向量数据库

其他功能

腾讯云

文档目录

其他功能
AI 套件
基本介绍

最近更新时间：2025-05-26 19:31:32

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，包含依赖模型的自动化文档解析、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。

快速接入

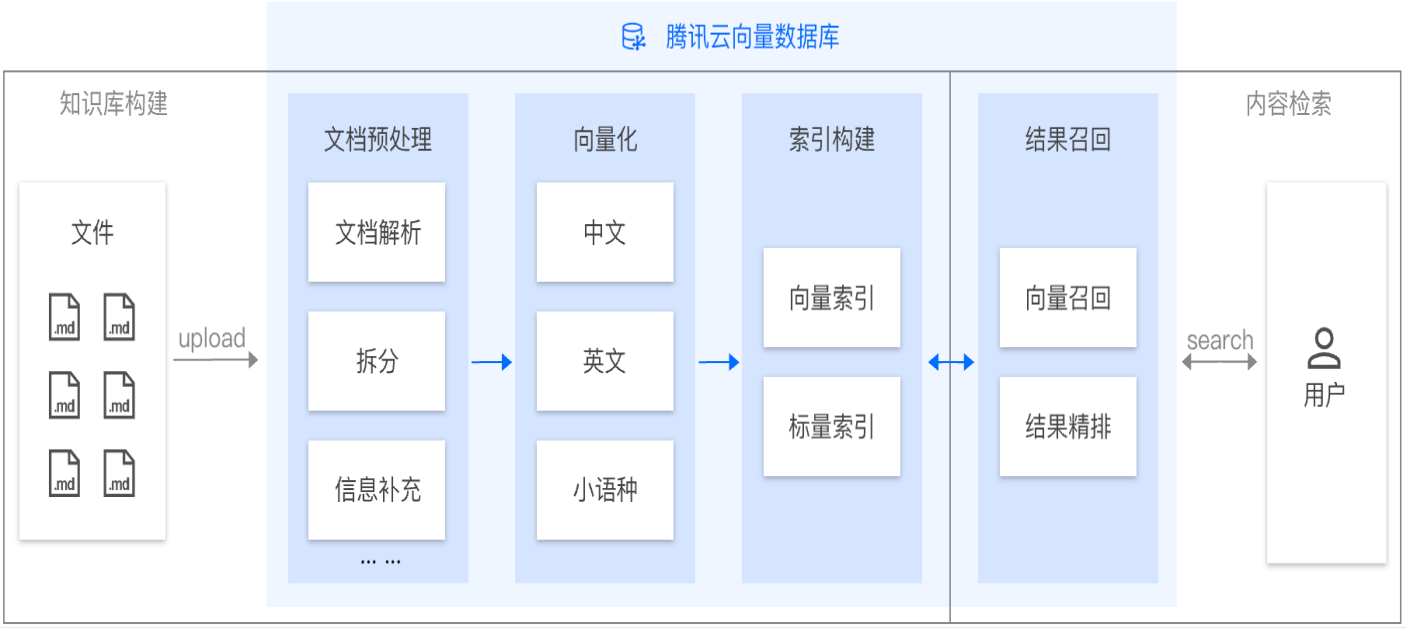
向量数据库支持基于 Collection 或 CollectionView 上传原始文本文件。基于 Collection 上传文件是调用了 CollectionView 的接口。

- （推荐）Collection：上传文件至 Collection 后，可通过标准接口访问数据并自主管理标量索引与账号权限，适用于需灵活管理、访问及修改拆分后数据的场景。具体代码示例，请参见 基于 Collection 上传文件 。
- CollectionView：文件上传至 CollectionView 后，通过其接口即可访问数据，适用于需快速构建并应用知识库的场景。具体代码示例，请参见 使用 AI 套件快速上传文件并检索 。

> **说明：**
> AI 套件当前可以免费体验，暂不计费，后续正式计费时间请留意腾讯云官方通知。

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、PDF、Word、PPT 等格式的文档文件。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。如下图以 Markdown 格式的文件为例。



基本概念

请先了解数据库设计的 逻辑结构 ，以便更好地理解 AI 套件相关的基本概念。

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。更多信息，请参见 Database 。

CollectionView

AI 类数据库文档组的集合视图，由多个 DocumentSet 组成，每个 DocumentSet 存储一组数据，对应一个文件数据。多个 DocumentSet 构成一个 CollectionView。

DocumentSet

相对 Document，DocumentSet 是 AI 类数据库中存储在 CollectionView 中的非结构化数据，是文件被拆分成多个 Document 的集合。每个 DocumentSet 存储一组数据，对应一个文件，是 CollectionView 下存储文件的最小单元。

Metadata

文件元数据，指上传文件时所携带的文件元数据信息，可以包括文件的名称、作者、创建日期、文件类型等信息。所有元数据被自动解析为标量字段，以 `Key-Value` 格式存储。用户可根据元数据构建标量字段的 Filter 索引，以检索并管理文件。

Word

词语，是智能文档检索中最小的分割粒度，通常由一个或多个字符组成。在结果召回时，将对召回段落中所有 Words 进行相似性计算，以便于根据词向量进一步对检索结果做精排。

约束与限制

1. 当前支持导入数据库的**文件类型**包含： Markdown(.md)、PDF(.pdf)、Word(.docx)、PPT(.pptx)，后续将逐步支持更多文件类型。
2. 每次只能上传一个文件，Markdown 类型文件最大限制为1MB，其余类型最大限制为10MB。若文件超过10MB，请 提交工单 处理。
3. 当前支持**地域**包含：北京、上海、广州、新加坡。

开发者工具

类别	Demo & API
Python SDK	SDK AI Demo
Java SDK	Java SDK Demo
Go SDK	Go SDK Demo
HTTP	HTTP API

基于 Collection 上传文件

最近更新时间：2025-05-26 14:46:42

向量数据库提供 Collection 的自动化文档处理能力，支持直接上传 Markdown、PDF、Word、PPT 等格式文件于 Collection。本文介绍基于 Collection 集合上传文件的关键操作。

SDK 参考

类别	SDK Demo
Python	upload_file.py
Java	VectorDBExampleWithCollectionUploadFile.java
Go	collection_upload_file_demo

创建 Collection

创建一个数据库集合来存储文件内容，同时为文件的关键字段（如文件名、文本内容、图片信息等）建立索引以优化查询效率。关键字段，如下表所示。

> **说明：**
> 创建数据库集合，文件的关键字段均可以**自定义**，在文件上传时，可通过 field_mappings 保留的文件参数逐一映射自定义的字段名。具体信息，请参见 上传文件 。

字段名	字段含义	索引设置要求或建议
file_name	标记文件名。	该字段**必须定义**为 string 类型的 filter 索引，以支持文件的过滤检索和处理同名文件覆盖的情况。
text	存放文件知识点的原始文本内容。	• 由于文本内容可能较大，**不建议**为该字段创建索引，以避免占用过多内存空间。 • 在查询时，只需要通过 output_fields 参数指定该字段即可返回原始文本。
section_num	标记段落（section）在原始文档中的顺序（如1, 2, 3, …）。	该字段可创建 filter 索引，以支持通过 section_num 的条件表达式，查询文件特定段落编号的内容。召回的内容也可以根据该值获取对应召回 chunk 所在段落的全部文本。
chunk_num	标记文本片段（chunk）在文件中的顺序（如1, 2, 3, …）。	该字段可创建 filter 索引，以支持通过 chunk_num 的条件表达式，按语块顺序范围

		检索（如查询第5-10块的内容）文件。本文示例为该字段创建索引。
image_list	存放 PDF 文件中图片的 Key 列表。 **说明：** 在原始检索出的文本块中，图片采用{key}占位符进行标记。系统会同步生成包含所有图片实际 URL 地址的索引列表，每个{key}对应唯一的图片链接。通过将文本块中的占位符替换为对应的<img>标签及真实 URL，即可完整还原原始文档的图文排版结构，最终生成符合网页标准的 HTML 文件。	**不建议**为该字段创建索引。如果需要创建索引，必须确保其为数组（array）类型，否则接口将报错，导致文件无法上传。

Python

```
import tcvectordb
from tcvectordb.model.enum import FieldType, IndexType, MetricType
from tcvectordb.model.index import Index, VectorIndex, FilterIndex,
HNSWParams
## 若需要使用文本信息进行相似性检索文件内容，则创建集合时需要指定 Embedding 模
型相关信息。
ebd = Embedding(vector_field='vector', field='text',
model_name='bge-base-zh')
coll = client.create_collection_if_not_exists(
    database_name="db-test",
    collection_name="coll-file-test",
    shard=2,
    replicas=1,
    description='test collection file upload',
    embedding=ebd,
    indexes=[
        FilterIndex('id', field_type=FieldType.String,
index_type=IndexType.PRIMARY_KEY),
        VectorIndex('vector', 768, IndexType.HNSW, MetricType.IP,
HNSWParams(m=16, efconstruction=200)),
```

```
        FilterIndex(name='file_name', field_type=FieldType.String,
index_type=IndexType.FILTER),
        FilterIndex(name='chunk_num', field_type=FieldType.Uint64,
index_type=IndexType.FILTER),
        FilterIndex(name='section_num', field_type=FieldType.Uint64,
index_type=IndexType.FILTER),
    ],
)

# print collection
print(vars(coll))
```

Java

```
package com.tencent.tcvectordb.examples;

import com.tencent.tcvectordb.client.VectorDBClient;
import com.tencent.tcvectordb.model.*;
import com.tencent.tcvectordb.model.Collection;
import com.tencent.tcvectordb.model.param.collection.*;
import com.tencent.tcvectordb.model.param.collectionView.*;
import com.tencent.tcvectordb.model.param.dml.*;
import com.tencent.tcvectordb.model.param.entity.GetImageUrlRes;
import com.tencent.tcvectordb.model.param.enums.EmbeddingModelEnum;
import com.tencent.tcvectordb.model.param.enums.ParsingTypeEnum;
import com.tencent.tcvectordb.utils.JsonUtils;
import com.tencent.tcvectordb.model.param.enums.OrderEnum;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.*;
import java.util.stream.Collectors;
// link database, client 为 VectorDBClient() 创建的客户端对象
Database db = client.database("db-test");
// 初始化 Colleciton 参数
CreateCollectionParam collectionParam =
CreateCollectionParam.newBuilder()
        .withName("coll-file-test")
        .withShardNum(2)
```

```
        .withReplicaNum(1)
        .withDescription("this is a collection upload file")
        .addField(new FilterIndex("id", FieldType.String,
IndexType.PRIMARY_KEY))
        .addField(new VectorIndex("vector", 768, IndexType.HNSW,
MetricType.IP, new HNSWParams(16, 200)))
        .addField(new FilterIndex("file_name", FieldType.String,
IndexType.FILTER))
        .addField(new FilterIndex("chunk_num", FieldType.Uint64,
IndexType.FILTER))
        .addField(new FilterIndex("section_num", FieldType.Uint64,
IndexType.FILTER))

.withEmbedding(Embedding.newBuilder().withModelName(EmbeddingModelEn
um.BGE_BASE_ZH.getModelName()).withField("text").withVectorField("ve
ctor").build())
        .build();
Collection collection = db.createCollection(collectionParam);
```

go

```
import (
    "github.com/tencent/vectordatabase-sdk-go/tcvectordb"
    "github.com/tencent/vectordatabase-sdk-go/tcvectordb/api"
    "github.com/tencent/vectordatabase-sdk-
go/tcvectordb/api/ai_document_set"
)
var (
    ctx        = context.Background()
    database   = "db-test"
    collection = "go-sdk-test"
)
db := client.Database(database)
index := tcvectordb.Indexes{}
index.VectorIndex = append(index.VectorIndex,
tcvectordb.VectorIndex{
    FilterIndex: tcvectordb.FilterIndex{
        FieldName: "vector",
        FieldType: tcvectordb.Vector,
        IndexType: tcvectordb.HNSW,
```

```
    },
    Dimension:  768,
    MetricType: tcvectordb.IP,
    Params: &tcvectordb.HNSWParam{
        M:              16,
        EfConstruction: 200,
    },
})
index.FilterIndex = append(index.FilterIndex,
tcvectordb.FilterIndex{FieldName: "id", FieldType:
tcvectordb.String, IndexType: tcvectordb.PRIMARY})
index.FilterIndex = append(index.FilterIndex,
tcvectordb.FilterIndex{FieldName: "file_name", FieldType:
tcvectordb.String, IndexType: tcvectordb.FILTER})
index.FilterIndex = append(index.FilterIndex,
tcvectordb.FilterIndex{FieldName: "chunk_num", FieldType:
tcvectordb.Uint64, IndexType: tcvectordb.FILTER})
index.FilterIndex = append(index.FilterIndex,
tcvectordb.FilterIndex{FieldName: "section_num", FieldType:
tcvectordb.Uint64, IndexType: tcvectordb.FILTER})
// 若需要使用文本信息进行相似性检索文件内容，则创建集合时需要指定 Embedding 模
型相关信息。
ebd := &tcvectordb.Embedding{VectorField: "vector", Field: "text",
ModelName: "bge-base-zh"}
coll, _ = db.CreateCollectionIfNotExists(ctx, collection, 2, 1,
"test collection", index, &tcvectordb.CreateCollectionParams{
    Embedding: ebd,
})
log.Printf("create collection if not exists success: %v: %v",
coll.DatabaseName, coll.CollectionName)
```

上传文件

为已创建的 Collection 上传文件，如下列出上传文件需要设置的关键参数。本示例，上传文件为 tcvdb.pdf 。具体接口，请参见 Python-upload_file() 、Java-UploadFile() 、Go-UploadFile() 。

关键参数	参数含义	补充说明
local_file_path	指定本地上传文件的路径。	-

metadata	文件的 Metadata 元数据信息，可自定义扩展字段。例如：author、tags 等。	• 上传文件时，可为创建 Collection 设置的 Filter 索引的字段赋值，以便在检索时，使用该字段的 Filter 表达式检索文件。 • 上传文件时，可以新增标量字段，但新增字段不会构建 Filter 索引。
splitter_process	文件拆分时，指定对关键字 keywords 和 Title 的处理规则。	• **append_title_to_chunk**：在对文件拆分时，配置是否将 Title 追加到切分后的段落后面一并 Embedding。 • **append_keywords_to_chunk**：在对文件拆分时，配置是否将关键字 **keywords** 追加到切分后的段落一并 Embedding。
parsing_process	指定 PDF 类型文件的解析方式.	• **VisionModelParsing**：文件依据解析模型解析，推荐使用，可解析 PDF 中双栏、表格等复杂格式。 • **AlgorithmParsing**：文件依据算法解析，系统默认解析方式。Markdown、Word、PPT 类型，无需配置该参数，默认使用 AlgorithmParsing 解析。
embedding_model	指定使用的 Embedding 模型的名称。您需根据业务的语言类型、数据维度要求等综合选择合适的模型。	取值如下所示。更多信息，参见 Embedding 介绍 。 • **bge-large-zh-v1.5**：适用中文，1024维，推荐使用。 • **bge-base-zh-v1.5**：适用中文，768维。 • **bge-large-zh**：适用中文，1024维。 • **bge-base-zh**：适用中文，768维。 • **m3e-base**：适用中文，768维。 • **e5-large-v2**：适用英文，1024维。 • **text2vec-large-chinese**：适用中文，1024维。 • **multilingual-e5-base**：适用于多种语言类型，768维。 • **BAAI/bge-m3**：适用于多种语言类型，1024维。
field_mappings	指定 Collection 中文件字段的映射关系。	• **filename**：指定文件名映射的字段。 • **text**：指定文件内容文本映射的字段。 • **image_list**：指定 PDF 文件图片 Key 列表存放的字段。 • **chunkNum**：标记文本片段（chunk）在文件中的顺序。 • **sectionNum**：标记段落（section）在原始文档中的顺序。

Pyhon

```
import tcvectordb
from tcvectordb.model.document import Filter, Document
from tcvectordb.model.collection_view import SplitterProcess,
ParsingProcess

# 上传文件
# 1. local_file_path：指定当前文件在客户端的路径
# 2. file_name：配置文件存储于向量数据库的名称，可不配置，从local_file_path
获取
# 3. splitter_process：配置文件拆分规则
# 4. parsing_process：指定文件解析方式
# 5. metadata：定义文件 meta 信息字段，示例中指定了author 与 tags
# 6. field_mappings：文件与数据库中存储的映射字段
# 7. embedding_model：Embedding 模型
res = client.upload_file(
            database_name='db-test',
            collection_name='coll-file-test',
            local_file_path="./tcvdb.pdf",
            metadata={
                'author': 'Tencent',
                'tags': ['向量', 'Embedding', 'AI']
            },
            splitter_process=SplitterProcess(
                append_keywords_to_chunk=True,
                append_title_to_chunk=False,
                # chunk_splitter 参数以正则表达式的方式配置文档拆分方式，如
下：\n{2,} 代表以两个及以上的换行进行拆分，常用在QA对文件拆分中。
                # chunk_splitter="\n{2,}",
            ),
            parsing_process=ParsingProcess(
                parsing_type='AlgorithmParsing'
            ),
            embedding_model='bge-base-zh',
            field_mappings={
                 "filename": "file_name",
                 "text": "text",
                 "imageList": "image_list",
                 "chunkNum": "chunk_num",
                 "sectionNum": "section_num",
```

```
            },
    )
print(res)
```

Java

```
// 配置文件 matadata 信息，示例中指定了author 与 tags
Map<String, Object> metaDataMap = new HashMap<>();
metaDataMap.put("author", "Tencent");
metaDataMap.put("tags", Arrays.asList("Embedding", "向量", "AI"));
// 配置文件与数据库中存储的映射字段
Map<String,String> columnMap = new HashMap<>();
columnMap.put("filename", "file_name");
columnMap.put("text", "text");
columnMap.put("imageList", "image_list");
columnMap.put("chunkNum", "chunk_num");
columnMap.put("sectionNum", "section_num");
// 上传文件
// 1. LocalFilePath：指定当前文件在客户端的路径
// 2. FileName:指定文件名
// 3. SplitterProcess：配置文件拆分规则
// 4. ParsingProcess：指定文件解析方式
// 5. metadata：定义文件 meta 信息字段，示例中指定了author 与 tags
// 6. FieldMappings：文件与数据库中存储的映射字段
// 7. EmbeddingModel：Embedding 模型
UploadFileParam param = UploadFileParam.newBuilder()
        .withLocalFilePath("../demo_files/tcvdb.pdf")

.withSplitterProcess(SplitterPreprocessParams.newBuilder().withAppen
dKeywordsToChunkEnum(true).Build())
        // parsingProcess is used for parsing pdf file by vision
model

.withParsingProcess(ParsingProcessParam.newBuilder().withParsingType
(ParsingTypeEnum.VisionModel).build())
        .withFileName("tcvdb.pdf")
        .withFieldMappings(columnMap)
        .withEmbeddingModel("bge-base-zh")
        .Build();
```

```
client.UploadFile("db-test", "coll-file-test", param, metaDataMap);
```

Go

```
appendKeywordsToChunk := true
appendTitleToChunk := false
localFilePath := "../demo_files/tcvdb.pdf"
param := tcvectordb.UploadFileParams{
    LocalFilePath: localFilePath,
    SplitterPreprocess: ai_document_set.DocumentSplitterPreprocess{
        AppendKeywordsToChunk: &appendKeywordsToChunk,
        AppendTitleToChunk:    &appendTitleToChunk,
    },
    ParsingProcess: &api.ParsingProcess{
        ParsingType: "AlgorithmParsing",
    },
    EmbeddingModel: "bge-base-zh",
    FieldMappings: map[string]string{
        "filename":  "file_name",
        "text":      "text",
        "imageList": "image_list",
        "chunkNum":   "chunk_num",
        "sectionNum": "section_num",
    },
    MetaData: map[string]interface{}{
        "author": "Tencent",
        "tags":   []string{"向量", "Embedding", "AI"},
    },
}
result, _ := client.UploadFile(ctx, database, collection, param)
log.Printf("UploadFile result: %+v", result)
```

相似性检索文件

指定检索的文本信息，指定检索的文件。检索与指定的文本信息相似，满足 filter 指定的条件表达式的内容。代码示例，如下所示。

Python

```
import json
res = client.search_by_text(
    database_name='db-test',
    collection_name='coll-file-test',
    embedding_items=['商标声明'],
    filter='file_name="tcvdb.pdf"',
    limit=2,
)
print(json.dumps(res, ensure_ascii=False))
```

输出信息，如下所示。

[[{"id": "13728601629889167 37", "score": 322.25299, "text": "本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。\n\n【服务声明】\n\n本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。\n\n【联系我们】\n\n我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。\n\n", "file_name": "tcvdb.pdf", "tags": ["向量", "Embedding", "AI"], "author": "Tencent", "chunk_num": 2, "section_num": 1}, {"id": "13728601629889167 36", "score": 312.57782, "text": "# 向量数据库产品简介\n\n【版权声明】\n\n©2013-2023 腾讯云版权所有\n\n本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。\n\n【商标声明】\n\n![vdb-image](51292004-6700-4777-82c3-95275913cef4.png)\n\n及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。", "tags": ["向量", "Embedding", "AI"], "chunk_num": 1, "image_list": ["51292004-6700-4777-82c3-95275913cef4.png"], "file_name": "tcvdb.pdf", "section_num": 1, "author": "Tencent"}]]。

Java

```
SearchByEmbeddingItemsParam searchByEmbeddingItemsParam =
SearchByEmbeddingItemsParam.newBuilder()
        .withEmbeddingItems(Arrays.asList("商标声明"))
        // 若使用 HNSW 索引，则需要指定参数 ef，ef 越大，召回率越高，但也会影
响检索速度
        .withParams(new HNSWSearchParams(200))
        // 设置标量字段的 Filter 表达式，过滤所需查询的文档
        .withFilter("file_name=\"tcvdb.pdf\"")
        .withRetrieveVector(false)
         // 指定 Top K 的 K 值
        .withLimit(2)
        .build();

List<List<Document>> siDocs = client.searchByEmbeddingItems("db-
test", "coll-file-test",
searchByEmbeddingItemsParam).getDocuments();
int i = 0;
for (List<Document> docs : siDocs) {
    System.out.println("\tres: " + i++);
    for (Document doc : docs) {
        System.out.println(doc.toString());
    }
}
```

输出信息，如下所示。

```
res: 0
{"id":"1374654342526382106","score":0.7422837018966675,"text":"本文档
涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体
不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云
及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。\n\n【服务声明】\n\n
本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容
可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的
商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或
保证。\n\n【联系我们】\n\n我们致力于为您提供个性化的售前购买咨询服务，及相应的技
术售后服务，任何问题请联系 4009100100。\n\n","author":"Tencent","tags":
["Embedding","向
量","AI"],"chunk_num":2,"section_num":1,"file_name":"tcvdb.pdf"}
{"id":"1374654342526382105","score":0.7221110463142395,"tags":
["Embedding","向量","AI"],"image_list":["42bad2a3-9313-47e3-b09e-
43c3643c446f.png"],"text":"# 向量数据库产品简介\n\n【版权声明】\n\n©2013-
```

```
2023 腾讯云版权所有\n\n本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。\n\n【商标声明】\n\n![vdb-image](42bad2a3-9313-47e3-b09e-43c3643c446f.png)\n\n及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。","section_num":1,"file_name":"tcvdb.pdf","chunk_num":1,"author":"Tencent"}
---------------------- get chunk text by chunk_num -----------------
-----
```

GO

```go
log.Println("------------------------------ search file chunks by text  ------------------------------")
filter := tcvectordb.NewFilter(`file_name="` + filename + `"`)
searchRes, err := client.SearchByText(ctx, database, collection,
    map[string][]string{"text": {"商标声明"}},
    &tcvectordb.SearchDocumentParams{
        Filter: filter,
        Params:         &tcvectordb.SearchDocParams{Ef: 200}, // 若使用HNSW索引，则需要指定参数ef，ef越大，召回率越高，但也会影响检索速度
        RetrieveVector: false,                                // 是否需要返回向量字段，False：不返回，True：返回
        Limit:          2,                                    // 指定 Top K 的 K 值
})

for i, docs := range searchRes.Documents {
    log.Printf("SearchDocumentResult, index: %d ==================", i)
    for _, doc := range docs {
        log.Printf("SearchDocument: %+v", doc)
    }
}
```

输出信息，如下。

```
2025/05/20 15:00:57 SearchDocumentResult, index: 0
==================
2025/05/20 15:00:57 SearchDocument: {Id:1374280682098630710 Vector:
[] SparseVector:[] Score:323.66516 Fields:map[chunk_num:2
file_name:tcvdb.pdf section_num:1 testInt:1024 testStr:v1 text:本文档
涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体
不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云
及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容
可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的
商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或
保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系
4009100100。

]}
2025/05/20 15:00:57 SearchDocument: {Id:1374280682098630709 Vector:
[] SparseVector:[] Score:312.57782 Fields:map[chunk_num:1
file_name:tcvdb.pdf image_list:[ef173e04-5503-421a-b4e6-
ae414b3e0ce9.png] section_num:1 testInt:1024 testStr:v1 text:# 向量数
据库产品简介

【版权声明】

©2013-2023 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司
单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄
袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措
施追究法律责任。

【商标声明】

![vdb-image](ef173e04-5503-421a-b4e6-ae414b3e0ce9.png)
```

精确查询

根据文件名查询文件

上传文件之后，可根据文件名查询文件。代码示例，如下所示。

Python

```
res = client.query(
    database_name='python-sdk-test-upload',
    collection_name='coll-file-test',
    filter='file_name="tcvdb.pdf"',
    limit=10
)
print(json.dumps(res, ensure_ascii=False))
```

Java

```
System.out.println("---------------------- query -------------------
---");
QueryParam queryParam = QueryParam.newBuilder()
        .withFilter("file_name=\"tcvdb.pdf\"")
        // limit 限制返回行数，1 到 16384 之间
        .withLimit(20)
        // 偏移
        .withOffset(0)
        // 是否返回 vector 数据
        .withRetrieveVector(false)
        .build();
List<Document> qdos = client.query("db-test", "coll-file-test",
queryParam);
for (Document doc : qdos) {
    System.out.println("\tres: " + doc.toString());
```

```
}
```

GO

```
log.Println("------------------------------ Query after waiting 15s
to parse file -------------------------------")
result, _ := client.Query(ctx, database, collection, []string{},
&tcvectordb.QueryDocumentParams{
    Filter: tcvectordb.NewFilter(`file_name="` + filename + `"`),
    Limit:  2,
})
ids := []string{}
for _, doc := range result.Documents {
    log.Printf("File chunk: %+v", doc)
    ids = append(ids, doc.Id)
}
```

获取文件图片 URL

使用 AI 套件上传 PDF 类型的文件后，支持获取图片 URL 地址列表。具体接口，请参见 Python-get_image_url()、Java-GetImageUrl()、Go-GetImageUrl()。

Python

```
res = client.query(
    database_name='python-sdk-test-upload',
    collection_name='coll-file-test',
    filter='file_name="tcvdb.pdf"',
    limit=10
)
print(json.dumps(res, ensure_ascii=False))
```

Java

```
System.out.println("---------------------- get image url -----------
-----------");
GetImageUrlRes getImageUrlRes = client.GetImageUrl("db-test", "coll-
file-test",
        GetImageUrlParam.newBuilder().setFileName("tcvdb.pdf")
                .setDocumentIds(qdos.stream().map(doc-
>doc.getId()).collect(Collectors.toList()))
                .build());
System.out.println("get image url res:");
System.out.println(JsonUtils.toJsonString(getImageUrlRes.getImages()
));
```

GO

```
log.Println("----------------------------- Get file chunks'
imageUrls ------------------------------")
res, err := d.client.GetImageUrl(ctx, database, collection,
tcvectordb.GetImageUrlParams{
    FileName:    filename,
    DocumentIds: ids,
})
if err != nil {
    return err
}
for _, docImages := range res.Images {
    for _, docImage := range docImages {
        log.Printf("docImage: %+v", docImage)
    }
}
```

设置语块序号（chunk_num）范围检索文件

当需要基于当前 chunk_num 值（chunk_num=2）补全其前后语块内容时，可通过 filter 指定条件表达式动态调整查询范围。例如，若需获取相邻 chunk_num 的上下文内容，可设置范围条件为 chunk_num >= {current-2} AND chunk_num <= {current+2}（若当前为3，则查询编号1、2、3 、4、5的语块），从而覆盖目标语块及其关联上下文，确保内容逻辑的连贯性与完整性。具体代码示例，如下所示。

Python

```
doc = res[0][0]
chunk_num = doc.get('chunk_num')
res = client.query(
    database_name='python-sdk-test-upload',
    collection_name='coll-file-test',
    limit=10,
    sort={
        'fieldName': 'chunk_num',
        'direction': 'asc'
    },
    filter=f'chunk_num>={0 if chunk_num-2 <= 0 else chunk_num-2} and
'
           f'chunk_num<={chunk_num+2} and file_name=\"tcvdb.pdf\"',
)
print(json.dumps(res, ensure_ascii=False))
```

Java

```
System.out.println("---------------------- get chunk text by
chunk_num ----------------------");
Long chunkNum = (Long) qdos.get(1).getObject("chunk_num");
Long sectionNum = (Long) qdos.get(1).getObject("section_num");
if (chunkNum==null || sectionNum==null){
    return;
}
Long startChunkNum = chunkNum-2;
if(startChunkNum < 0){
    startChunkNum = 0L;
}

QueryParam queryParam = QueryParam.newBuilder()
        .withFilter("file_name=\"tcvdb.pdf\"  and chunk_num >= "+
startChunkNum +"  and chunk_num <=" + (chunkNum+2))
        // limit 限制返回行数，1 到 16384 之间
        .withLimit(10)
```

```
        // 输出按照 chunk_num 排序

.withSort(OrderRule.newBuilder().withFieldName("chunk_num").withDire
ction(OrderEnum.DESC).build())
        // 偏移
        .withOffset(0)
        // 是否返回 vector 数据
        .withRetrieveVector(false)
        .build();
// 输出相似性检索结果，检索结果为二维数组，每一位为一组返回结果，分别对应 search
时指定的多个向量
List<Document> docs = client.query("db-test", "coll-file-test",
queryParam);
for (Document doc : docs) {
    System.out.println("\tres: " + doc.toString());
}
```

Go

```
log.Println("------------------------------ Query file neighbor
chunks ------------------------------")
if len(result.Documents) != 2 {
    return nil
}
chunkDoc := result.Documents[1]
if _, ok := chunkDoc.Fields["chunk_num"]; !ok ||
chunkDoc.Fields["chunk_num"].Type() != tcvectordb.Uint64 {
    return nil
}

chunkNum := chunkDoc.Fields["chunk_num"].Uint64()
leftChunkNum := uint64(0)
if chunkNum >= 2 {
    leftChunkNum = chunkNum - 2
}
rightChunkNum := chunkNum + 2

result, err = client.Query(ctx, database, collection, []string{},
&tcvectordb.QueryDocumentParams{
```

```
    Filter: tcvectordb.NewFilter(`file_name="` + filename +
`"`).And(`chunk_num>=` +
        strconv.Itoa(int(leftChunkNum)) + ` and chunk_num<=` +
strconv.Itoa(int(rightChunkNum))),
    Limit: 10,
    Sort: []document.SortRule{
        {
                FieldName: "chunk_num",
                Direction: "asc",
        },
    },
})
if err != nil {
    return err
}

for _, doc := range result.Documents {
    log.Printf("File expand chunk: %+v", doc)
}
```

输出信息，以 Go SDK 为例，如下所示。

```
2025/05/22 11:12:40 ------------------------------ Query file neighbor
chunks ------------------------------
2025/05/22 11:12:40 File expand chunk: {Id:1374948009337135161 Vector:[]
SparseVector:[] Score:0 Fields:map[chunk_num:5 file_name:tcvdb.pdf
section_num:4 testInt:1024 testStr:v1 text:### 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。

]}
2025/05/22 11:12:40 File expand chunk: {Id:1374948009337135162 Vector:[]
SparseVector:[] Score:0 Fields:map[chunk_num:6 file_name:tcvdb.pdf
section_num:5 testInt:1024 testStr:v1 text:### 关键概念

如果您不熟悉向量数据库和相似性搜索领域，请优先阅读以下基本概念，便于您对向量数据库有一
个初步的了解。更多名词解释，请阅读 关键概念。
```

```
]}
2025/05/22 11:12:40 File expand chunk: {Id:1374948009337135163 Vector:[]
SparseVector:[] Score:0 Fields:map[chunk_num:7 file_name:tcvdb.pdf
section_num:6 testInt:1024 testStr:v1 text:#### 什么是向量?

向量是指在数学和物理中用来表示大小和方向的量。它由一组有序的数值组成，这些数值代表了向
量在每个坐标轴上的分量。

]}
2025/05/22 11:12:40 File expand chunk: {Id:1374948009337135164 Vector:[]
SparseVector:[] Score:0 Fields:map[chunk_num:8 file_name:tcvdb.pdf
section_num:7 testInt:1024 testStr:v1 text:#### 什么是非结构化数据?

非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模
型或组织方式，通常更难以处理和分析。

]}
2025/05/22 11:12:40 File expand chunk: {Id:1374948009337135165 Vector:[]
SparseVector:[] Score:0 Fields:map[chunk_num:9 file_name:tcvdb.pdf
section_num:8 testInt:1024 testStr:v1 text:#### 什么是 AI 中的向量表示?

当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非
结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾
讯云向量数据库提供的模型能力，目前在开发调试中。具体上线时间，请关注 产品动态。

]}
```

设置段落序号（section_num）检索文件

基于段落序号（section_num）和语块编号（chunk_num），可以精准定位文件内容。例如，若需提取某段落（如 section_num=5）的前五个语块（chunk_num 范围为1至4），可通过 filter 条件表达式设置 section_num = {目标值} AND chunk_num BETWEEN 1 AND 4，限定查询范围在目标段落内，避免跨段数据的干扰。具体示例，如下所示。

Python

```python
section_num = res[1].get('section_num')
res = client.query(
    database_name='python-sdk-test-upload',
    collection_name='coll-file-test',
    limit=10,
```

```
    sort={
        'fieldName': 'chunk_num',
        'direction': 'asc',
    },
    filter=f'chunk_num>={0 if chunk_num-2 <= 0 else chunk_num-2} and
'
           f'chunk_num<={chunk_num+2} and section_num={section_num}
and file_name=\"tcvdb.pdf\"',
)
print(json.dumps(res, ensure_ascii=False))

vdb_client.close()
```

Java

```
System.out.println("---------------------- get chunk text by
chunk_num ----------------------");
Long chunkNum = (Long) qdos.get(1).getObject("chunk_num");
Long sectionNum = (Long) qdos.get(1).getObject("section_num");
if (chunkNum==null || sectionNum==null){
    return;
}
Long startChunkNum = chunkNum-2;
if(startChunkNum < 0){
    startChunkNum = 0L;
}

QueryParam queryParam = QueryParam.newBuilder()
        .withFilter("file_name=\"tcvdb.pdf\"  and chunk_num >= "+
startChunkNum +"  and chunk_num <=" + (chunkNum+2) + " and
section_num=" + sectionNum)
        // limit 限制返回行数，1 到 16384 之间
        .withLimit(20)
        // 输出按照 chunk_num 排序

.withSort(OrderRule.newBuilder().withFieldName("chunk_num").withDire
ction(OrderEnum.DESC).build())
        // 偏移
        .withOffset(0)
        // 是否返回 vector 数据
```

```
        .withRetrieveVector(false)
        .build();
// 输出相似性检索结果，检索结果为二维数组，每一位为一组返回结果，分别对应 search
时指定的多个向量
List<Document> docs = client.query("db-test", "coll-file-test",
queryParam);
for (Document doc : docs) {
    System.out.println("\tres: " + doc.toString());
}
```

Go

```
log.Println("------------------------------ Query file neighbor
chunks  with same section ------------------------------")
if _, ok := chunkDoc.Fields["section_num"]; !ok ||
chunkDoc.Fields["section_num"].Type() != tcvectordb.Uint64 {
    return nil
}
sectionNum := chunkDoc.Fields["section_num"].Uint64()
result, err = client.Query(ctx, database, collection, []string{},
&tcvectordb.QueryDocumentParams{
    Filter: tcvectordb.NewFilter(`file_name="` + filename +
`"`).And(`chunk_num>=` +
        strconv.Itoa(int(leftChunkNum)) + ` and chunk_num<=` +
strconv.Itoa(int(rightChunkNum))).And(
        `section_num=` + strconv.Itoa(int(sectionNum))),
    Limit: 10,
    Sort: []document.SortRule{
        {
            FieldName: "chunk_num",
            Direction: "asc",
        },
    },
})
if err != nil {
    return err
}
for _, doc := range result.Documents {
    log.Printf("File expand chunk with same section: %+v", doc)
```

```
}
```

Java SDK 输出示例，如下所示。

```
---------------------- get chunk text by chunk_num ---------------------
-
    res:
{"id":"1374654342526382105","score":0.0,"file_name":"tcvdb.pdf","chunk_n
um":1,"text":"# 向量数据库产品简介\n\n【版权声明】\n\n©2013-2023 腾讯云版权所有
\n\n本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司
单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传
播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律
责任。\n\n【商标声明】\n\n![vdb-image](42bad2a3-9313-47e3-b09e-
43c3643c446f.png)\n\n及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及
其关联公司所有。","author":"Tencent","section_num":1,"tags":
["Embedding","向量","AI"],"image_list":["42bad2a3-9313-47e3-b09e-
43c3643c446f.png"]}
    res: {"id":"1374654342526382106","score":0.0,"text":"本文档涉及的第三方
主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对
前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的
侵犯，腾讯云将依法采取措施追究法律责任。\n\n【服务声明】\n\n本文档意在向您介绍腾讯云
全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的
腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，
否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。\n\n【联系我们】\n\n我们致力
于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系
4009100100。\n\n","chunk_num":2,"tags":["Embedding","向
量","AI"],"author":"Tencent","section_num":1,"file_name":"tcvdb.pdf"}
```

基于 AI 套件快速导入文件并检索

最近更新时间：2025-04-11 09:59:51

腾讯云向量数据库（Tencent Cloud VectorDB）AI 套件 功能支持直接写入 Markdown 文件。本章节介绍如何应用 AI 套件上传文件写入数据，并基于输入的文本信息对文件内容进行相似性检索的方法。以 Linux 操作系统为例，使用 Python、Java、Go SDK 示例代码分别演示。运行本章节所提供的示例代码，您将初步了解 AI 套件一站式文档检索的解决方案。

导入 SDK 依赖模块

Python

```
import tcvectordb
from tcvectordb.model.ai_database import AIDatabase
from tcvectordb.model.collection_view import Embedding,
SplitterProcess, Language, CollectionView
from tcvectordb.model.document import Filter, Document
from tcvectordb.model.document_set import DocumentSet
from tcvectordb.model.enum import FieldType, IndexType,
ReadConsistency
from tcvectordb.model.index import Index, FilterIndex
```

Java

```
import com.tencent.tcvectordb.client.RPCVectorDBClient;
import com.tencent.tcvectordb.model.*;
```

Go

```
package main

import (
```

```
    "context"
    "time"
    "log"
    "github.com/tencent/vectordatabase-sdk-go/tcvectordb"
)
```

创建 Client

导入 SDK 所需的模块之后，需先创建一个向量数据库的客户端对象，与向量数据库服务器连接才能进行交互。

> **说明：**
> 如下示例 **url** 与 **key** 需要分别替换为已购买实例的 外网访问地址 与 API Key。请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**外网地址**，在**密钥管理**页面直接复制密钥。

Python

```
#create a database client object
client = tcvectordb.RPCVectorDBClient(url='http://10.0.X.X',
username='root', key='eC4bLRy2va******************************',
read_consistency=ReadConsistency.EVENTUAL_CONSISTENCY, timeout=30)
```

Java

```
public class VectorDBExample {
    public static void main(String[] args) {
        // 创建VectorDB Client
        ConnectParam connectParam = ConnectParam.newBuilder()
                .withUrl("http://10.0.X.X:80")
                .withUsername("root")
                .withKey("eC4bLRy2va******************************")
                .withTimeout(30)
                .build();
        VectorDBClient client = new
RPCVectorDBClient(connectParam,ReadConsistencyEnum.EVENTUAL_CONSISTE
NCY);
```

```
    }
}
```

Go

```
func main() {
    // 初始化客户端
    var defaultOption = &tcvectordb.ClientOption{
        Timeout:             time.Second * 5,
        MaxIdldConnPerHost: 2,
        IdleConnTimeout:    time.Minute,
        ReadConsistency:    tcvectordb.EventualConsistency,
    }
    client, err := tcvectordb.NewRpcClient("http://10.0.X.X:80",
"root", "eC4bLRy2va*****************************", defaultOption)
    if err != nil {
        panic(err)
    }
}
```

创建数据库

基于已创建的客户端对象，创建数据库 **db-test-ai**。

Python

Python SDK 支持通过 create_ai_database() 接口创建数据库。

```
db = client.create_ai_database(database_name='db-test-ai')
```

Java

Java SDK 支持通过 createAIDatabase() 接口创建数据库。

```
AIDatabase db = client.createAIDatabase("db-test-ai");
```

Go

Go SDK 支持通过 CreateAIDatabase() 接口创建数据库。

```
var (
    ctx                 = context.Background()
    aiDatabase          = "db-test-ai"
)
db, _ := client.CreateAIDatabase(context.Background(), aiDatabase)
log.Printf("Create AI database success, %s", db.DatabaseName)
```

创建集合视图

创建 CollectionView 之前，需要针对预上传的文件数据选取可作为 Filter 索引的标量字段，以便使用该字段的 Filter 条件表达式过滤查找文件。通常选取文件的 Metadata 信息字段。如下示例，预以文件的作者字段为 Filter 索引，将字段名 **author** 设置 Filter 索引。

Python

Python SDK 通过 create_collection_view() 创建一个名为 **coll-ai-files** 集合视图。

```
# 第一步：设计索引，为文件 meta 信息标量字段 author 配置 Filter 索引
index = Index()
index.add(FilterIndex('author', FieldType.String, IndexType.FILTER))
# 第二步：创建集合视图
coll_view = db.create_collection_view(name='coll-ai-files',
                                      description='This is a collectionView',
                                      index=index)
print(vars(coll_view))
```

Java

Java SDK 通过 createCollectionView() 创建一个名为 coll-ai-files 集合视图。

```
// link database, client 为 VectorDBClient() 创建的客户端对象
AIDatabase db = client.aiDatabase("db-test-ai");
// 初始化 ColletionView 参数
CreateCollectionViewParam collectionParam =
CreateCollectionViewParam.newBuilder()
                .withName("coll-ai-files")
                .withDescription("This is a collectionView")
                .addField(new FilterIndex("author",
FieldType.String, IndexType.FILTER))
                .build();
// Create CollectionView
CollectionView coll = db.createCollectionView(collectionParam);
```

Go

Go SDK 通过 CreateCollectionView() 创建一个名为 coll-ai-files 集合视图。

```
var (
    ctx                = context.Background()
    aiDatabase         = "db-test-ai"
    collectionViewName = "coll-ai-files"
)
// 第一步：设计索引，为文件 meta 信息标量字段 author 配置 Filter 索引
index := tcvectordb.Indexes{
    FilterIndex: []tcvectordb.FilterIndex{
        {
            FieldName: "author",
            FieldType: tcvectordb.String,
            IndexType: tcvectordb.FILTER,
        },
    },
}
// 第二步：创建集合视图
coll, _ := db.CreateCollectionView(ctx, collectionViewName,
tcvectordb.CreateCollectionViewParams{
    Description: "This is a collectionView",
    Indexes:     index,
```

```
})
log.Printf("CreateCollectionView success: %v: %v",
coll.DatabaseName, coll.CollectionViewName)
```

上传文件

集合视图创建完成之后，便可指定上传文件所在路径，上传文件于数据库中。如下示例，文件路径为 `/tmp/腾讯云向量数据库.pdf` ，将该文件内容及其向量数据存储于集合视图 coll-ai-files 中。

> **说明：**
> 如果需要进行腾讯云向量数据库的测试，请获取 腾讯云向量数据库.pdf 文档。

Python

Python SDK 通过 load_and_split_text() 接口上传文件 `腾讯云向量数据库.pdf` 于集合 coll-ai-files。

```
# 上传文件
# 1. 指定文件在客户端的存放路径
# 2. 自定义文件meta数据
res = coll_view.load_and_split_text(local_file_path="/tmp/腾讯云向量数
据库.pdf",
            metadata={
            'author': 'Tencent'
        }
    )
```

Java

Java SDK 通过 loadAndSplitText() 接口上传文件 `腾讯云向量数据库.pdf` 于集合 coll-ai-files。

```
// link database, client 为 VectorDBClient() 创建的客户端对象
AIDatabase db = client.aiDatabase("db-test-ai");
// link collectionView
CollectionView collection = db.describeCollectionView("coll-ai-
files");
// LocalFilePath 配置上传文件的本地路径
```

```
// DocumentSetName 指定文件存储于数据库的名称
LoadAndSplitTextParam param = LoadAndSplitTextParam.newBuilder()
                .withLocalFilePath("/tmp/腾讯云向量数据库.pdf")
                .Build();
// 配置文件 Meatdata 标量字段的值
Map<String, Object> metaDataMap = new HashMap<>();
metaDataMap.put("author", "Tencent");
// 调用 loadAndSplitText() 上传文件
collection.loadAndSplitText(param, metaDataMap);
```

Go

Go SDK 通过 LoadAndSplitText() 接口上传文件 `腾讯云向量数据库.pdf` 于集合 coll-ai-files。

```
metaData := map[string]interface{}{
        "author": "Tencent",
    }
UpsertResult, _ := coll.LoadAndSplitText(ctx,
tcvectordb.LoadAndSplitTextParams{
    LocalFilePath: "/tmp/腾讯云向量数据库.pdf",
    MetaData:      metaData,
})
log.Printf("LoadAndSplitText success: %+v", UpsertResult)
```

查询文件

上传文件可能存在延迟，查询文件，以便确认文件已在后台解析完成。如下示例，指定文件名，查询存储于数据库中的文件状态。返回参数 indexedStatus 将显示文件预处理、Embedding 向量化的状态。

- **New**：等待解析。
- **Loading**：文件解析中。
- **Failure**：文件解析、写入出错。
- **Ready**：文件解析、写入完成。

Python

Python SDK 通过 get_document_set() 获取文件当前状态。

```
res = coll_view.get_document_set(document_set_name="腾讯云向量数据
库.pdf")
print(vars(res))
```

Java

Java SDK 通过 getFile() 获取文件当前状态。

```
String fileId="";
String fileName = "腾讯云向量数据库.pdf";
System.out.println(collection.getFile(fileName,fileId).toString());
```

Go

Go SDK 通过 GetDocumentSetByName() 获取文件当前状态。

```
result, _ := coll.GetDocumentSetByName(ctx, "腾讯云向量数据库.pdf")
log.Printf("GetDocumentSetByName success: %+v", result)
```

相似性内容检索

确认文件已解析完成之后，便可开始进行相似性内容检索。如下示例，检索信息 `什么是向量数据库` ，默认返回相似度最高的信息。

Python

Python SDK 通过 search() 接口进行相似性检索。

```
# content 指定需检索的文本内容
# document_set_name 指定检索的文件名
doc_list = coll_view.search(
            content='向量是指什么',
            document_set_name = ['腾讯云向量数据库.pdf']
        )
```

```
for doc in doc_list:
        print(vars(doc))
```

检索结果，如下所示。

关键信息	子参数	参数含义
score	-	• 表示查询向量与检索结果向量之间的相似性计算分数。 • 输出结果按照相似程度得分由高到低逐一排列。
data	text	检索到的结果。


```
{
  'score': 0.8473929762840271,
  'data': {
    'text': '### 什么是向量？\n向量是指在数学和物理中用来表示大小和方向的量。它由一组有序的数值组成，这些数值代表了向量在每个坐标轴上的分量。\n',
    'startPos': 441,
    'endPos': 508,
    'pre': [
      '## 关键概念\n如果您不熟悉向量数据库和相似性搜索领域，请优先阅读以下基本概念，便于您对向量数据库有一个初步的了解。\n'
    ],
    'next': [
      '### 什么是非结构化数据？\n非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。\n'
    ]
  },
  'documentSet': {
    'documentSetId': '1192727327592550400',
    'documentSetName': '腾讯云向量数据库.pdf',
    'author': 'Tencent'
  }
}{
  'score': 0.7868989706039429,
  'data': {
    'text': '### 什么是 AI 中的向量表示？\n当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。\n',
    'startPos': 585,
    'endPos': 784,
    'pre': [
```

```
        '### 什么是非结构化数据？\n非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。\n'
      ],
      'next': [
        '### 什么是向量检索？\n向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。\n'
      ]
    },
    'documentSet': {
      'documentSetId': '1192727327592550400',
      'documentSetName': '腾讯云向量数据库.pdf',
      'author': 'Tencent'
    }
  }{
    'score': 0.7792050242424011,
    'data': {
      'text': '### 什么是向量检索？\n向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。\n',
      'startPos': 784,
      'endPos': 876,
      'pre': [
        '### 什么是 AI 中的向量表示？\n当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。\n'
      ],
      'next': [
        '## 为什么是腾讯云向量数据库？\n腾讯云向量数据库作为一种专门存储和检索向量数据的服务提供给用户， 在高性能、高可用、大规模、低成本、简单易用、稳定可靠、智能运维等方面体现出显著优势。 \n'
      ]
    },
    'documentSet': {
      'documentSetId': '1192727327592550400',
      'documentSetName': '腾讯云向量数据库.pdf',
      'author': 'Tencent'
    }
  }
```

Java

Java SDK 通过 search() 接口进行相似性检索。

```
// link database， client 为 VectorDBClient() 创建的客户端对象
AIDatabase db = client.aiDatabase("db-test-ai");
// link collectionView
CollectionView collection = db.describeCollectionView("coll-ai-
files");
// 设置查询参数
// Content 配置需检索的文本信息
// DocumentSetNames 指定需要查找的文件名，可批量设置
SearchByContentsParam searchByContentsParam =
SearchByContentsParam.newBuilder()
                .withContent("向量是指什么")
                .withDocumentSetNames(Arrays.asList("腾讯云向量数据
库.pdf"))
                .build();
// 根据配置的查询条件进行内容检索
List<Document> searchRes = collection.search(searchByContentsParam);
// 输出检索结果
int i = 0;
for (Document doc : searchRes) {
    System.out.println("\tres" +(i++)+": "+ doc.toString());
}
```

检索结果，如下所示。

关键信息	子参数	参数含义
score	-	• 表示查询向量与检索结果向量之间的相似性计算分数。 • 输出结果按照相似程度得分由高到低逐一排列。
data	text	检索到的结果。

```
res0: {
  "score": 0.8938464522361755,
  "data": {
    "text": "### 什么是向量？\n向量是指在数学和物理中用来表示大小和方向的量。
它由一组有序的数值组成，这些数值代表了向量在每个坐标轴上的分量。\n",
    "endPos": 508,
```

```
      "startPos": 441,
      "next": [
        "### 什么是非结构化数据？\n非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。\n"
      ],
      "pre": [
        "## 关键概念\n如果您不熟悉向量数据库和相似性搜索领域，请优先阅读以下基本概念，便于您对向量数据库有一个初步的了解。\n"
      ]
    },
    "documentSet": {
      "documentSetName": "\"腾讯云向量数据库.pdf\"",
      "documentSetId": "\"1182525034158882816\"",
      "docFields": [
        {
          "name": "author",
          "value": "Tencent"
        },
        {
          "name": "tags",
          "value": [
            "Embedding",
            "向量",
            "AI"
          ]
        }
      ]
    }
  }res1: {
    "score": 0.8378515839576721,
    "data": {
      "text": "### 什么是 AI 中的向量表示？\n当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。\n",
      "endPos": 784,
      "startPos": 585,
      "next": [
        "### 什么是向量检索？\n向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。\n"
      ],
      "pre": [
```

```
      "### 什么是非结构化数据？\n非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。\n"
    ]
  },
  "documentSet": {
    "documentSetName": "\"腾讯云向量数据库.pdf\"",
    "documentSetId": "\"1182525034158882816\"",
    "docFields": [
      {
        "name": "author",
        "value": "Tencent"
      },
      {
        "name": "tags",
        "value": [
          "Embedding",
          "向量",
          "AI"
        ]
      }
    ]
  }
}res2: {
  "score": 0.8152828216552734,
  "data": {
    "text": "### 什么是向量检索？\n向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。\n",
    "endPos": 876,
    "startPos": 784,
    "next": [
      "## 为什么是腾讯云向量数据库？\n腾讯云向量数据库作为一种专门存储和检索向量数据的服务提供给用户， 在高性能、高可用、大规模、低成本、简单易用、稳定可靠、智能运维等方面体现出显著优势。 \n"
    ],
    "pre": [
      "### 什么是 AI 中的向量表示？\n当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。\n"
    ]
  },
  "documentSet": {
```

```
    "documentSetName": "\"腾讯云向量数据库.pdf\"",
    "documentSetId": "\"1182525034158882816\"",
    "docFields": [
      {
        "name": "author",
        "value": "Tencent"
      },
      {
        "name": "tags",
        "value": [
          "Embedding",
          "向量",
          "AI"
        ]
      }
    ]
  }
}
```

Go

Go SDK 通过 Search() 接口进行相似性检索。

```
// 等待文件解析完成
time.Sleep(time.Second * 30)
searchRes, _ := coll.Search(ctx,
tcvectordb.SearchAIDocumentSetsParams{
    Content: "向量是指什么",
    DocumentSetName: []string{"腾讯云向量数据库.pdf"},
})
for _, doc := range searchRes.Documents {
    log.Printf("document: %+v", doc)
}
```

检索结果，如下所示。

关键信息	子参数	参数含义
score	-	• 表示查询向量与检索结果向量之间的相似性计算分数。 • 输出结果按照相似程度得分由高到低逐一排列。

data	text	检索到的结果。

2024/01/05 16:18:22 document: {DatabaseName:db-test-ai CollectionViewName:coll-ai-files DocumentSetId:1192743758593921024 DocumentSetName:腾讯云向量数据库.pdf Score:0.8473929762840271 SearchData:{Text:### 什么是向量?
向量是指在数学和物理中用来表示大小和方向的量。它由一组有序的数值组成，这些数值代表了向量在每个坐标轴上的分量。
StartPos:441 EndPos:508 Pre:[## 关键概念
如果您不熟悉向量数据库和相似性搜索领域，请优先阅读以下基本概念，便于您对向量数据库有一个初步的了解。
] Next:[### 什么是非结构化数据?
非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。
]} ScalarFields:map[author:Tencent]}
2024/01/05 16:18:22 document: {DatabaseName:db-test-ai CollectionViewName:coll-ai-files DocumentSetId:1192743758593921024 DocumentSetName:腾讯云向量数据库.pdf Score:0.7868989706039429 SearchData:{Text:### 什么是 AI 中的向量表示?
当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。
StartPos:585 EndPos:784 Pre:[### 什么是非结构化数据?
非结构化数据，是指图像、文本、音频等数据。与结构化数据相比，非结构化数据不遵循预定义模型或组织方式，通常更难以处理和分析。
] Next:[### 什么是向量检索?
向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。
]} ScalarFields:map[author:Tencent]}
2024/01/05 16:18:22 document: {DatabaseName:db-test-ai CollectionViewName:coll-ai-files DocumentSetId:1192743758593921024 DocumentSetName:腾讯云向量数据库.pdf Score:0.7792050242424011 SearchData:{Text:### 什么是向量检索?
向量检索是一种基于向量空间模型的信息检索方法。将非结构化的数据表示为向量存入向量数据库，向量检索通过计算查询向量与数据库中存储的向量的相似度来找到目标向量。
StartPos:784 EndPos:876 Pre:[### 什么是 AI 中的向量表示?
当我们处理非结构化数据时，需要将其转换为计算机可以理解和处理的形式。向量表示是一种将非结构化数据转换为嵌入向量的技术，通过多维度向量数值表述某个对象或事物的属性或者特征。腾讯云向量数据库提供的模型能力，目前在开发调试中。
] Next:[## 为什么是腾讯云向量数据库?

腾讯云向量数据库作为一种专门存储和检索向量数据的服务提供给用户， 在高性能、高可用、大规模、低成本、简单易用、稳定可靠、智能运维等方面体现出显著优势。

删除数据库

Python

Python SDK 通过 drop_ai_database() 接口删除数据库 db-test-ai。

```
client.drop_ai_database(database_name='db-test-ai')
```

Java

Java SDK 通过 dropAIDatabase() 删除数据库 db-test-ai。

```
client.dropAIDatabase("db-test-ai");
```

Go

Go SDK 通过 DropAIDatabase() 删除数据库 db-test-ai。

```
result, _ := client.DropAIDatabase(context.Background(), aiDatabase)
```