> 目前大家已基本上都已经训练完表示模型，接下来要做的就是把数据库中的论文标题进行**向量化**，并保存到数据库中。这里我们通常使用**向量数据库**，我以 Milvus（https://milvus.io/docs）为例进行介绍。
>
> 实现 Milvus 服务包括两步：
>
> - 在服务器上用 docker 进行服务部署
> - 写 python 代码将向量上传
>
> 我统一是用的是 v2.0.x 版本，大家如果打算使用 v1.0.x 版本，就直接按照官网教程，不要参考下面的内容。

Docker 部署

- 按照教程 https://milvus.io/docs/v2.0.x/install_standalone-docker.md 把 Milvus 服务跑起来
- 为了方便管理，可以再运行一个图形化界面的服务 Attu（https://github.com/zilliztech/attu），如果是 v1 的话可以用这个（https://zilliz.com/products/em）
- Milvus 服务的默认端口为 19530，我现在已经把大家 19530 端口的访问权限打开了，如果你们不用 Milvus，打算用其他的工具，也可以联系我开新的端口。

代码

以下为部分 Milvus2.0 的参考代码，v1.0.x 和 v2.0.x 差别较大，大家不要弄混淆了。另外，我提供的只是参考，各位一定要仔细阅读官方 API 文档。

```
# milvus_utils.py
from pymilvus import connections
from pymilvus import CollectionSchema, FieldSchema, DataType, Collection

_HOST = '127.0.0.1'
_PORT = '19530'

def get_milvus_connection():
    '''建立Milvus连接'''
    connections.connect(
      alias="default", host=_HOST, port=_PORT
)

def create_milvus_collection():
    '''创建 collection'''
    milvus_id = FieldSchema(
      name="milvus_id",
      dtype=DataType.INT64,
      is_primary=True,
      auto_id=True
    )
    vector = FieldSchema(
      name="vector",
      dtype=DataType.FLOAT_VECTOR,
```

```
      dim=256
    )
    schema = CollectionSchema(
      fields=[milvus_id, vector],
      description="O2E"
    )
    collection_name = "O2E"
    collection = Collection(
        name=collection_name,
        schema=schema,
        using='default',
        # shards_num=2,
        # consistency_level="Strong"
    )


def get_milvus_collection(name):
    '''获取指定collection'''
    collection = Collection(name)
    return collection

def milvus_insert(collection_name, data, partition_name=None):
    '''
    插入数据。
    param: collection_name: collection名称
    param: partition_name: partition名称
    param: data: 待插入的数据，list-like(list, tuple)
    return: Milvus生成的ids
    '''
    collection = get_milvus_collection(collection_name)
    res = collection.insert(partition_name=partition_name, data=data)
    ids = res.primary_keys # 这个id是由Milvus生成的，大家注意要和论文id对应起来保存
    return ids


def milvus_search(collection_name, partition_names, query_vectors, topk, expr=None):
    '''
    查询相关向量。
    param: collection_name: collection名称
    param: partition_names: partition名称列表
    param: query_vectors: 待查询的数据，list-like(list, tuple)
    param: topk: 每个query返回的最相似个数
    param: expr: 条件表达式
    return: ids_list, (list,list)
    '''
    collection = get_milvus_collection(collection_name)
    res = collection.search(
        partition_names=partition_names,
        data=query_vectors,
```

```
        anns_field="vector",
        limit=topk,
        expr=expr,
        param={"metric_type": "L2", "params": {"nprobe": 10}}
    )
    ids_list = []
    for item in res:
        ids = []
        for p in item:
            ids.append(p.id)
        ids_list.append(ids)
    return ids_list


def milvus_get_by_id(collection_name, id):
    '''根据id获取数据'''
    collection = get_milvus_collection(collection_name)
    res = collection.query("milvus_id == {}".format(id))
    for item in res:
        print(item)


if __name__ == '__main__':
    get_milvus_connection()
    collection = get_milvus_collection("O2E")
    print(collection._schema)
```

> 在完成向量化之后，大家就可以拿到关键词之后将其向量化，然后在 Milvus 里面召回成果，再将成果和专家匹配进行专家排序。
>
> 你们可以先测试一下这种方式的耗时是否可以接受（应该没问题），比较一下 10 万和 100 万级数据量下的时间差异。
>
> 如果实在不行，那我们之后可以提前从论文中抽取关键词，把关键词和论文的对应关系保存到 ES 里面，进一步加速查询。