Transformer

Transformer Paper: Attention is All You Need

Google 2017年发布的论文： Attention is all you need.

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

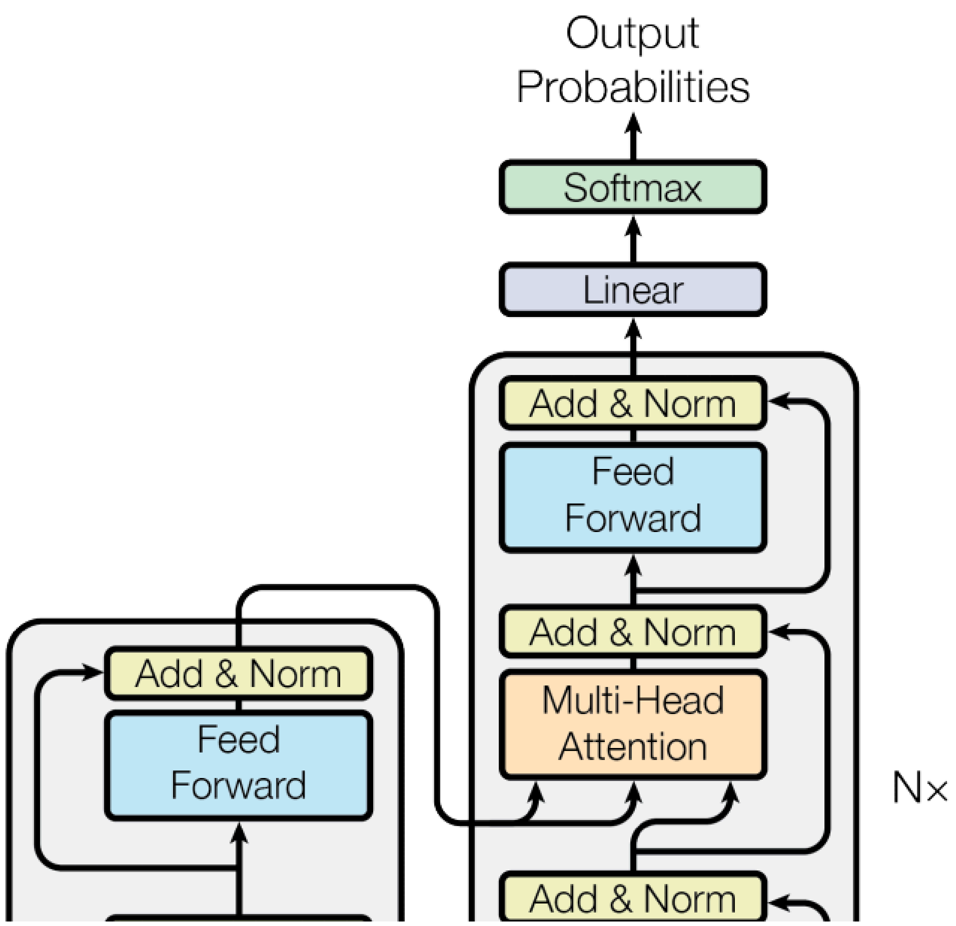
Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Transformer Model Architecture



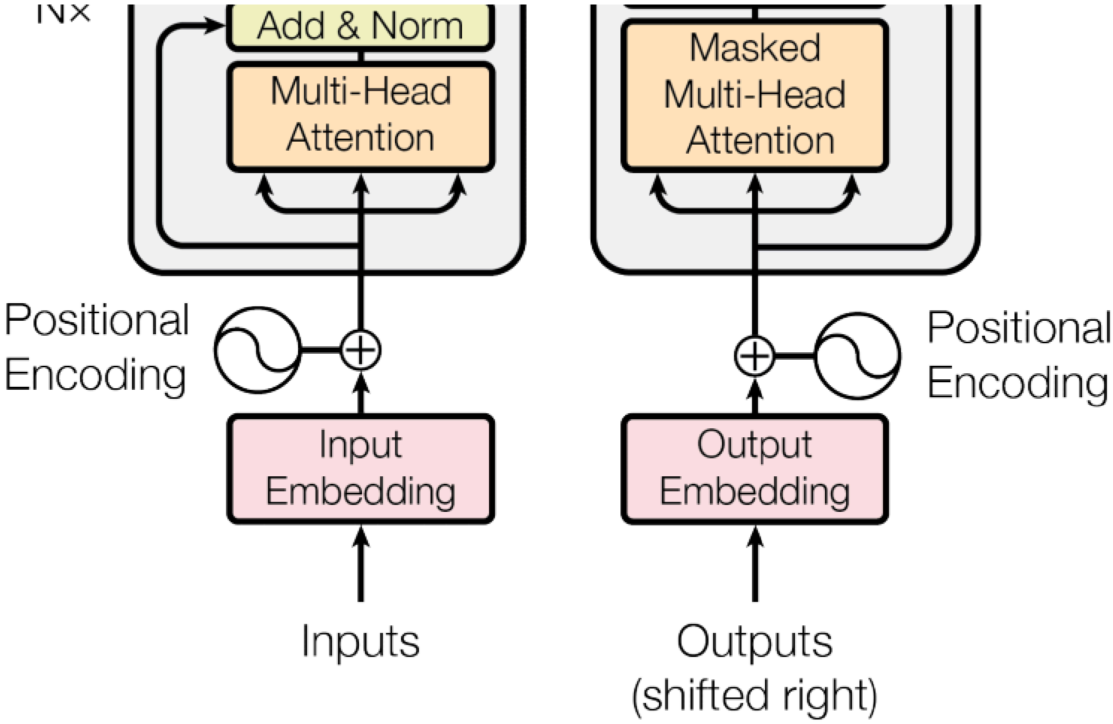


Figure 1: The Transformer - model architecture.

Embeddings -- 将单词表示为向量

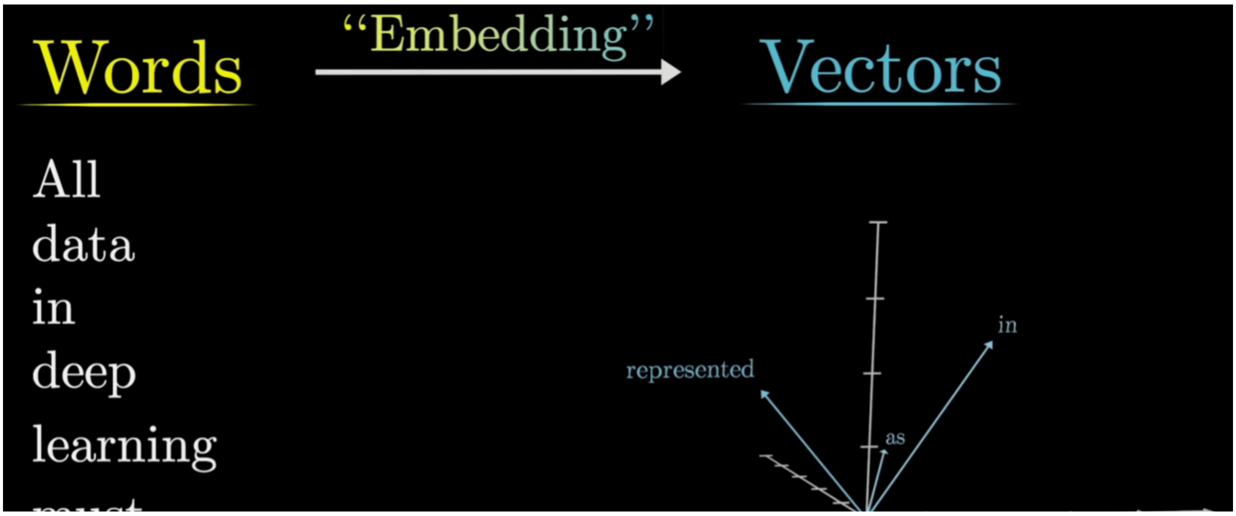
In [ ]:
```
# install gensim
%pip install gensim
```

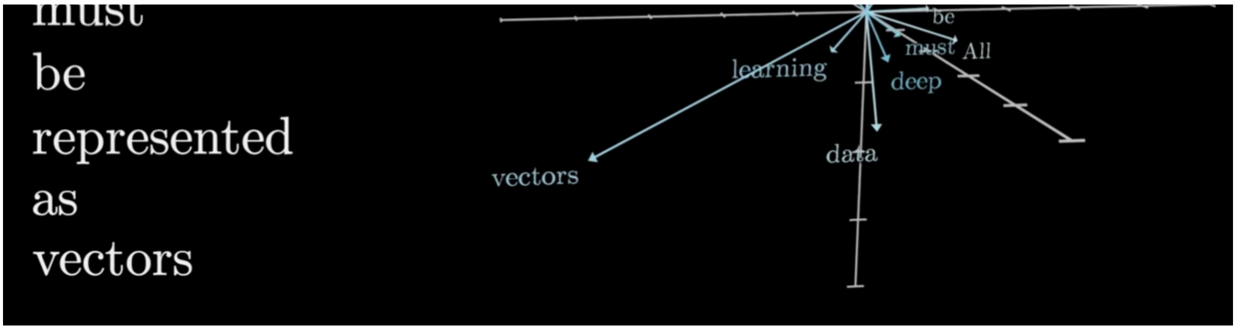
In [6]:
```
import gensim.downloader
model2 = gensim.downloader.load('glove-wiki-gigaword-50')
```

In [2]:
```
# 该模型中的每个向量100个维度
import gensim.downloader
model = gensim.downloader.load('glove-wiki-gigaword-100')
```

In [3]:
```
# 返回该模型包括的词汇量
len(model)
```

Out[3]: 400000



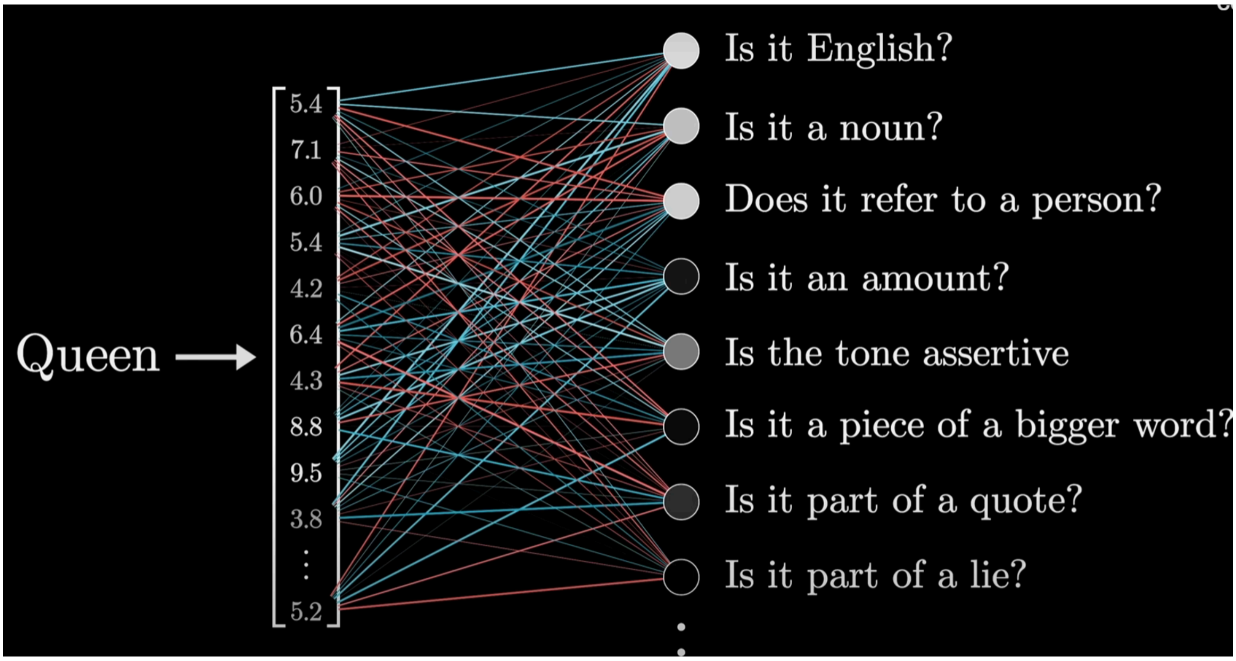


In [5]:

```
# 'glove-wiki-gigaword-100'模型每个单词的向量长度为100
# GPT3 的向量长度是12,288

model['queen']
```

Out[5]:

```
array([-0.50045 , -0.70826 ,  0.55388 ,  0.673   ,  0.22486 ,  0.60281 ,
       -0.26194 ,  0.73872 , -0.65383 , -0.21606 , -0.33806 ,  0.24498 ,
       -0.51497 ,  0.8568  , -0.37199 , -0.58824 ,  0.30637 , -0.30668 ,
       -0.2187  ,  0.78369 , -0.61944 , -0.54925 ,  0.43067 , -0.027348,
        0.97574 ,  0.46169 ,  0.11486 , -0.99842 ,  1.0661  , -0.20819 ,
        0.53158 ,  0.40922 ,  1.0406  ,  0.24943 ,  0.18709 ,  0.41528 ,
       -0.95408 ,  0.36822 , -0.37948 , -0.6802  , -0.14578 , -0.20113 ,
        0.17113 , -0.55705 ,  0.7191  ,  0.070014, -0.23637 ,  0.49534 ,
        1.1576  , -0.05078 ,  0.25731 , -0.091052,  1.2663  ,  1.1047  ,
       -0.51584 , -2.0033  , -0.64821 ,  0.16417 ,  0.32935 ,  0.048484,
        0.18997 ,  0.66116 ,  0.080882,  0.3364  ,  0.22758 ,  0.1462  ,
       -0.51005 ,  0.63777 ,  0.47299 , -0.3282  ,  0.083899, -0.78547 ,
        0.099148,  0.039176,  0.27893 ,  0.11747 ,  0.57862 ,  0.043639,
       -0.15965 , -0.35304 , -0.048965, -0.32461 ,  1.4981  ,  0.58138 ,
       -1.132   , -0.60673 , -0.37505 , -1.1813  ,  0.80117 , -0.50014 ,
       -0.16574 , -0.70584 ,  0.43012 ,  0.51051 , -0.8033  , -0.66572 ,
       -0.63717 , -0.36032 ,  0.13347 , -0.56075 ], dtype=float32)
```



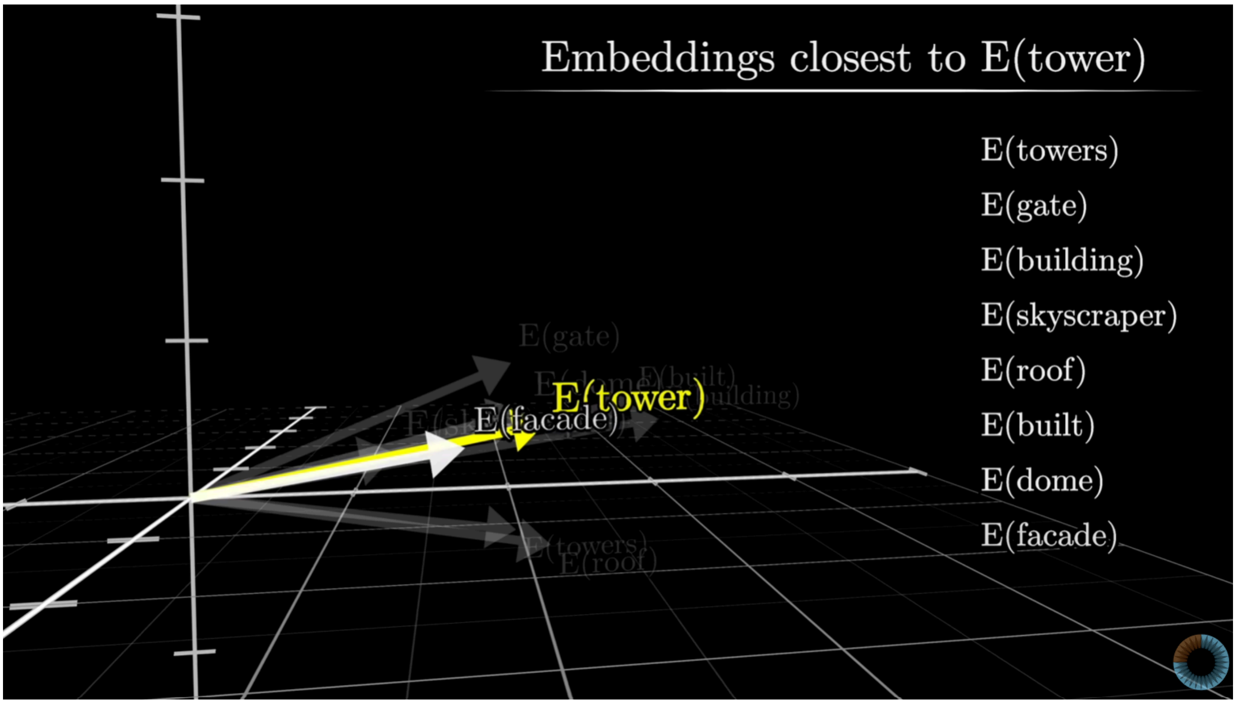
- 所有的词的向量都是通过大量语料训练学习得来的
- 向量的维度表达了单词的语义信息
- 但是每个维度的语义信息都是模糊的，没有准确定义的
- 将单词转换为向量后，可以进行向量的运算，可以寻找：
    - 最接近的词
    - 最不接近的词

In [8]:

```python
# tower 的近义词

import numpy as np
model.most_similar('tower')
```

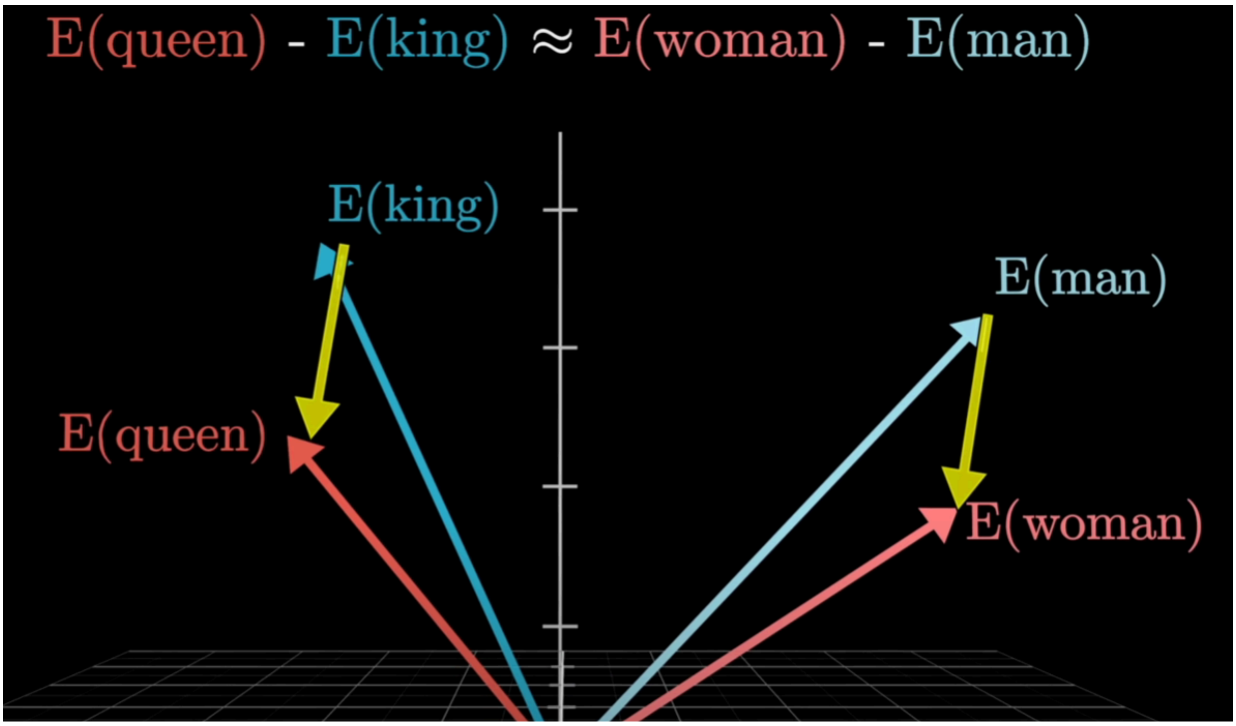
Out[8]:

```
[('towers', 0.8470372557640076),
 ('building', 0.725898027420044),
 ('dome', 0.6875219345092773),
 ('spire', 0.6807529926300049),
 ('gate', 0.671362578868866),
 ('skyscraper', 0.6699519753456116),
 ('roof', 0.6561244130134583),
 ('walls', 0.6556639075279236),
 ('built', 0.6550073623657227),
 ('buildings', 0.6522013545036316)]
```



In [ ]:

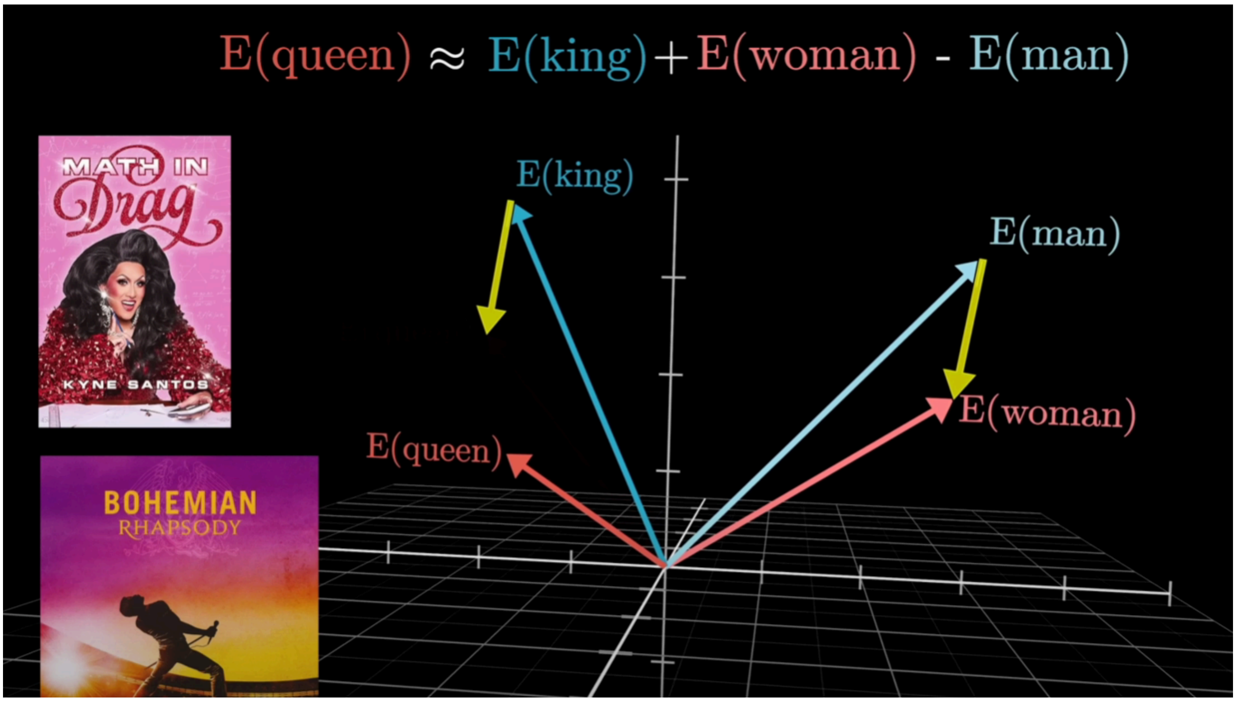
```python
# queen 的最不相似的词

model.most_similar(negative=['queen'])
```



In [9]:
```
# woman + king - man ~= queen

model.most_similar(positive=['woman', 'king'], negative=['man'])
```



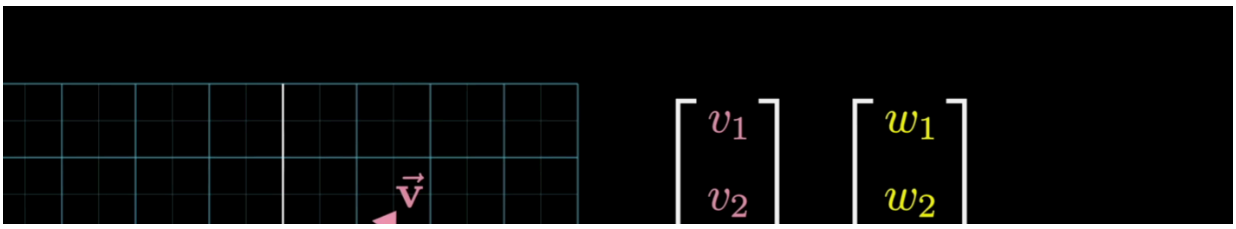
```
('princess', 0.6520534157752991),
('prince', 0.6517034769058228),
('elizabeth', 0.6464517712593079),
('mother', 0.631171703338623),
('emperor', 0.6106470823287964),
('wife', 0.6098655462265015)]
```

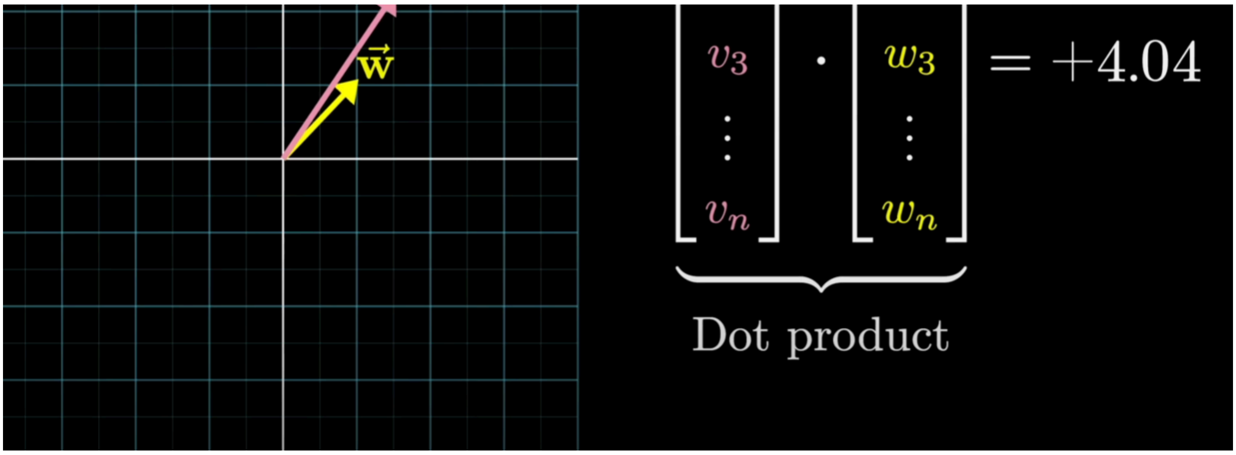


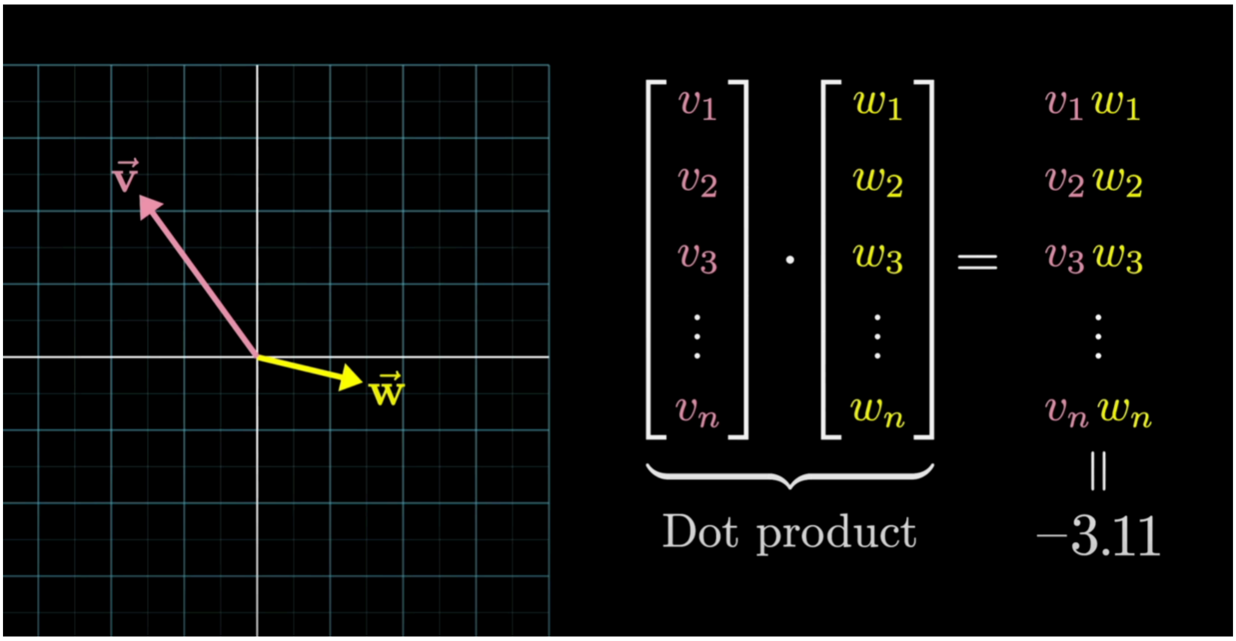
In [15]:
```
# good + happy - bad - sad ~= ?

model.most_similar(positive=['good', 'happy'], negative=['bad', 'sad'])
```

Out[15]:
```
[('enjoy', 0.4552291929721832),
 ('chance', 0.4535176753997803),
 ('ready', 0.45224252343177795),
 ('opportunity', 0.4434261918067932),
 ('excellent', 0.4415234923362732),
 ('free', 0.44127118587493896),
 ('maintain', 0.440281480550766),
 ('comfortable', 0.4352276027202606),
 ('healthy', 0.433483868835657),
 ('better', 0.43163517117500305)]
```

如何计算两个向量之间的相似度?





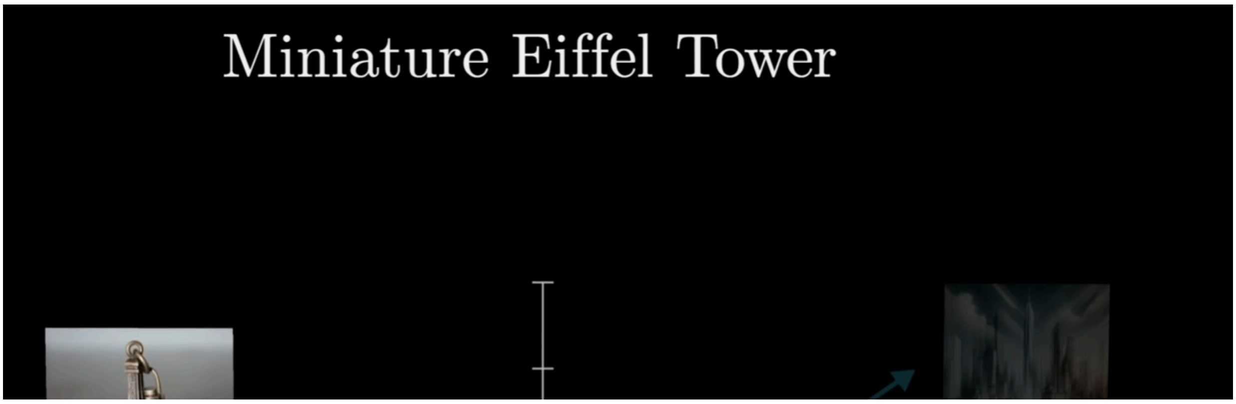


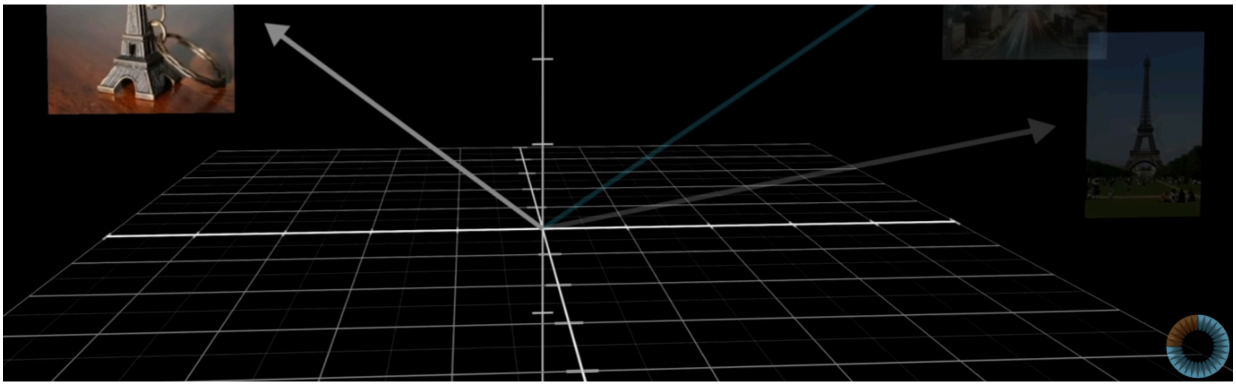
Attention

Embeddings向量的局限性：

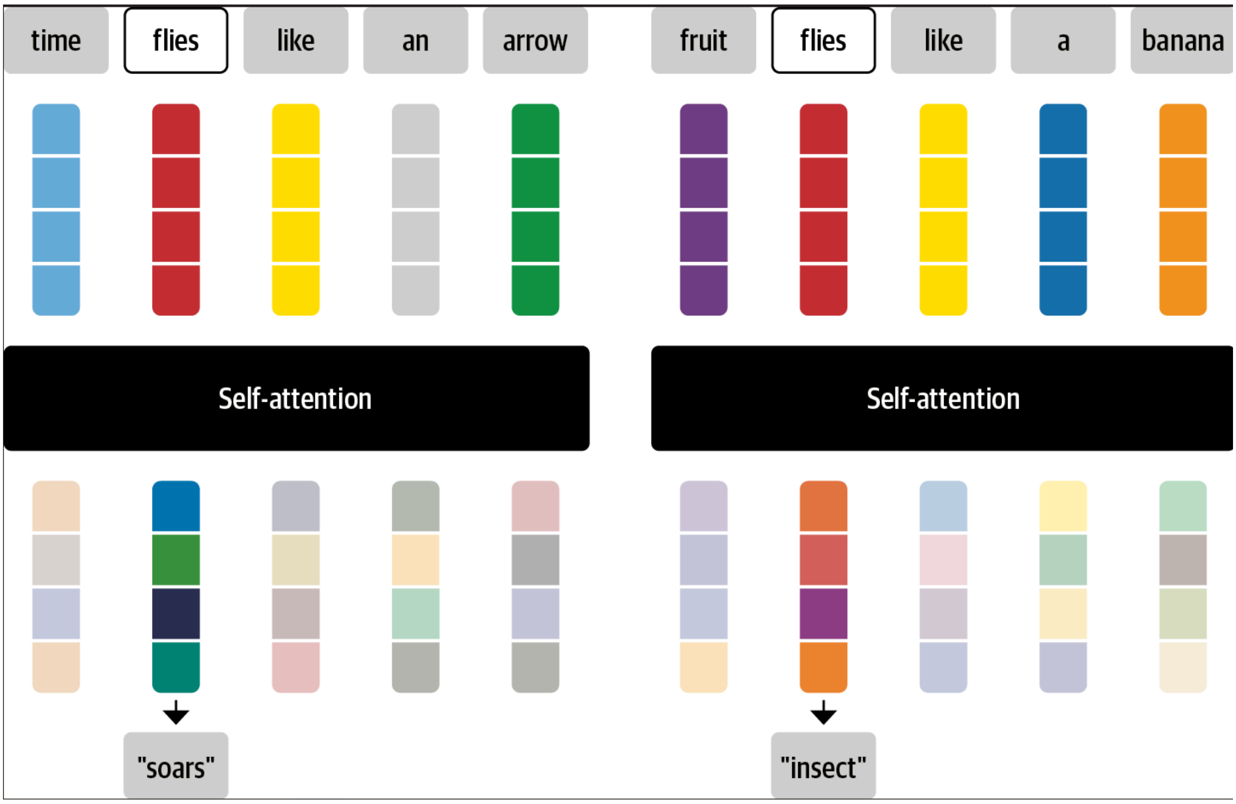
- 每个单词通常会有多个不同的含义，都包含在了一个向量中
- 缺乏上下文信息，无法区分不同的含义
- 例如：
    - `apple`
        - 可能是水果，也可能是苹果公司
    - `python`
        - 可能是一种动物，也可能是一种编程语言

Attention机制使得模型可以关注输入序列中不同位置的不同部分，从而更好地捕捉上下文信息。





Self Attention



Transformer Book