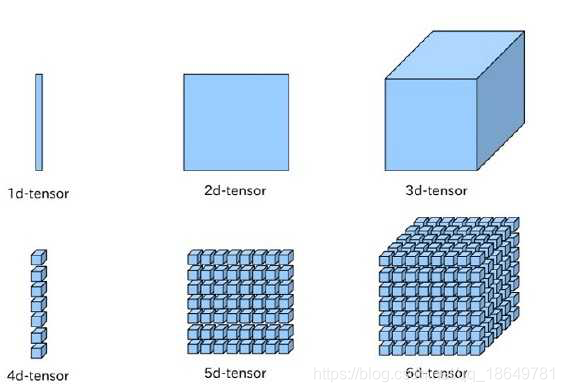
# 大模型

## Tips

与存储标量值的传统数据库不同，向量数据库专门设计用于处理多维数据点(通常称为向量)。这些向量表示多个维度的数据，可以被认为是指向空间中特定方向和大小的箭头。

Tensor张量 张量和shape和rank

**一阶张量可以理解成一个向量，二阶张量可以理解成矩阵，三阶张量可以理解成立方体，四阶张量可以理解成立方体组成的一个向量，五阶张量可以理解成立方体组成的矩阵，依次类推**



Agent 包含

Planning 任务拆解

Memeory 这不是session么

和tool use

真值

Rag是解决大模型幻觉的重要手段 和 解决时效性问题的重要手段

Json换 yaml就快了很多了

同时输出推理过程 看下模型为啥推错

Prompt推理步骤 写上可以调整 不指定的话他推理的不符合预期 问山问水propmt就有推理步骤

EbTiny关安全策略 可以提高速度

ann应该在5ms

有的query得引导 不然用户不会问 要搞些提示次

speed就干不了生成yaml json这种事

Copilot 副驾驶

AI应用的演化大体会遵循自主系统的发展过程，即Copilot→Agent→Al Worker。其中Age

nt的核心能力是决策和独立完成某些任务，可以自我学习和完善、高度个性化。AlWorker的

核心则是像人类一样融入社会组织，独立完成某个角色的工作，可以在工作中学习、培训，

并且可以与其他人或AI协作完成复杂项目。目前AI应用绝大部分处于Copilot阶段。

压缩 多轮query为一句话 来保存上下文 而不是 搞多个query 压缩 多轮为一个上下文

你得告诉模型尾号是什么 而不是让模型自己弄

tot tree of thouht

Cot chain of thought

看来教模型一下是真的可以有用 周一周日限行规则 告诉他 举个例子就会编号

可以给模型好几个示例

# 角色任务

你是一个专业的旅行规划师，你的任务是结合上下文，对用户问询需求进行意图理解，并按推理步骤填写YAML格式的各个字段。

##推理步骤

1. 理解用户的需求，生成找点或算路(ask\_main\_agent) 或者 行程问询(qa\_for\_plan)；注意，生成的指令只有一条；

2. 找点步骤，包括找酒店、美食、景点等地点及其相关问询。function为ask\_main\_agent, sub\_query为结合上下文理解后改写的找点query;

3. 算路步骤，包括计算两点或多点之间的路线及其相关问询。function为ask\_main\_agent，sub\_query为结合上下文理解后改写的算路query;

4. 行程问询，包括对行程方案的各项内容问询，一般为只有一个步骤，即只有一个step。sub\_query为结合上下文理解后改写的行程问询query;

# 示例1：上海迪士尼门票价格多少？要预约吗？

```YAML

- step: 1

function: ask\_main\_agent

sub\_query: 上海迪士尼门票价格多少？要预约吗？

```

# 示例2：到达昆明后，从机场到酒店坐地铁要多长时间

```YAML

- step: 1

function: ask\_main\_agent

sub\_query: 到达昆明后，从机场到桔子大酒店坐地铁要多长时间

```

# 示例3：最后一天的景点要预约吗？门票多少钱？

```YAML

- step: 1

function: ask\_main\_agent

# 提取上下文的最后一天的景点并改写原始query，且只生成一条指令

sub\_query: 故宫、颐和园要预约吗？门票多少钱？

```

# 示例3：这次总共花了多少钱

```YAML

- step: 1

function: qa\_for\_plan

sub\_query: 这次总共花了多少钱

```

# 上下文

- day1: 抵达昆明，游览滇池与西山，品尝过桥米线

城市: 昆明

景区列表:

- 滇池: 冬季观鸥的最佳打卡地

人流量：适中

营业时间：24小时营业

门票：免费

- 西山: 俯瞰滇池美景，体验自然风光

酒店列表:

- 昆明洲际酒店: 价格便宜、设施齐全

价格: 600元

美食列表:

- 老滇山寨·云南特色民族菜(金碧店)

- day2: 游览石林景区，感受世界自然遗产的壮丽

城市: 昆明

景区列表:

- 石林景区: 世界自然遗产，世界地质公园，5A级景区

酒店列表:

- 昆明喜来登酒店: 价格便宜、设施齐全

价格: 523元

美食列表:

- 菌故·野生菌火锅(南屏店)

- day3: 前往大理，游览洱海与大理古城，体验白族文化

城市: 大理

景区列表:

- 洱海: 来大理必玩景点，拍照出片，超美超治愈

- 大理古城: 文艺安静慢生活，体验白族文化

酒店列表:

- 大理国际大酒店: 价格便宜、设施齐全

价格: 860元

美食列表:

- 大理段公子·天龙八部特色体验店

用户需求：

这次行程都有哪些景点

第三天的酒店到机场有多远？20点的飞机，几点出发合适？

军事博物馆周六开门吗？要预约吗？

你的意图理解：

带意图理解的tool就是agent

没有sft的模型上线太低了

sub\_query为结合上下文理解后改写的找点query

“ERNIE-4.0-8K 文心一言”中的“8K”表示该模型的上下文窗口长度为8000个字符或字词。也就是说，模型在处理输入时，最多可以同时处理8000个字符的内容。这一指标通常用于衡量大型语言模型在单次交互中能够处理的最大文本长度。

上下文窗口长度（Context Window Size）是指模型在一次推理过程中能够“记住”或处理的文本长度。

让子agent去改写query 问询的能力 搜点的能力 算路的能力传给 agent去做

Sft的模型的输出 不用在prompt里面搞

## 机器学习 深度学习 神经网络

机器学习是让机器从经验中学习，逐渐提高完成任务的能力，而不是通过严格的规则来指导。

深度学习是机器学习的一种方式，就像大脑的神经网络一样，通过多层次的学习处理复杂的任务。

这3个名词其实是第三代神经网络里运用非常多3大算法：

DNN（深度神经网络）、

RNN（递归神经网络）、

CNN（卷积神经网络）。

## Sft和loar

SFT（Supervised Fine-Tuning，监督微调）和 LoRA（Low-Rank Adaptation，低秩适应）都是用于微调大型语言模型的方法，但它们在实现方式和目的上有所不同。

1. SFT（监督微调）：

基本概念： SFT 是指使用有标注的监督数据，对预训练的语言模型进行微调，使其在特定任务上获得更好的性能。

实现方式： 在 SFT 中，模型的全部参数都会参与训练。通过在 labeled 数据集上进行梯度更新，模型能够学习到更贴合特定任务需求的表示。

优点：

能够充分调整模型，使其性能达到最佳。

适用于各种需要精准模型表现的任务。

缺点：

对于大型模型，训练成本高，资源消耗大。

需要大量的计算资源和时间。

2. LoRA（低秩适应）：

基本概念： LoRA 是一种高效的参数高效微调方法，旨在减少微调大型模型时需要训练的参数数量，从而降低计算和存储成本。

实现方式：

LoRA 通过在预训练模型的权重矩阵中引入低秩分解，仅对一些新增的低秩矩阵进行训练，而将原始模型参数冻结（不更新）。

具体来说，LoRA 在模型的线性层中添加了可训练的低秩矩阵，这些矩阵的参数量远小于原始权重矩阵。

优点：

大幅减少了需要训练和存储的参数数量，降低了显存和计算资源的需求。

提高了微调的效率，使得在资源有限的环境中也能微调大型模型。

方便地在多个任务之间共享基础模型，仅需存储不同任务的 LoRA 参数。

缺点：

由于只更新了部分参数，模型的性能提升可能不如全参数微调（如 SFT）那么显著。

可能需要仔细调节低秩矩阵的秩值等超参数，以获得最佳效果。

3. 区别总结：

更新参数的范围：

SFT：更新模型的全部参数，训练过程中所有权重都会调整。

LoRA：仅训练新增的低秩矩阵参数，原始模型参数保持冻结状态。

资源消耗：

SFT：资源消耗大，需要高算力和大量显存，特别是对于数十亿参数的大模型。

LoRA：资源消耗低，显著降低了显存占用和计算需求，适合在普通硬件上运行。

适用场景：

SFT：适用于对模型性能要求极高，且有充足计算资源的场景。

LoRA：适用于需要高效微调，资源有限，或需要在多个任务间快速切换的场景。

模型性能：

SFT：由于更新了所有参数，理论上可以达到最佳性能。

LoRA：虽然资源占用低，但性能可能略逊于全参数微调，需要在效率和效果之间权衡。

总结：

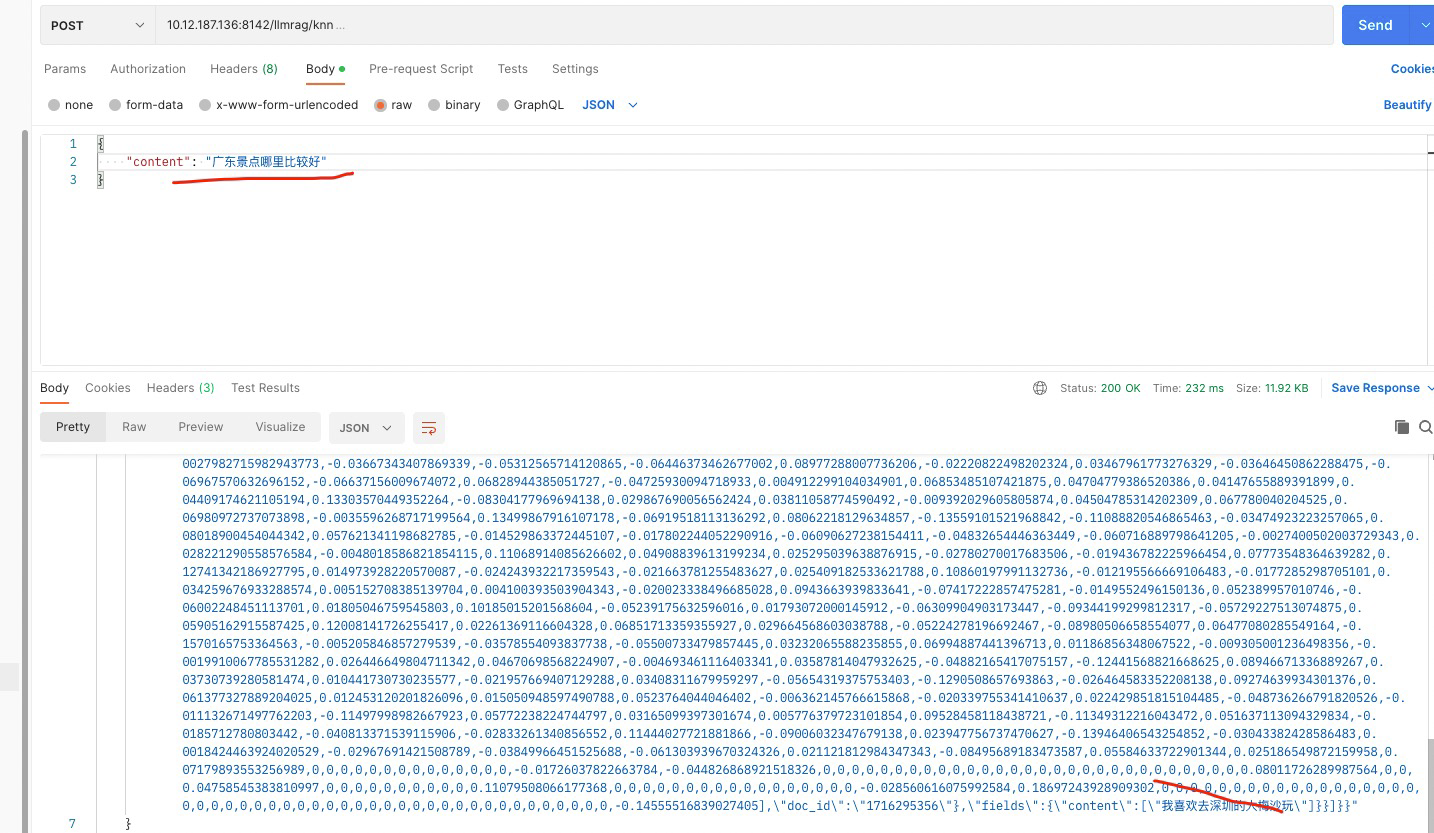
SFT 是一种微调策略，利用有监督的数据对模型进行全参数更新，以适应特定任务。

LoRA 是一种微调技术，旨在提高微调大型模型的效率，通过引入低秩适应，仅更新少量新增参数。

两者并非相互排斥的关系，实际上，LoRA 可以用于 SFT 中，即在监督微调时，通过 LoRA 方法高效地更新参数。因此，LoRA 是对微调方法的一种优化，可以在保持模型性能的同时，大幅降低微调的资源消耗。

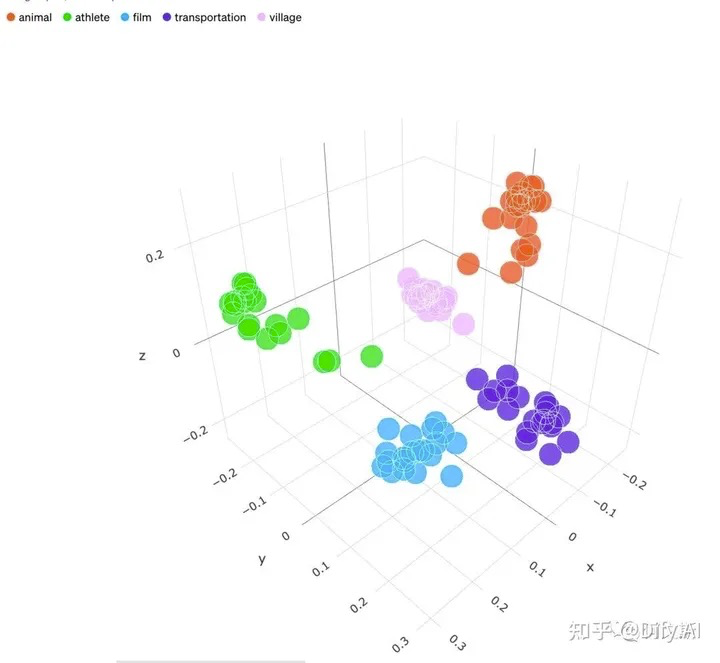
## 向量相关

### 向量



这两个一个字都不一样 都可以根据语义召回回来 溜啊 这是es所不能的

我们只有 2 个维度：X 轴和 Y 轴。实际上，Embedding 模型会提供更多的维度来表示人类语言的复杂度。比如 OpenAI 的 Embedding 模型 text-embedding-ada-002 会输出 1536 个维度。这足以让计算机来理解文本语义的细微差别。对于相似文本的分类，在三维空间内看可能是这样的（见下图），动物、运动员、电影、交通工具、村庄这些文本在空间中的距离是相近的。



slaver的emb的维度是384

维度越多 表示的信息越多

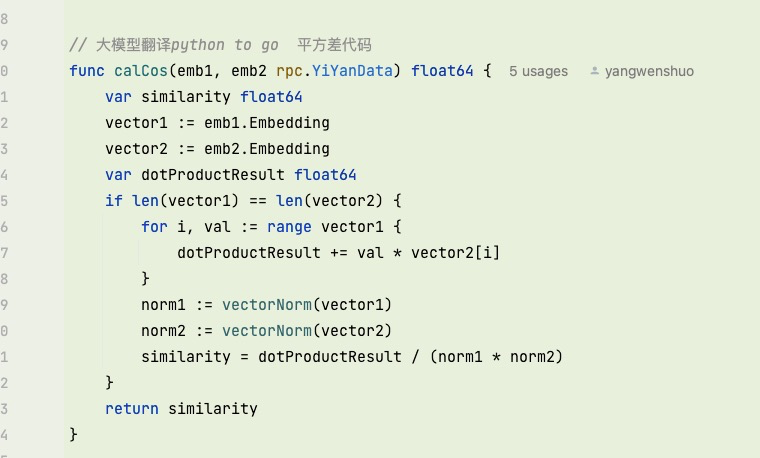
向量度量有四种：欧式距离、余弦、内积、海明距离

在使用向量内积代表相关性时，有一个前提：向量归一化到单位向量，本质是余弦距离

余弦相似度在 -1 到 1 的范围内建立相似度。通过测量向量空间中两个向量之间的角度的余弦，它可以确定向量是完全相反的（用 -1 表示）、正交的（用 0 表示）或 相同（用 1 表示）。

欧氏距离通过测量向量之间的直线距离来确定 0 到无穷大范围内的相似度。 相同的向量用0表示，而较大的值表示向量之间的差异较大。

点积相似性方法确定负无穷大到无穷大范围内的向量相似性。 通过测量两个向量的大小和它们之间角度的余弦的乘积，点积将负值分配给彼此远离的向量，将 0 分配给正交向量，将正值分配给指向相同方向的向量。



这个实际是点积除范数得到的相似性

KNN（K-Nearest Neighbor）

近似近邻（Approximate Nearest Neighbor，ANN）

KNN算法，全称为K最近邻（K-Nearest Neighbor）算法，是一种基于实例的学习或对实例的学习（也称为懒惰学习）方法。它的核心思想是如果一个样本在特征空间中的K个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。KNN算法不仅可以用于分类，还可以用于回归。通过找出一个样本的K个最近邻居，将这些邻居的属性的平均值赋给该样本，就可以得到该样本的属性。更有用的方法是将不同距离的邻居对该样本产生的影响给予不同的权值（weight），如权值与距离成反比。12

KNN算法的主要优点包括：

理论成熟，思想简单，既可以用来做分类又可以做回归。

可以用于非线性分类。

训练时间复杂度比支持向量机之类的算法低。

和朴素贝叶斯之类的算法比，对数据没有假设，准确度高，对异常点不敏感。

由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属的类别，因此对于类域的交叉或重叠较多的待分类样本集来说，KNN方法较其他方法更为适合。

然而，KNN算法也存在一些缺点：

计算量大，尤其是特征数非常多的时候。

样本不平衡的时候，对稀有类别的预测准确率低。

KD树、球树之类的模型建立需要大量的内存。

是慵懒散学习方法，基本上不学习，导致预测时速度比起逻辑回归之类的算法慢。

相比决策树模型，KNN模型的可解释性不强。

KNN算法的另一个不足之处是当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数。此外，KNN算法只计算“最近的”邻居样本，某一类的样本数量很大，那么或者这类样本并不接近目标样本，或者这类样本很靠近目标样本。无论怎样，数量并不能影响运行结果。可以采用权值的方法（和该样本距离小的邻居权值大）来改进。

首先我们了解下什么是向量，所谓向量就是由n个数字（二值向量由n个比特组成）组成的数组，我们称之为n维向量。而向量检索就是在一个给定向量数据集中，按照某种度量方式，检索出与查询向量相近的K个向量（K-Nearest Neighbor，KNN），但由于KNN计算量过大，我们通常只关注近似近邻（Approximate Nearest Neighbor，ANN）问题

余弦相似度计算就是看两个向量的夹角是不是够小

这里是二维平面，由于把每一个数都映射到[0,1]之间故，实际把坐标(x,y)在实数域的点，都映射到[0,1]之间，最终在一个长宽都在[0,1]之间的方块空间内。多维空间也是一样，最终这个多维体的每个维度都在[0,1]之间

归一化保证了点的各个维度都是同比例缩放

和点击相关的是余弦相似度来衡量相似性的，只是取点积的化，可以通过让向量变大的方式来作弊，因为 a→∗b→=|a|→|b|→cosθ{\vec{a}}\*\vec{b}={\vec{|a|}}\vec{|b|}cos\theta {\vec{a}}\*\vec{b}={\vec{|a|}}\vec{|b|}cos\theta ,那么为什么要这样做呢1、点积是最简单的操作，如果余弦相似需要更多的数学运算，增加计算复杂度2、如果一个向量特别大，去计算单词k的概率也会大，结果导致周围其他单词的概率也相应变大，所以不能通过增大向量来作弊。因此，以上两个原因导致可以用点积的大小表示相似度的大小

向量量化

前面把主流的向量检索的索引技术基本都概述了一遍，经过索引结构对考察的向量做了裁剪以后，对高维向量而言，单个的计算量还是很大，有没有方法来减少计算量呢？量化正是基于这个目的技术，所谓量化是指把一个很大的值空间量化到一个较小的值范围，举个简单的例子，全世界有60多亿人口，也就是地球人的表征有60多亿种，我们可以把人量化为男人和女人两种类型，这样就把一个60多亿的空间量化成只有两个值的范围

### 向量检索

向量检索有 基于树的方法 哈希方法 基于倒排方法 基于图的方法 HNSW

Elasticsearch uses the HNSW algorithm to support efficient kNN search. Like most kNN algorithms, HNSW is an approximate method that sacrifices result accuracy for improved search speed. This means the results returned are not always the true k closest neighbors.

Es用的是hnsw

解决的问题

向量检索从本质上讲，其思维框架和传统的检索方法没有什么区别，后面在讲解向量检索的索引结构部时，体会能更深刻一些。

1. 减少候选向量集

和传统的文本检索类似，向量检索也需要某种索引结构来避免在全量的数据上做匹配，传统文本检索是通过倒排索引来过滤掉无关文档，而向量检索是通过对向量建立索引结构来绕过不相关的向量，本文重点讨论相关的向量索引结构。

2. 降低单个向量计算的复杂度

传统文本检索在排序时通常会采用漏斗模型，上层计算比较轻量，越往下计算越精细，但随着过滤的进行，需要计算的文档数也逐级下降，对高维向量而言，单个向量的计算量还是比较重，通常会对向量进行量化，对向量做近似计算，最后在一个很小的数据集上做原始向量的排序。

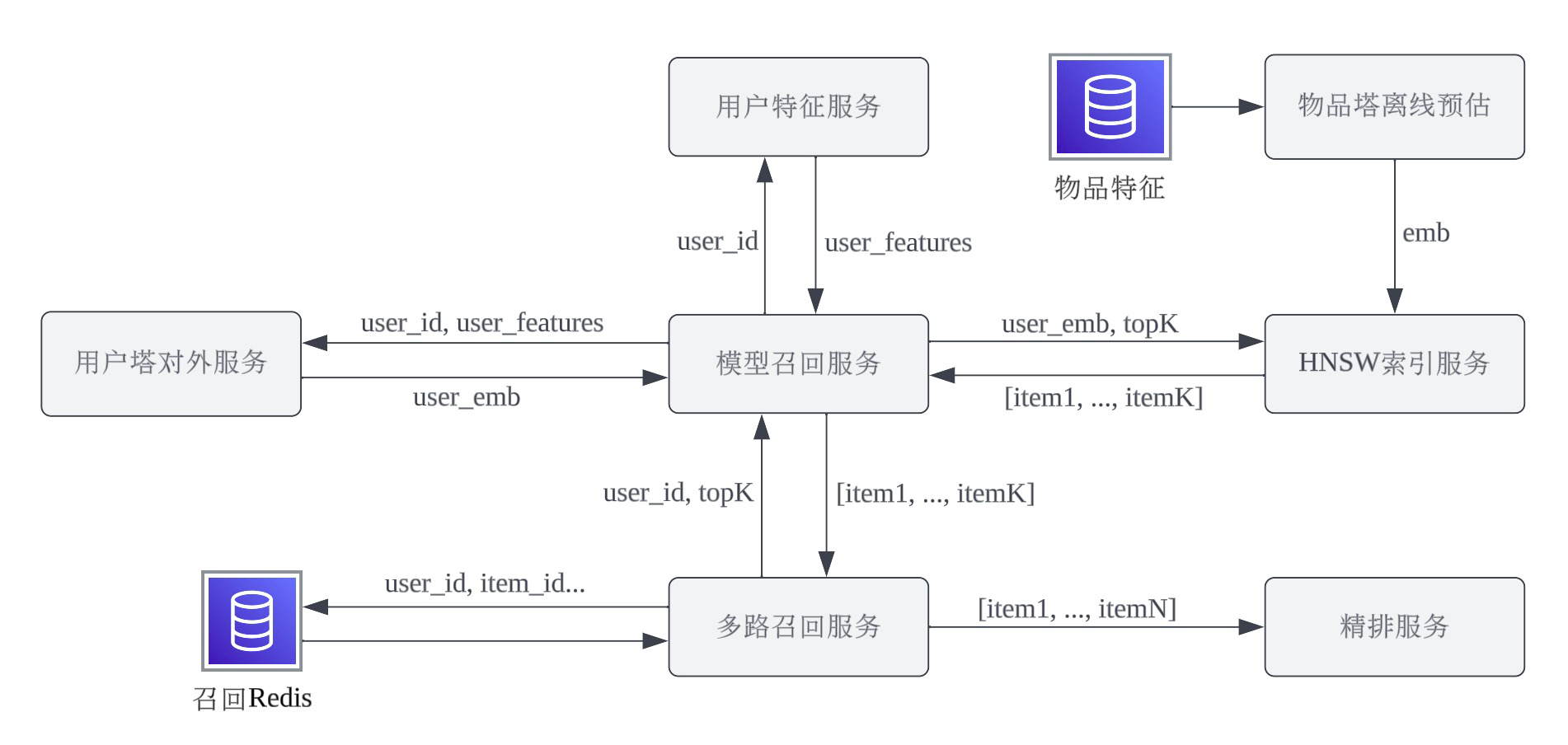
向量也可以搞倒排索引 类似 把向量再映射成 一个数 差不多的就用这个数倒排

### 向量库

Es天生就有向量的存储与检索

<https://blog.csdn.net/u014745465/article/details/135951141>

### 双塔



有用户有item的时候才是双塔模型

推荐场景

双塔模型结构简单，因此整体计算复杂度较低。用户和物品的特征之间没有交互，两个塔可以拆开独立使用。一般物品塔在离线使用生成全部物品向量，用户塔在线上使用生成用户向量。线上服务时结合ANN检索方法，可做到相关物品的快速检索。双塔召回一直是工业界召回算法中主流方式

## 反思强飞轮

反思就是影响模型的输入

长短期记忆