Q,K相关度)
前文词汇语义 - V

概念	概念类比（网络搜索）
Query	查询问题：“**适合孩子的北京景点**”
Key	网页标签：“亲子游”,“博物馆” ,“北京”,…
Value	网页内容

网页检索相关信息：
-分配不同的“注意力”在检索的网页上
- 相关网页--- 分配更多“注意力”
- 不相关网页 --- 分配更少“注意力”

注意力机制 Attention – Q, K ,V



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

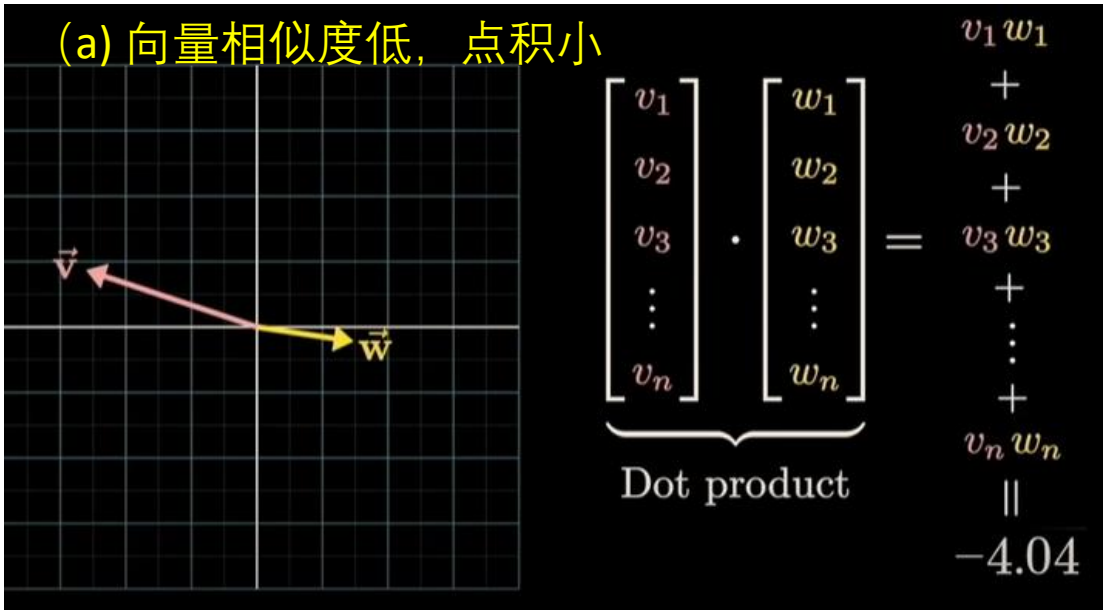
如何让*当前词(E)*获得**前文所有词汇的有效信息 (ΔE)**?

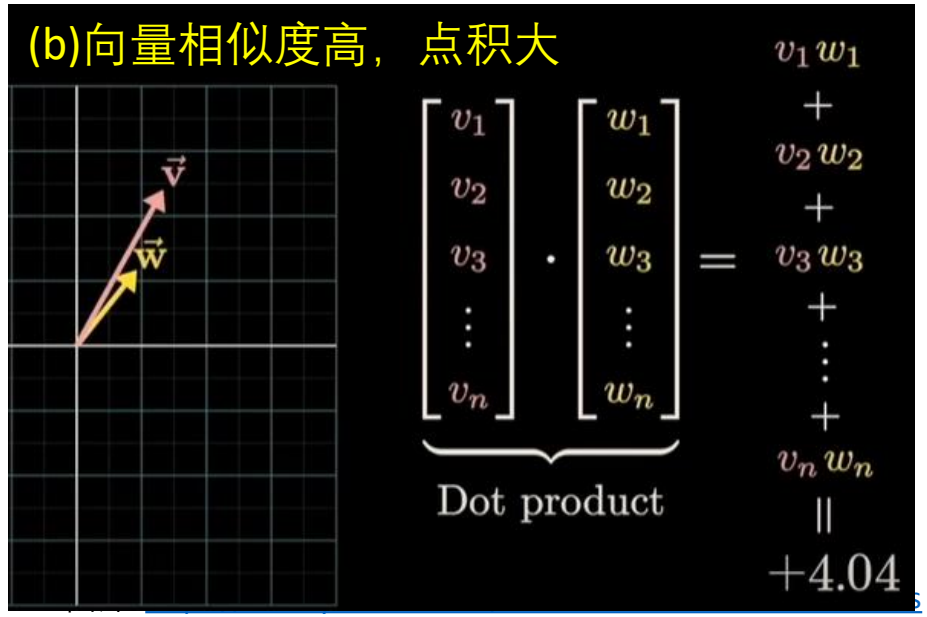
1. **询问**：对前文每一个词语义信息进行**查询/询问 (Q)**
2. **答案**：获得前文每一个词得到查询**答案 (K)**
   1. 如果被查询的前文词汇的答案K和问题Q相关度高（答案匹配问题）→ 此词汇对当前词汇重要性高，应该被分配更多的**“注意力”** - **注意力系数 = f(Q,K)**
3. **根据注意力融合**：根据每一个前文词汇分配的“注意力”，将**前文词汇语义（V）融合**到当前词汇中
   1. 每一个前文词汇的有效信息= 注意力系数f(Q,K) * 语义V
   2. 将每一个前文词汇的有效信息 融入到 当前词汇（E）中

$$E' = E + \Delta E$$

$$\Delta E = \sum (\text{注意力系数} * V) = \sum (f(Q,K) * V)$$

注意力机制 Attention – Q, K ,V





如何让*当前词(E)*获得**前文所有词汇的有效信息 (ΔE)**?

1. **询问**：对前文每一个词语义信息进行**查询/询问 (Q)**
2. **答案**：获得前文每一个词得到查询**答案 (K)**
   1. 如果被查询的前文词汇的答案K和问题Q相关度高（答案匹配问题）→ 此词汇对当前词汇重要性高，应该被分配更多的**“注意力”** - **注意力系数 = f(Q,K)**
3. **根据注意力融合**：根据每一个前文词汇分配的“注意力”，将**前文词汇语义（V）融合**到当前词汇中
   1. 每一个前文词汇的有效信息= 注意力系数f(Q,K) * 语义V
   2. 将每一个前文词汇的有效信息 融入到 当前词汇（E）中

$$E' = E + \Delta E$$

$$\Delta E = \sum(\text{注意力系数} * V) = \sum(f(Q,K) * V)$$

Q和K是如何反应前文词汇对当前词汇的重要性 （如何计算注意力系数）？

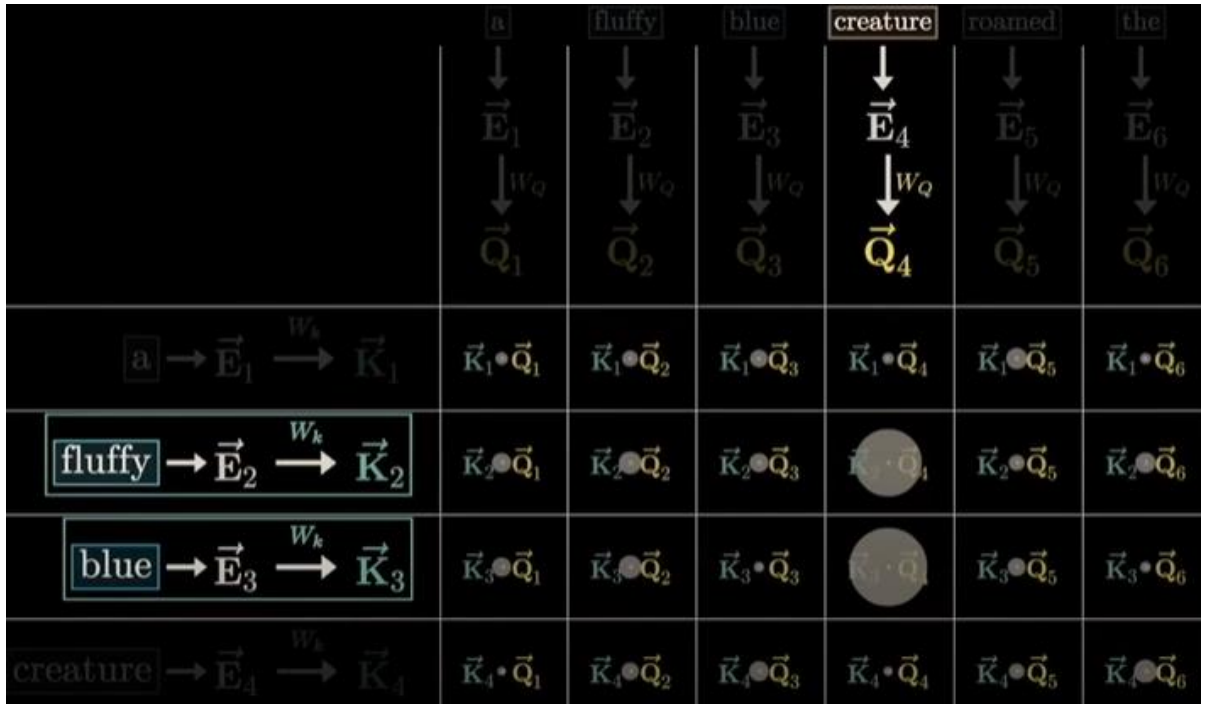
- **注意力系数 - 通过向量点积KQ的大小**
  - 反映了***KQ向量的相似度, 如果* KQ相似度高**
    - 当前词向量答案K和问题Q匹配度高，当前词向量对于此问题关联度高，注意力高，注意力系数高
- 注意力系数 = f (KQ)

注意力机制 Attention – Q, K ,V

Q和K是如何反应前文词汇对当前词汇的重要性 （如何计算注意力系数）？

- **注意力系数 - 通过向量点积KQ的大小**
  - 反映了***KQ向量的相似度, 如果*** **KQ相似度高**
    - 当前词向量答案K和问题Q匹配度高， 当前词向量对于此问题关联度高，注意力高，注意力系数高
- 注意力系数 = f (KQ) 为了统一衡量，方便计算，归一化KQ点积的大小, 通过Softmax函数 （0 ~1）

$$\text{Attention Weights} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)$$



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

如何让***当前词(E)*** 获得**前文所有词汇的有效信息 (ΔE)**?

1. **询问**：对前文每一个词语义信息进行**查询/询问 (Q)**
2. **答案**：获得前文每一个词得到查询**答案 (K)**
   1. 如果被查询的前文词汇的答案K和问题Q相关度高（答案匹配问题）→ 此词汇对当前词汇重要性高，应该被分配更多的**“注意力”** - **注意力系数 = f(Q,K)**
3. **根据注意力融合**：根据每一个前文词汇分配的“注意力”，将**前文词汇语义（V）融合**到当前词汇中
   1. 每一个前文词汇的有效信息= 注意力系数f(Q,K) * 语义V
   2. 将每一个前文词汇的有效信息 融入到 当前词汇（E）中

$$E' = E + \Delta E$$

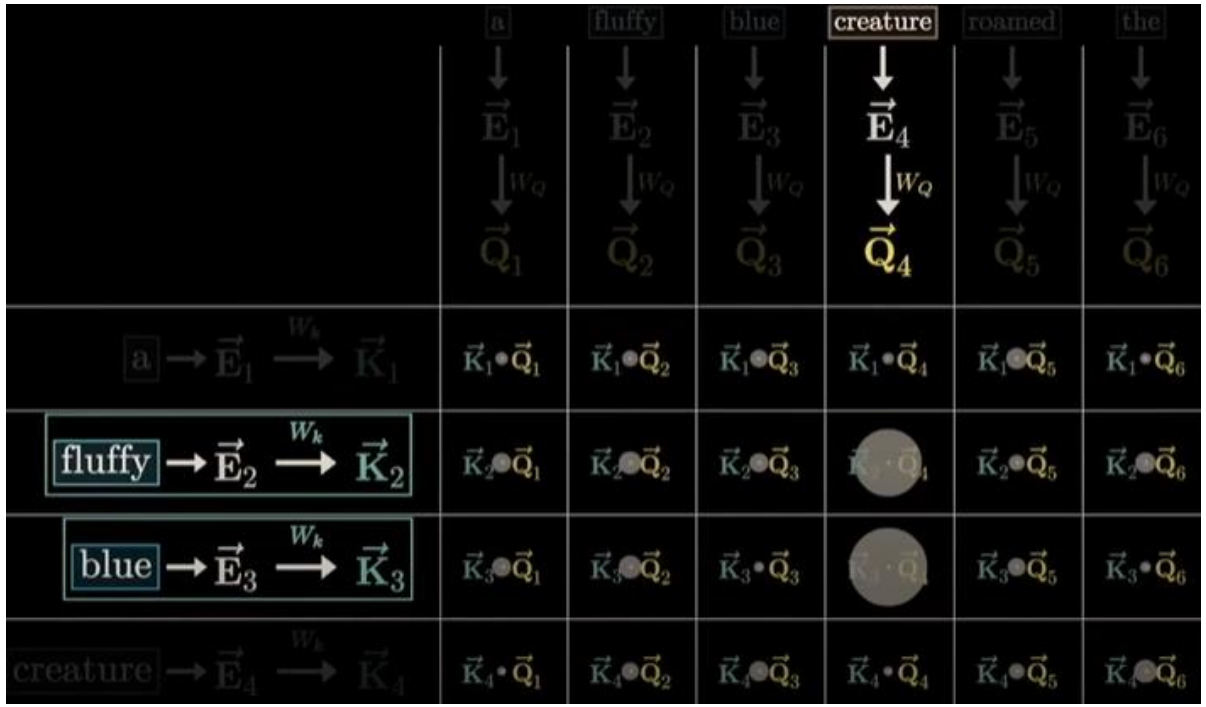
$$\Delta E = \sum(\text{注意力系数} * V) = \sum(f(Q,K) * V)$$

注意力机制 Attention – Q, K ,V

Q和K是如何反应前文词汇对当前词汇的重要性 （如何计算注意力系数）？

- **注意力系数 - 通过向量点积KQ的大小**
  - 反映了***KQ向量的相似度, 如果* KQ相似度高**
    - 当前词向量答案K和问题Q匹配度高， 当前词向量对于此问题关联度高，注意力高，注意力系数高
- 注意力系数 = f (KQ) 为了统一衡量，方便计算，归一化KQ点积的大小, 通过Softmax函数 （0 ~1）

$$\text{Attention Weights} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)$$



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

如何让***当前词(E)*** 获得**前文所有词汇的有效信息 (ΔE)**?

1. **询问**：对前文每一个词语义信息进行**查询/询问 (Q)**
2. **答案**：获得前文每一个词得到查询**答案 (K)**
   1. 如果被查询的前文词汇的答案K和问题Q相关度高（答案匹配问题）→ 此词汇对当前词汇重要性高，应该被分配更多的**“注意力”** - **注意力系数 = f(Q,K)**
3. **根据注意力融合**：根据每一个前文词汇分配的“注意力”，将**前文词汇语义（V）融合**到当前词汇中
   1. 每一个前文词汇的有效信息= 注意力系数f(Q,K) * 语义V
   2. 将每一个前文词汇的有效信息 融入到 当前词汇（E）中

$$E' = E + \Delta E$$

$$\Delta E = \sum (\text{注意力系数} * V) = \sum (f(Q,K) * V)$$

V- **Values 值**

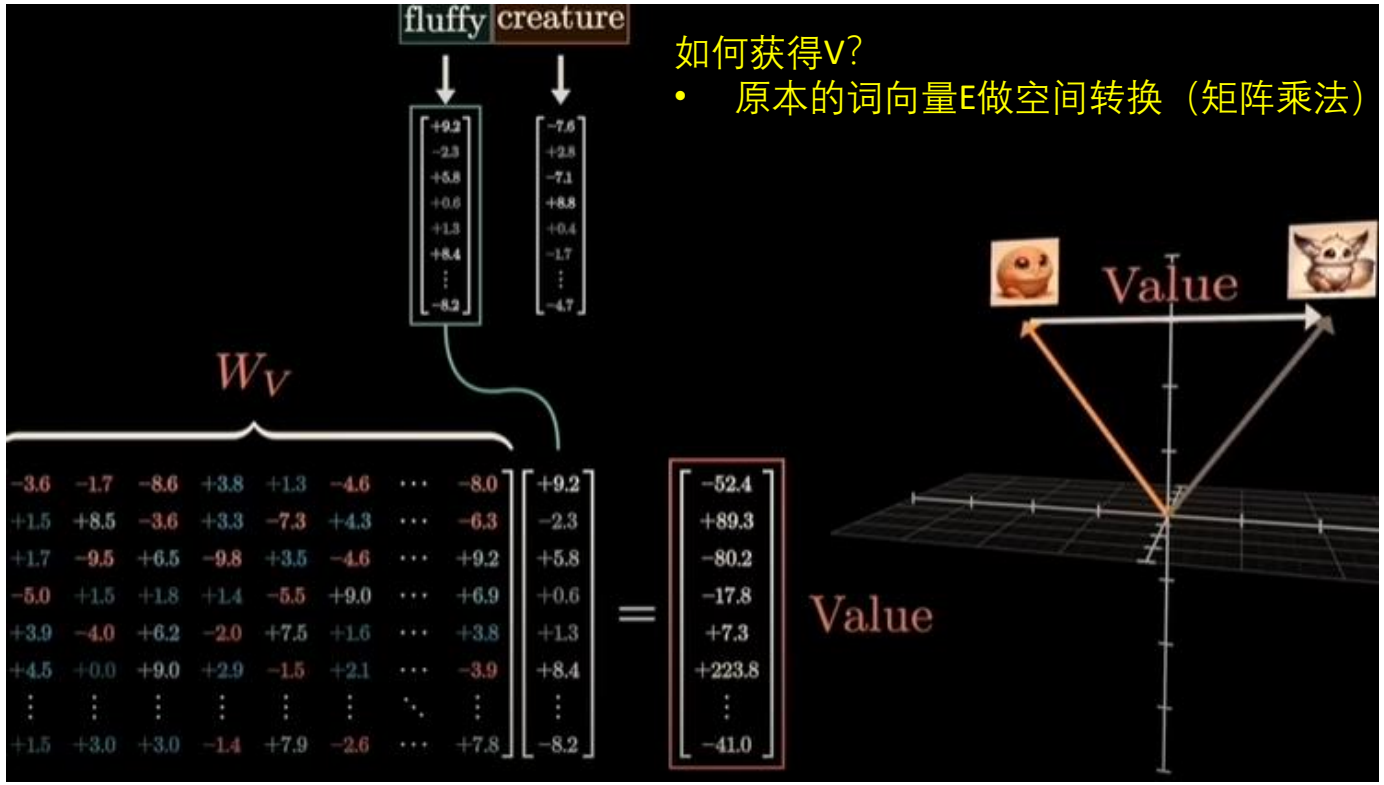
V = 当前词汇的部分语义 （单头Single Head）

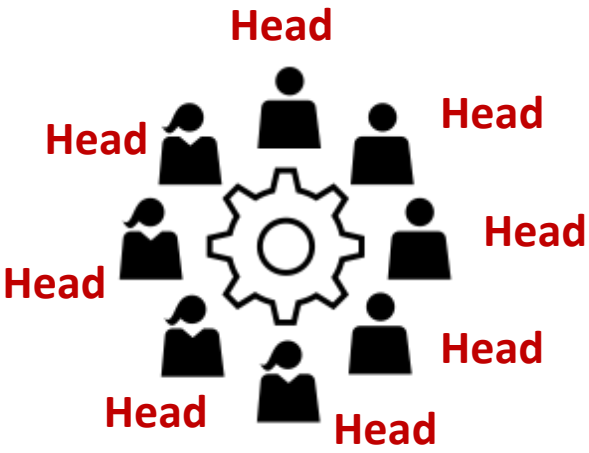
E = 当前词汇的全部语义

V和E的区别?

- V包含了E的部分语义，在**多头**机制下，每一个头 (Head) 的V都包含了E的一部分语义
  - 假设E的维度是12288， 一共有16头，那么每头（Head)中 V的维度 = 12288/16 = 768维

多头设计的作用：并行计算效率增加；提高稳定性；多个角度（头）捕捉语义关系





图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

V- Values 值

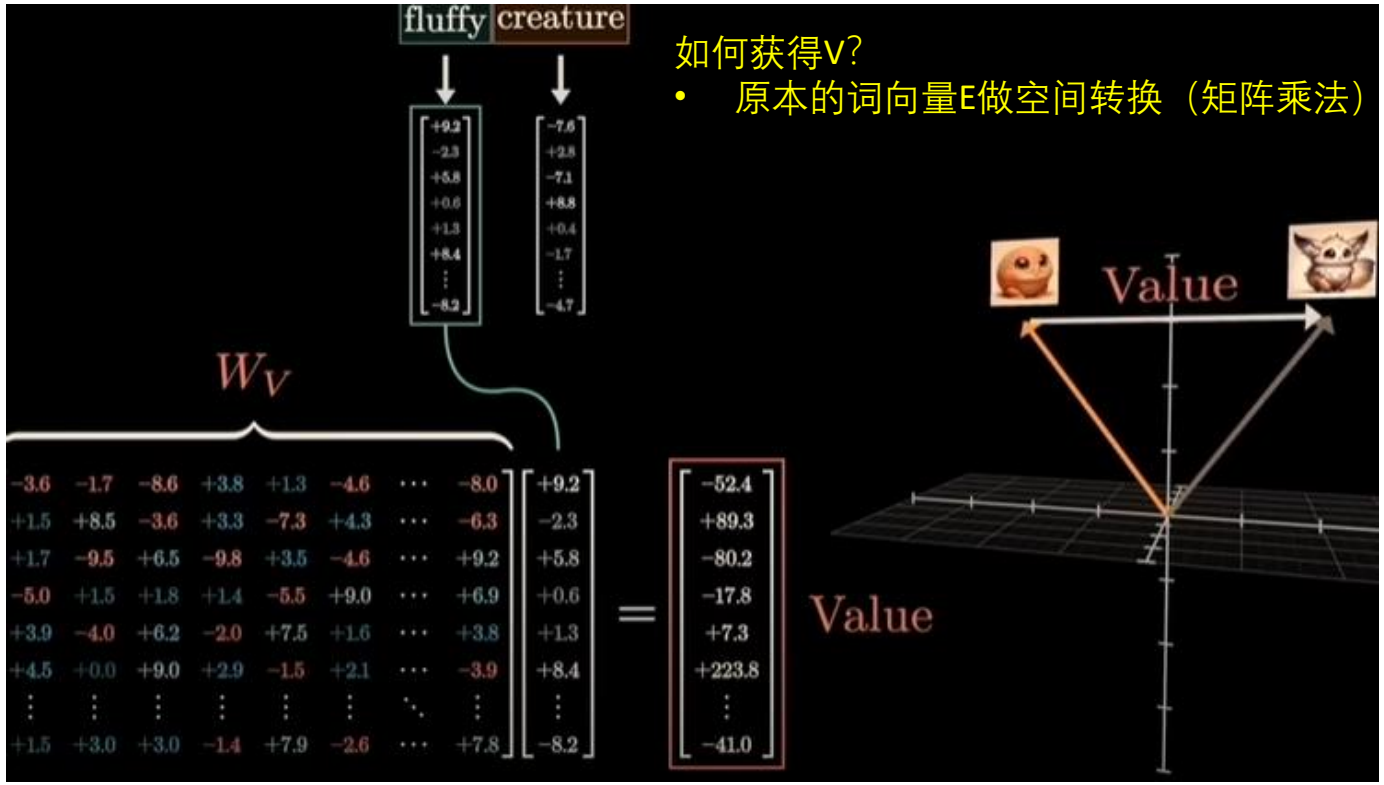
V = 当前词汇的部分语义 （单头Single Head）

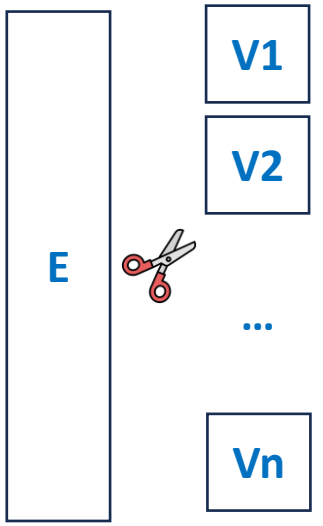
E = 当前词汇的全部语义

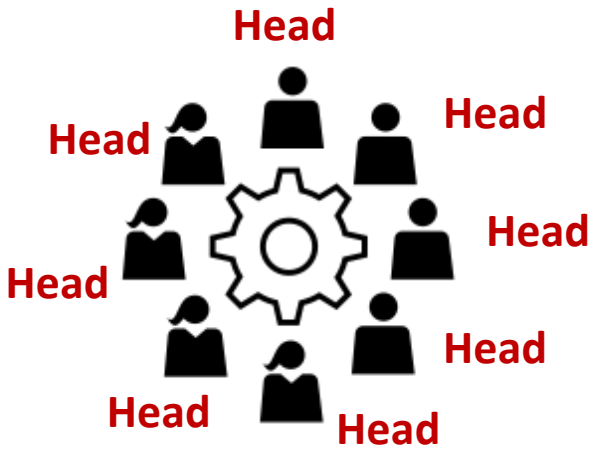
V和E的区别?

- V包含了E的部分语义，在**多头**机制下，每一个头 (Head) 的V都包含了E的一部分语义
  - 假设E的维度是12288， 一共有16头，那么每头（Head)中 V的维度 = 12288/16 = 768维

多头设计的作用：并行计算效率增加；提高稳定性；多个角度（头）捕捉语义关系







多头(Multi-Head) 机制

- 将语义理解任务分配给多个专家（头）
- 每个专家（头）负责一部分的语义理解 V
- 多个专家组同时进行工作， 最后汇总所有专家（头）的结果

图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

V- Values 值

V = 当前词汇的部分语义 （单头Single Head）

E = 当前词汇的全部语义

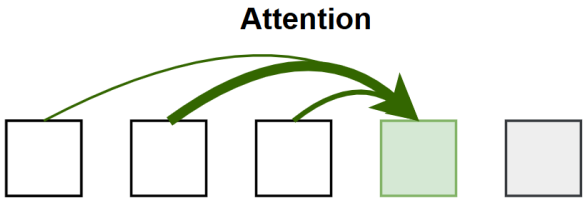
V和E的区别?

E' = E + ΔE

ΔE = ?

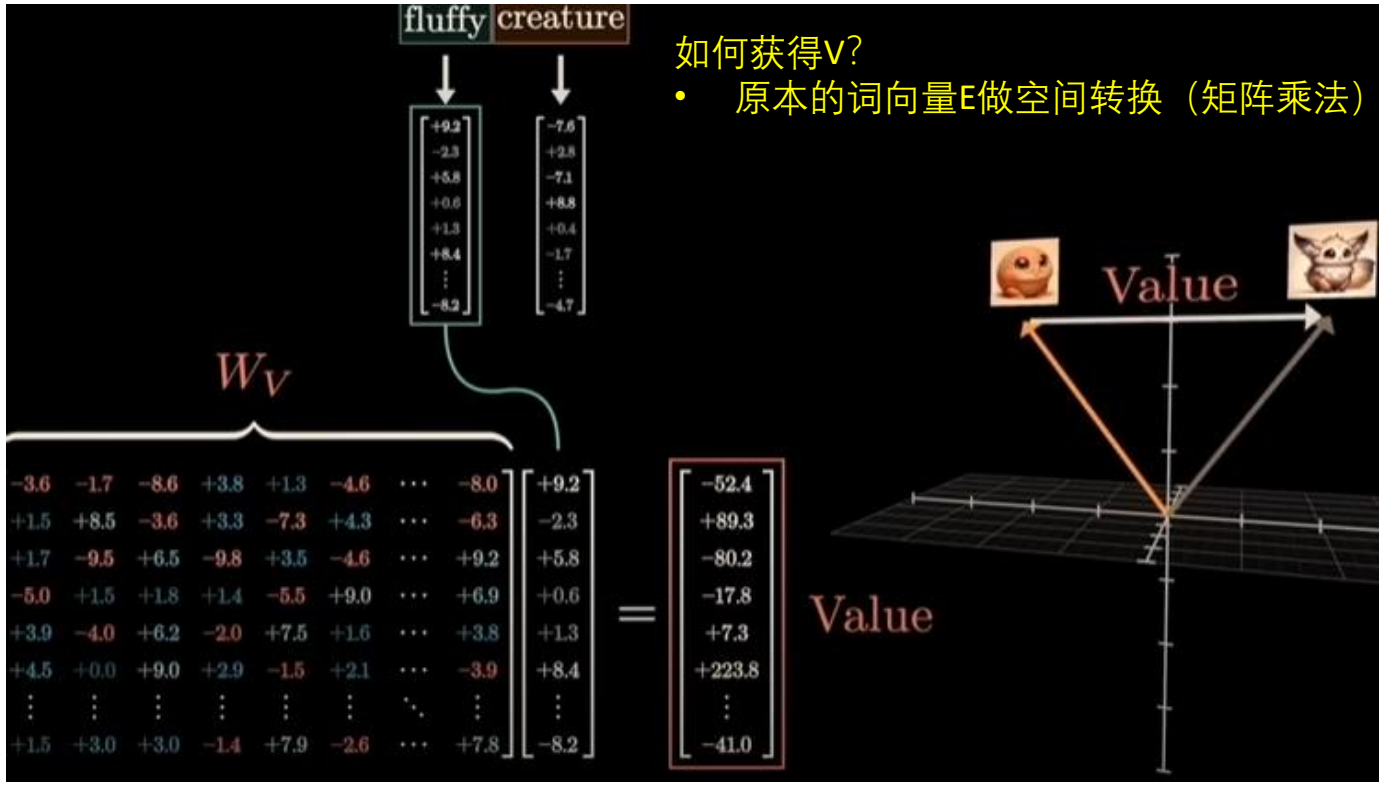
1. Attention Weights = f(QK)

2. Attention = ΔE = Attention Weights * V

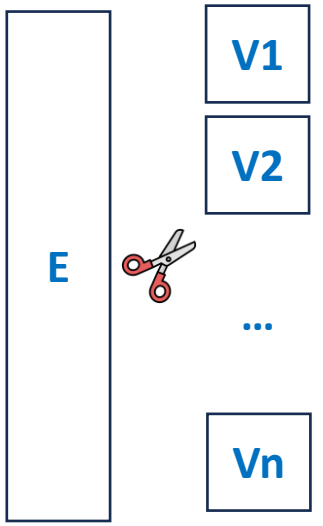


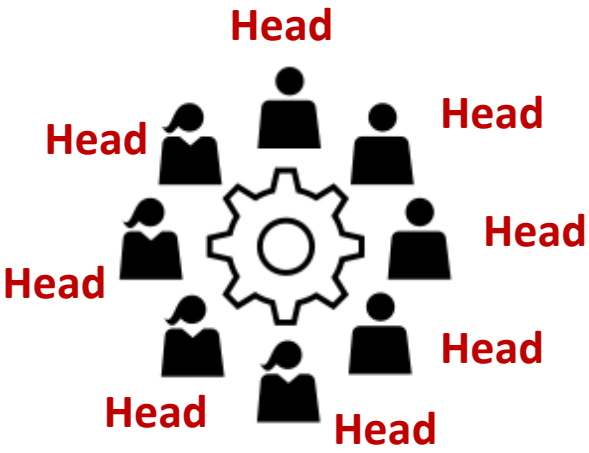
- V包含了E的部分语义，在**多头**机制下，每一个头 (Head) 的V都包含了E的一部分语义
  - 假设E的维度是12288， 一共有16头，那么每头（Head)中 V的维度 = 12288/16 = 768维

多头设计的作用：并行计算效率增加；提高稳定性；多个角度（头）捕捉语义关系



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s





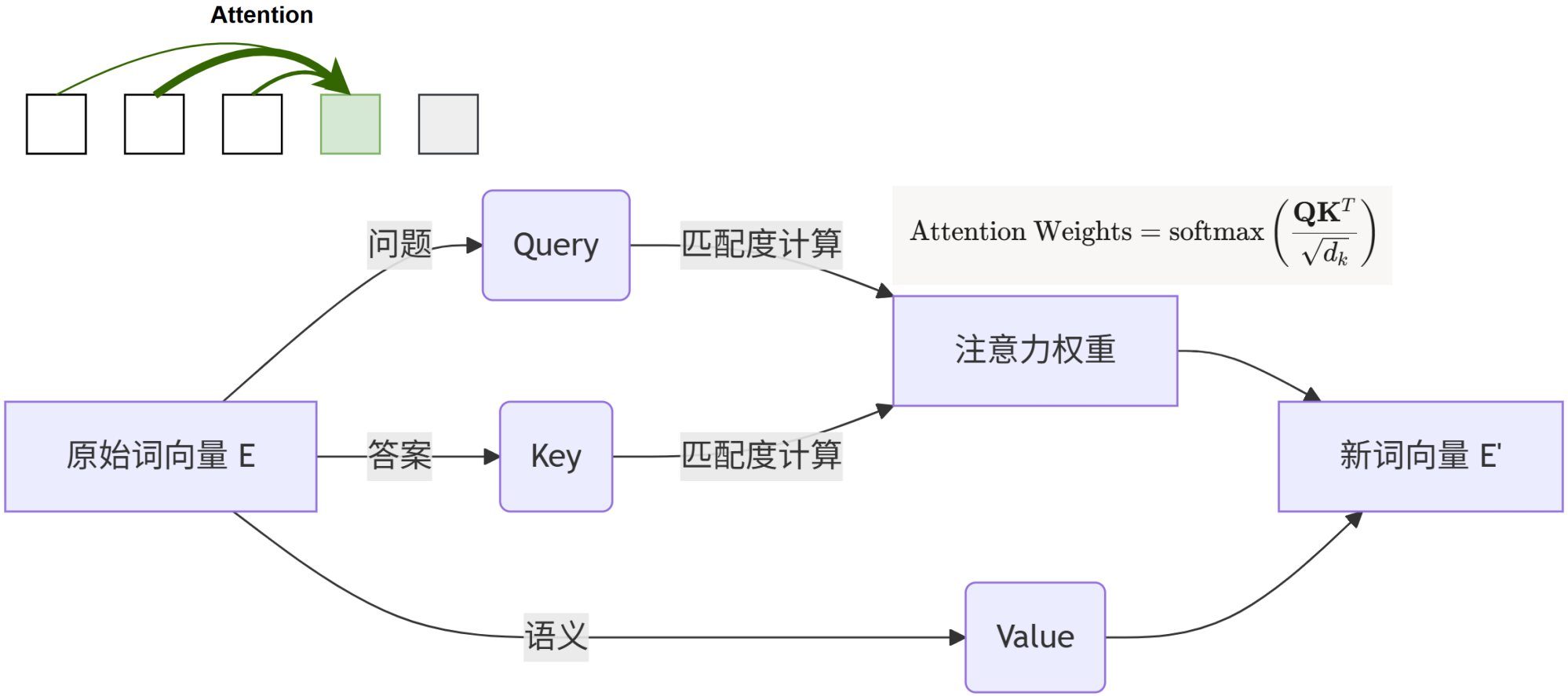
多头(Multi-Head) 机制

- 将语义理解任务分配给多个专家（头）
- 每个专家（头）负责一部分的语义理解 V
- 多个专家组同时进行工作， 最后汇总所有专家（头）的结果

注意力（Attention）机制总结

E ΔE E’ = E + ΔE

最终目标： 增强语言模型对文本的理解能力 ←→ 如何让当前(每一个)词向量获得前文所有词汇的有效信息？

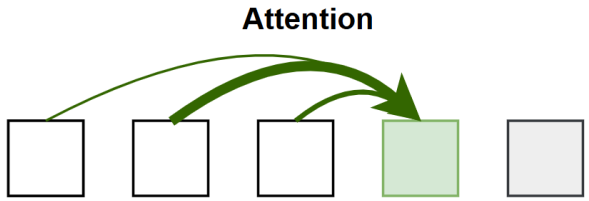


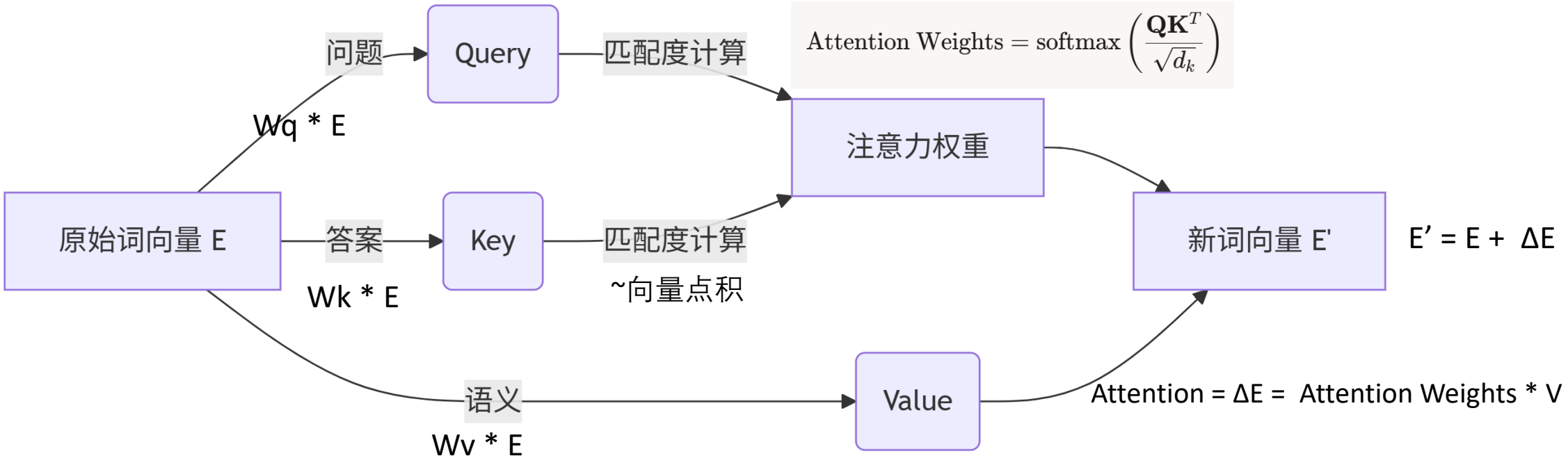
图片： https://github.com/dvgodoy/PyTorchStepByStep

注意力（Attention）机制总结

E ΔE E’ = E + ΔE

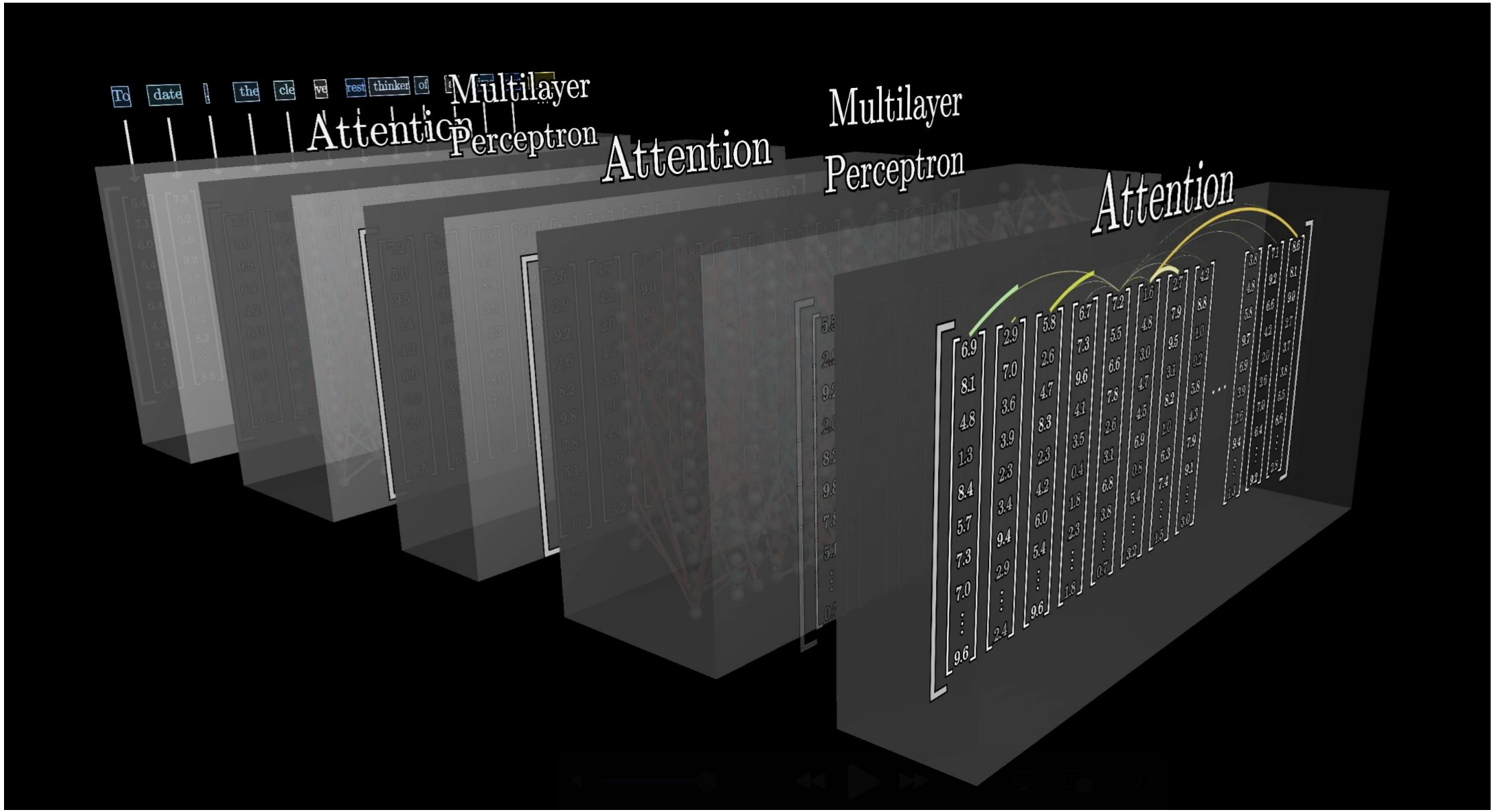
最终目标： 增强语言模型对文本的理解能力 ←→ 如何让当前(每一个)词向量获得前文所有词汇的有效信息?





图片： https://github.com/dvgodoy/PyTorchStepByStep

Transformer结构 – 多个 注意力 （Attention) 和全连接神经网络 (MLP) 模块



图片: https://www.youtube.com/watch?v=KJtZARuO3JY&t=2989s

结束

主讲：东东东

B站视频链接:
https://www.bilibili.com/video/BV1BZTszKEbv/?share_source=copy_web&vd_source=f23fdab1cf57871b257305ebe143b9c2