向量数据库

实践教程

腾讯云

文档目录

实践教程
向量数据库 AI 套件 + LLM 大模型：打造专属知识的问答服务

最近更新时间：2024-04-15 15:59:31

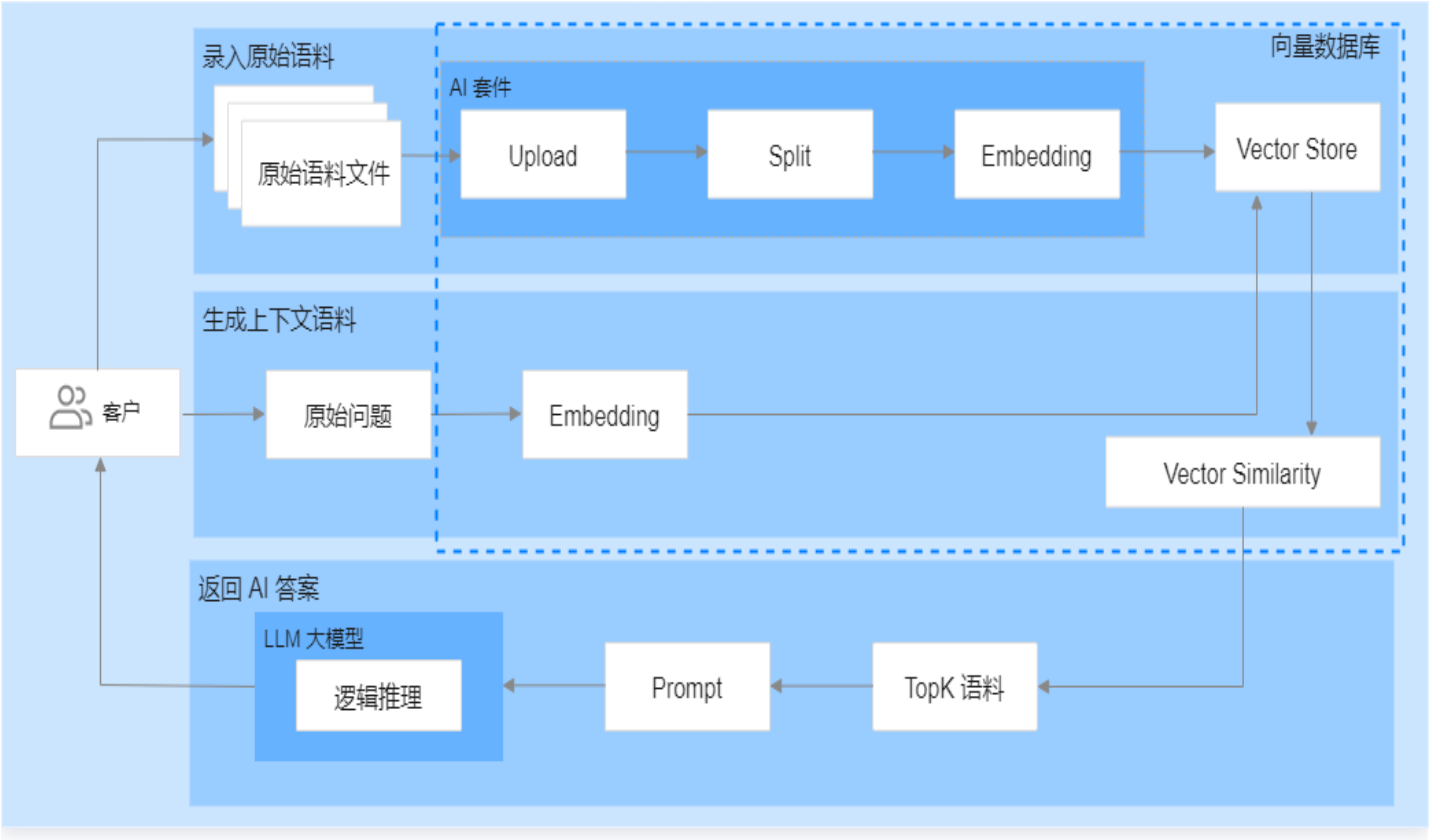
本教程演示如何使用腾讯云向量数据库（Tencent Cloud VectorDB）AI 套件一站式文档检索方案进行相似性检索，结合 LLM 大模型，构建专属知识库问答服务的方法。

背景信息

大语言模型（LLM）作为 NLP 服务领域的核心技术，具有丰富的 NLP 服务能力。然而，其训练数据主要涵盖普适知识和一般常识性知识，对于特定领域（如医疗保健、金融、科技等）的知识处理存在一定局限性。为了让 LLM 能够理解并获取存在于其训练知识范围外的特定领域知识，可以通过特定的提示构造来引导 LLM 在回答特定领域问题时理解意图，并根据注入的领域知识做出回答。本文以向量数据库作为向量检索的基础，搭建专属领域智能问答 AI 小助手，使 LLM 大模型的知识范畴得到针对性扩展。

实现思路

基于腾讯云向量数据库的 AI 套件对知识库文件进行上传、拆分和向量化，将文件和向量化数据存储于数据库中。借助腾讯云向量数据库的 Embedding 功能，将用户提出的问题转化为向量，在向量数据库中进行相似性检索，找出与问题相似度最高的语料。最后，将用户提出的问题和相似性检索的上下文语料进行组装，送入 LLM 大模型，进行逻辑匹配，生成问题答案。实现方案，如下图所示。



腾讯云向量数据库的 AI 套件提供了一套完整的一站式向量检索方案，包括数据切割和 Embedding 服务，无需自行编写拆分和向量化相关代码，减少了算法工程投入，极大简化了整个实现过程，降低了业务接入门槛。同时，相似性检索的上下文语料可以更有效地指导 LLM 大模型生成更精准的答案，进一步提升回答的准确性。并且，腾讯云向量数据库采用了灵活的存储策略，可根据实际变化的需求，及时优化更新知识库，保证了系统的稳定性。

实现步骤

本操作使用 Python SDK 录入 腾讯云向量数据库的产品文档，构建向量数据库的语料库。

步骤1：导入 Python SDK 依赖库

```
import requests
import json
import tcvectordb
import os
import tcvectordb
from tcvectordb.model.enum import ReadConsistency
from tcvectordb.model.collection_view import Embedding
```

步骤2：创建客户端对象，连接数据库实例

```
vdbclient = tcvectordb.VectorDBClient(url='{your vdb url}', username='root',
key='{your vdb key}')
```

步骤3：知识库初始化

声明 knowledgeInit() 函数，初始化知识库。

1. 为腾讯云向量数据库专有知识库创建一个 AI 类数据库 **testdb**。
2. 在已创建的 AI 类数据库中，创建集合视图 **knowledge**。
3. 上传 腾讯云向量数据库的产品文档 所有 md 文件于数据库集合视图中。

```
def knowledgeInit():
    #创建DB
    db = vdbclient.create_ai_database(database_name='testdb')
    #创建CollectionView
    collView =
db.create_collection_view(name='knowledge',embedding=Embedding(enable_wor
ds_embedding=True))
    #上传文件
    file_path = "{/yourLocalPath/files/}"
    for file_name in os.listdir(file_path):
        if file_name.endswith(".md"):
            print("\n上传:"+file_name)
            collView.load_and_split_text(local_file_path=file_path+file_name)
    print('upload all file sucess')
```

步骤4：传入问题进行知识内容相似性检索

声明 searchKnowlege() 函数，传入用户 question，返回知识库中与用户 question 最相似的内容。

```
def searchKnowlege(question):
    db = vdbclient.database('testdb')
    collView = db.collection_view('knowledge')
    doc_list = collView.search(
            content=question,
            limit=3
        )
    knowledge = ''
    print("查询向量数据库：")
    for count,doc in enumerate(doc_list):
            print("知识条目：", count, "----------")
            print(doc.data.text)
            knowledge += doc.data.text
    return knowledge
```

步骤5：将用户问题与知识库检索的相似性语料，送入大语言模型（LLM），生成问题答案

> **说明：**
> 如下以 Baichuan2-Turbo 大模型为例，检索所获取的相似性语料将更有效地指引 大语言模型（LLM）进行逻辑推理，生成更准确的答案。

```
def generate_answer(msg):
    url = "https://api.baichuan-ai.com/v1/chat/completions"
    api_key = "baichuan_api_key"
    data = {
        "model": "Baichuan2-Turbo",
        "messages": [{
                "role": "user",
                "content": msg
            }]
    }
    json_data = json.dumps(data)
    headers = {
        "Content-Type": "application/json",
        "Authorization": "Bearer " + api_key
    }
    response = requests.post(url, data=json_data, headers=headers,timeout=60)
    if response.status_code == 200:
        print("=================================================")
        print("知识条目+大模型推理回答:\n", json.loads(response.text)["choices"][0]
["message"]["content"])
    else:
        print(response.text)
        print("请求失败，状态码:", response.status_code)
```

步骤6：构建 main() 函数

1. 调用 knowledgeInit() 初始化知识库。
2. 输入问题，调用 searchKnowledge() ，传入请求问题，在知识库检索与问题相似的知识。
3. 将检索结果的 Topk 条 **knowledges** 与 输入的问题 **question** 进行内容组装。
4. 调用 generate_answer() ，传入组装后的内容，送入大模型，生成问题答案。

```
if __name__ == "__main__":
    knowledgeInit()
    question = input("请输入你的问题:\n")
```

```
    print("问题：", question)

print("==========================================================")

    knowledges = searchKnowlege(question)
    content = json.dumps({
            "请回答问题": question,
            "背景知识如下": knowledges
        },ensure_ascii=False)
    generate_answer(content)
```

问答效果体验

腾讯云向量数据库产品文档中的一些内容，通用的大语言模型（LLM）在预训练时若没有相关的数据，出现了幻觉问题，导致在回答时的效果较差。在使用腾讯云向量数据库 AI 套件搭建知识库对大语言模型（LLM）进行知识补充后，如下三个问题都能准确地回答，对比只使用大语言模型（LLM）的效果显著提升。

问题1：什么是腾讯云向量数据库的 Embedding 功能?

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。


```
{
    "index": 0,
    "message": {
        "role": "assistant",
        "content": "腾讯云向量数据库（TencentDB for TensorBase）的Embedding功能是一种针对大规模高维稀疏特征数据设计的存储和检索技术。它主要用于处
            理深度学习中的嵌入向量，如Word2Vec、GloVe等词嵌入模型生成的向量。\n\nEmbedding功能的主要特点包括：\n\n1. 高效存储：支持将高维稀疏特征压
            缩为低维稠密向量，并存储在数据库中，降低存储空间需求。\n\n2. 快速检索：提供高效的向量检索功能，支持多种相似度计算方式，如余弦相似度、欧几里
            得距离等。\n\n3. 分布式扩展：支持水平扩展，通过分片和复制提高系统的吞吐量和可用性。\n\n4. 灵活的数据处理：支持实时更新和查询操作，方便用户
            对数据进行实时分析和处理。\n\n5. 兼容性强：支持多种编程语言和开发框架，方便用户进行二次开发和集成。"
    },
    "finish_reason": "stop"
}
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

```
问题： 什么是腾讯云向量数据库的Embedding功能
==========
查询向量数据库:
知识条目： 0 ----------
Embedding 功能是腾讯云向量数据库（Tencent Cloud VectorDB）提供将非结构化数据转换为向量数据的能力，目前已支持文本 Embedding 模型，能够覆盖多种主流语言的向量转换，包括但不限于中文、英文。开启 Embedding 功能并在创建 Collection 时配置模型，在插入、更新和相似性检索数据时直接传入原始文本，向量数据库会自动将原始文本进行转换，生成对应的向量数据后插入数据库或进行相似性计算，大幅提高业务接入效率。

知识条目： 1 ----------
## Embedding 实现架构
腾讯云向量数据库（Tencent Cloud VectorDB）通过 HTTP API 将这些非结构化文本数据送入向量数据库，向量数据库将原始文本数据转交给 Embedding 模型进行向量化，再将转换后的向量数据以及原始文本一并存储在向量数据库中。其整个实现架构，如下图所示。

知识条目： 2 ----------
## 计费说明
腾讯云向量数据库（Tencent Cloud VectorDB）默认开通 Embedding 功能。在使用 Embedding 功能时，腾讯云向量数据库（Tencent Cloud VectorDB）将会根据输入文本的 **Token **数量进行计费。目前在公测阶段，暂不计费。
> **说明：**
>
> 在 Embedding 模型中，Token 是指文本数据处理的基本单元。通常在文本中，一个 Token 可以是一个字或词，也可以是一个标点符号。
==========
知识条目+大模型推理回答：
 腾讯云向量数据库（Tencent Cloud VectorDB）的Embedding功能是一种将非结构化数据（如文本）转换为向量数据的能力。这种转换是通过预训练的Embedding模型来实现的，这些模型能够将文本数据映射到高维空间中的向量表示。

Embedding功能支持多种主流语言，包括中文和英文。当您启用此功能并在创建Collection时配置相应的模型，您可以直接将原始文本数据插入、更新和用于相似性检索。向量数据库会自动处理这些文本数据，将其转换为向量形式，从而简化了数据处理流程并提高了业务接入效率。

腾讯云向量数据库通过HTTP API接收非结构化文本数据，然后将这些数据传递给Embedding模型进行向量化处理。转换后的向量数据和原始文本一起存储在向量数据库中。

关于计费方面，腾讯云向量数据库默认开通Embedding功能。目前，在使用Embedding功能时，会根据输入文本的Token数量进行计费。在公测阶段，这项服务暂不收费。在Embedding模型中，Token是指文本数据处理的基本单元，它可以是一个字、词或标点符号。
```

问题2：什么是腾讯云向量数据库中的 AI 套件?

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。


```
"choices": [
    {
        "index": 0,
        "message": {
            "role": "assistant",
            "content": "腾讯云向量数据库中的AI套件是一种工具集，它为开发者提供了在向量数据库上运行机器学习算法的能力。这个套件可以帮助用户更轻松地实现图像识别、自然语言处理等AI功能，并优化这些功能的性能和效率。\n\n具体来说，AI套件可能包括以下组件：\n\n1. 预训练模型：提供一些已经训练好的机器学习模型，如图像分类、物体检测等，方便用户直接使用。\n\n2. 模型训练与优化：提供模型训练和优化的工具，帮助用户提高模型的准确性和性能。\n\n3. 数据处理：提供数据清洗、特征提取等功能，帮助用户更好地处理和分析数据。\n\n4. 接口与SDK：提供API接口和软件开发包（SDK），方便用户在自己的应用程序中集成和使用这些AI功能。\n\n5. 监控与分析：提供实时监控和分析工具，帮助用户了解AI功能的运行状况和效果。"
        },
        "finish_reason": "stop"
    }
],
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

```
问题： 什么是腾讯云向量数据库中的AI套件
========================================
查询向量数据库:
知识条目： 0 ----------------
## 什么是 AI 套件?
AI 套件是腾讯云向量数据库（Tencent Cloud VectorDB）提供的一站式文档检索解决方案，包含自动化文档解析、信息补充、向量化、内容检索等能力，并拥有丰富的可配置项，助力显著提升文档检索召回效果。用户仅需上传原始文档，数分钟内即可快速构建专属知识库，大幅提高知识接入效率。

知识条目： 1 ----------------
### AI 类 Database
AI 类 Database 是专门用于 AI 套件上传和存储文件的向量数据库系统，可用于构建知识库。用户可以直接将文件上传至 AI 类 Database 下的 CollectionView 中，自动构建个性化的知识库。
> **说明：**
>
> - AI 类 Database 不支持直接对向量数据进行操作，已上传的文件不支持更新文件内容。
> - 为便于区别，腾讯云向量数据库将可直接操作向量数据的数据库称为 **Base 类 Database**。用户可以将向量数据上传至 Base 类 Database 中进行存储和管理，并可以直接对向量数据进行操作和处理。
知识条目： 2 ----------------
## 设计思想
AI 套件检索方案提供完整的文档预处理和灵活的内容检索能力。用户只需上传 Markdown 格式的文档文件。腾讯云向量数据库将自动进行文本切分（Split）、信息补充、向量化（Embedding）和索引构建等一系列操作，完成知识库的建立。在进行检索时，会先基于切分后的内容进行相似度计算，并结合词（Words）向量进一步对检索结果进行精排，最终返回排名靠前的 Top K 条数据和其上下文内容。这种综合利用词级别做精排的检索方式，提供了更专业、更精确的内容检索体验。

========================================
知识条目+大模型推理回答：
 腾讯云向量数据库中的AI套件是一站式文档检索解决方案，它包括自动化文档解析、信息补充、向量化、内容检索等功能。AI套件可以帮助用户快速构建专属知识库，提高知识接入效率。

AI套件使用AI类Database来存储和管理文件，用户可以将文件上传到AI类Database下的CollectionView中，自动构建个性化的知识库。需要注意的是，AI类Database不支持直接对向量数据进行操作，也不支持更新已上传的文件内容。

在设计思想上，AI套件提供完整的文档预处理和灵活的内容检索能力。用户只需上传Markdown格式的文档文件，腾讯云向量数据库会自动进行文本切分、信息补充、向量化和索引构建等一系列操作，完成知识库的建立。在进行检索时，会先基于切分后的内容进行相似度计算，并结合词向量进一步对检索结果进行精排，最终返回排名靠前的Top K条数据和其上下文内容。这种综合利用词级别做精排的检索方式，提供了更专业、更精确的内容检索体验。
```

问题3：腾讯云向量数据库支持的最大 QPS

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。


```
"choices": [
    {
        "index": 0,
        "message": {
            "role": "assistant",
            "content": "腾讯云向量数据库TencentDB for OLTR（One-Click Low-Latency Time-series）的最大QPS取决于所选实例规格。不同实例规格具有不同的CPU、内存和IO能力，从而影响其处理请求的能力。\n\n具体支持的QPS数值，建议参考腾讯云官方文档或联系腾讯云客服获取详细信息。"
        },
        "finish_reason": "stop"
    }
],
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

```
问题： 腾讯云向量数据库最大支持多大QPS
==========
查询向量数据库:
知识条目： 0 ----------
## 节点类型
腾讯云向量数据库依据存储节点 CPU 与内存资源分配比例不同，分为**存储型**和**计算型**两类。
- **存储型：**主要用于存储和管理大规模的向量数据，其主要优势在于：提供低查询延迟，能够高效地存储和管理向量数据，特别适用于数据量大、数据增长快、查询 QPS 相对较低的场景，例如：人脸识别、图像搜索等。
- **计算型：**主要用于快速查找和检索向量数据，支持高并发的查询请求，其主要优势在于：提供更高的查询 QPS 和更低的查询延迟，适用于流量大、延迟敏感的场景，例如：实时推荐、广告投放等。

知识条目： 1 ----------
## 腾讯云向量数据库是什么?
腾讯云向量数据库是一款全托管的自研企业级分布式数据库服务，专用于存储、检索、分析多维向量数据。该数据库支持多种索引类型和相似度计算方法，单索引支持10亿级向量规模，可支持百万级 QPS 及毫秒级查询延迟。腾讯云向量数据库不仅能为大模型提供外部知识库，提高大模型回答的准确性，还可广泛应用于推荐系统、NLP 服务、计算机视觉、智能客服等 AI 领域。

知识条目： 2 ----------
具体信息，请参见 [鉴权方式](https://write.woa.com/document/116658115942817792)。
- **连接方式**：腾讯云向量数据库支持通过 HTTP 协议进行数据写入和查询等操作。具体信息，请参见 [连接方式](https://write.woa.com/document/116551590185791488)。
- **检索方法**：腾讯云向量数据库支持通过精确检索、相似度检索、混合检索的方法。
- 精确查询：基于标量（指一个单独的数值，例如文本字段、数值字段或日期字段，区别于向量等多维数据结构）字段精确查找数据的方式。
==========
知识条目+大模型推理回答：
 腾讯云向量数据库的最大支持QPS取决于具体的配置和部署环境。根据您提供的背景知识，腾讯云向量数据库可以支持百万级 QPS 及毫秒级查询延迟。然而，确切的QPS值可能会因实际应用场景、硬件资源、网络条件等因素而有所不同。

如果您需要了解特定场景下的最大QPS，建议您联系腾讯云的技术支持团队，以便他们为您提供个性化的建议和解决方案。
```

IVF 系列索引应用指南

最近更新时间：2024-05-28 21:48:13

IVF 索引方式数据导入速度快，且内存空间占用低，特别**适合于亿级大规模**高维向量数据的相似性检索，本文介绍通过 Python SDK 应用 IVF 索引的操作流程。

IVF 索引介绍

IVF 索引的基本原理是将向量数据集划分为多个子集，每个子集称为一个聚类中心或一个簇。每个簇都有一个代表性的向量，称为聚类中心向量。通过构建一个倒排表，将聚类中心向量与属于该簇的向量进行关联。
在进行搜索时，首先根据查询向量找到与之最相似的聚类中心向量，然后在该聚类中心对应的倒排表中查找更接近查询向量的具体向量。这种两级索引结构可以极大地减少搜索的计算量，提高搜索效率。

使用限制

- IVF 索引需要在插入一定量的数据后才能开始训练。因此，在插入数据时无需构建索引，等待数据插入完成后需重建 IVF 索引。
- 创建集合时，需配置参数 nlist， 聚类中心的数量，建议取值范围为 `[sqrt(单分片数据量)*4, sqrt(单分片数据量)*16]` 。每个分片内至少需要写入 30*nlist 条数据，最多选取 256*nlist 条数据进行模型训练。
- 重建索引过程中，不允许数据库写入，无法停止任务。

应用示例

导入 SDK

```
import tcvectordb
from tcvectordb.model.document import Document, SearchParams, Filter
from tcvectordb.model.enum import FieldType, IndexType, MetricType,
ReadConsistency
from tcvectordb.model.index import Index, VectorIndex, FilterIndex, IVFPQParams
```

创建客户端连接

如下示例可直接复制，运行之前，您需在文本编辑器将 api_key=A5VOgsMpGWJhUI0WmUbY********** 与 10.0.X.X 依据实际情况进行替换。

```
# 1. 创建客户端连接
client = tcvectordb.VectorDBClient(url='http://10.x.x.x', username='root',
                                   key='A5VOgsMpGWJhUI0WmUbY**********',
timeout=30,
```

```
                read_consistency=ReadConsistency.STRONG_CONSISTENCY)
```

创建 Database

```
# 2. 创建 Database
db = client.create_database(database_name='db_test_ivf')
```

创建 Collection

如下列出创建集合时，应用 IVF 系列索引需特别设置的关键参数。

参数	参数含义	取值说明
indexType	索引类型，目前支持的 IVF_FLAT、IVF_PQ、IVF_SQ4, IVF_SQ8, IVF_SQ16。	本文以 IVF_PQ 为例介绍。
nlist	在 IVF_FLAT 算法中，向量空间被划分为 nlist 个聚类中心。	建议取值范围为：[sqrt(单分片数据量)*4, sqrt(单分片数据量)*16]
M	指乘积量化中原始数据被拆分的子向量的数量。将原始数据向量拆分为 M 个子向量。每个子向量的维度为 D/M，其中 D 是原始向量的维度。然后，对每个子向量进行独立的量化，得到 M 个码本（codebook），每个码本对应一个子向量的离散化表示。最终，将 M 个码本拼接起来，得到原始向量的 PQ 编码（code）。	• 仅 IVF_PQ 类型涉及该参数，IVF 其他系列无需设置。 • M 必须能被 D （原始向量的维度）整除。

```
# 3. 创建Collection，并使用IVF_PQ索引
# 3.1  定义索引，其中向量索引为IVF_PQ
index = Index(
    FilterIndex(name='id', field_type=FieldType.String,
index_type=IndexType.PRIMARY_KEY),
    FilterIndex(name='name', field_type=FieldType.String,
index_type=IndexType.FILTER),
    VectorIndex(name='vector', dimension=128, index_type=IndexType.IVF_PQ,
            metric_type=MetricType.COSINE, params=IVFPQParams(m=2, nlist=10))
# nlist建议取值范围为[sqrt(单分片数据量)*4, sqrt(单分片数据量)*16]
)

# 3.2 创建集合
coll = db.create_collection(
    name='test_ivf',
    shard=1,
```

```
    replicas=2,
    description='this is a collection of test IVF_PQ',
    index=index
)
```

写入数据

> ⚠ **注意：**
>
> 如果创建 Collection 选择的索引类型为 IVF 系列：
>
> - 当第一次写入时，当前集合还没有向量索引，此时 **buildIndex** 必须为 **false**。插入原始数据之后，需通过 rebuild 训练数据并重建索引。
> - 当集合已经调用过 rebuild 创建索引后，集合已经存在向量索引，此时：
>   - 如果 **buildIndex** = **true**，表示新写入的数据会加入到已有的 IVF 索引中，但不会更新索引结构，此时新写入的数据可以被检索到。
>   - 如果 **buildIndex** = **false**，表示新写入的数据不会加入到已有的 IVF 索引中，此时新写入的数据无法被检索到。

```
# 4. 写入数据（此处写入的数据，均为随机生成的测试数据）
# 注意：IVF索引需要写入一定数据量后，才能开始训练，
#     每个分片内至少需要写入 30*nlist 条数据，最多能训练 256*nlist 条数据
data_num = 300
for i in range(data_num):
    tmp_id = str(i)
    tmp_name = ''.join(random.sample(string.ascii_lowercase, 5))
    tmp_vector = np.random.randn(1, 128)[0].tolist()
    res = coll.upsert(
        documents=[
            Document(id=tmp_id, vector=tmp_vector, name=tmp_name)
        ],
        build_index=False   # IVF索引在首次写入数据时，需要设置build_index为False
    )
```

Rebuild 索引

> ⚠ **注意：**
>
> - 单分片内行数少于 30*nlist（nlist 为聚类中心数量）不支持训练。
> - 重建索引过程中，不允许数据库写入，无法停止任务。

如下列出 Rebuild 时需配置的关键参数，请参见下表。

参数	参数含义	配置方法及要求
drop_before_rebuild	标识在重建索引时，是否需先删除旧索引再重建新索引。 **说明：** 重建索引需要占用额外的内存空间，数据量越大，消耗的内存空间越大。在重建索引之前，您需根据实际资源情况选择是否需先删除旧索引再重建，避免引起内存占满而阻塞业务正常运行。	取值如下所示： • **True**：重建之前，先删除旧索引再重建索引。 **说明：** 内存资源不足时，可先删除旧索引，在新索引还没有创建完成之前，该表无法正常读写。 • **False**：重建之前，不删除旧索引，创建新索引完成之后再删除旧索引。默认为 False。 **说明：** 内存资源足够的情况下，可不删除旧索引。在新索引还没有创建完成之前，该表可读，禁止写入。
throttle	标识是否限制构建索引的单节点 CPU 核数。 **说明：** 重建索引会消耗 CPU 资源，为防止资源打满影响写入或者检索等操作，请根据业务实际配置重建索引的 CPU 核数。	取值如下所示： • **0**：不限制 CPU 核数。在模型训练期间，会消耗大量的 CPU 资源。重建索引任务将会尽快执行，但可能会对其他集合的读写操作产生影响。 • **1**：CPU 核数为 1，即仅使用 CPU 1 核进行模型训练，可避免构建索引期间对其他集合产生影响，但任务执行较慢。

```
# 5. 重建索引
# dropBeforeRebuild：标识重建索引之前，是否需要先删除旧索引，再创建新索引。
# throttle：标识是否限制构建索引的单节点 CPU 核数。默认为 1，即使用1核进行训练；可设置为 0，表示不限制 CPU 核数。
coll.rebuild_index(drop_before_rebuild=False, throttle=1, timeout=30)
```

查看索引状态

```
# 6. 查询索引是否重建完成
db = client.database('db_test_ivf')
```

```
res = db.describe_collection('test_ivf')
print(vars(res))
```

返回参数 **indexStatus** 中的 **status** 标识当前 Collection 重建索引的状态，**startTime** 显示重建索引开始的时间。

- **ready**：表示当前 Collection 准备就绪，可正常使用。
- **training data**：IVF 索引下特定状态，表示当前 Collection 正在进行数据训练，即训练模型已生成向量数据。
- **building index**：表示当前 Collection 正在重建索引，即将生成的向量数据存储到新的索引中。
- **failed**: 重建索引失败，可能会影响集合读写操作。

> ⚠ **注意：**
> **training data** 与 **building index** 状态期间不可写入数据。若重建索引之前先删除旧索引，则集合不可进行相似性查询，只能进行精确查询。

```
{
  "code": 0,
  "msg": "operation success",
  "collection": {
    "database": "db_test_ivf",
    "collection": "test_ivf",
    "documentCount": 4,
    "indexes": [
      ......
    ],
    "indexStatus": {
      "status": "ready",
      "startTime": ""
    }
  }
}
```