In [ ]:
```
%pip install -r requirements.txt
```

In [ ]:
```
#Connectings to Milvus and Redis
import redis
from pymilvus import connections, DataType, FieldSchema, CollectionSchema, 

connections.connect(host = '127.0.0.1', port = 19530)
red = redis.Redis(host = '127.0.0.1', port=6379, db=0)
```

In [ ]:
```
#Creating collection

import time

red.flushdb()
time.sleep(.1)
collection_name = "audio_test_collection"

if utility.has_collection(collection_name):
    print("Dropping existing collection...")
    collection = Collection(name=collection_name)
    collection.drop()

#if not utility.has_collection(collection_name):
field1 = FieldSchema(name="id", dtype=DataType.INT64, descrition="int64", i
field2 = FieldSchema(name="embedding", dtype=DataType.FLOAT_VECTOR, descrit
schema = CollectionSchema(fields=[ field1,field2], description="collection 
collection = Collection(name=collection_name, schema=schema)
print("Created new collection with name: " + collection_name)
```

```
Dropping existing collection...
Created new collection with name: audio_test_collection
```

In [ ]:
```
#Indexing collection

if utility.has_collection(collection_name):
    collection = Collection(name = collection_name)
default_index = {"index_type": "IVF_SQ8", "metric_type": "L2", "params": {"
status = collection.create_index(field_name = "embedding", index_params = d
if not status.code:
    print("Successfully create index in collection:{} with param:{}".format
```

```
Successfully create index in collection:audio_test_collection with param:
{'index_type': 'IVF_SQ8', 'metric_type': 'L2', 'params': {'nlist': 16384}}
```

Download audio dataset from CORTX S3

In [ ]:
```
import boto3

# create bucket named the current date
ACCESS_KEY = 'sgiamadmin'
SECRET_ACCESS_KEY = 'ldapadmin'
END_POINT_URL = 'http://192.168.1.14:31949'

s3_client = boto3.client('s3', endpoint_url=END_POINT_URL,
                         aws_access_key_id=ACCESS_KEY,
                         aws_secret_access_key=SECRET_ACCESS_KEY,
                         verify=False)

s3_resource = boto3.resource('s3', endpoint_url=END_POINT_URL,
                             aws_access_key_id=ACCESS_KEY,
```

```
                                aws_secret_access_key=SECRET_ACCESS_KEY,
                                region_name='None',
                                verify=False)
```

In [ ]:
```
buckets = s3_client.list_buckets()

if buckets['Buckets']:
    for bucket in buckets['Buckets']:
        print(bucket)
```

```
{'Name': 'datasets', 'CreationDate': datetime.datetime(2022, 7, 4, 3, 7, 3
6, 664000, tzinfo=tzutc())}
{'Name': 'milvus', 'CreationDate': datetime.datetime(2022, 7, 3, 12, 14, 2
1, 955000, tzinfo=tzutc())}
{'Name': 'mybucket', 'CreationDate': datetime.datetime(2022, 6, 30, 13, 1,
26, 77000, tzinfo=tzutc())}
```

In [ ]:
```
s3_resource.Bucket('datasets').download_file('example_audio.zip', 'example_
```

In [ ]:
```
import zipfile

with zipfile.ZipFile("example_audio.zip","r") as zip_ref:
    zip_ref.extractall("./example_audio")
```

In [ ]:
```
import os
import librosa
import gdown
import zipfile
import numpy as np
from panns_inference import SoundEventDetection, labels, AudioTagging

data_dir = './example_audio'
at = AudioTagging(checkpoint_path=None, device='cpu')

def embed_and_save(path, at):

    audio, _ = librosa.core.load(path, sr=32000, mono=True)
    audio = audio[None, :]
    try:
        _, embedding = at.inference(audio)
        embedding = embedding/np.linalg.norm(embedding)
        embedding = embedding.tolist()[0]
        mr = collection.insert([[embedding]])
        ids = mr.primary_keys
        collection.load()
        red.set(str(ids[0]), path)
    except Exception as e:
        print("failed: " + path + "; error {}".format(e))


print("Starting Insert")
for subdir, dirs, files in os.walk(data_dir):
    for file in files:
        path = os.path.join(subdir, file)
        embed_and_save(path, at)
print("Insert Done")
```

```
Checkpoint path: /home/sumit/panns_data/Cnn14_mAP=0.431.pth
/home/sumit/.local/lib/python3.8/site-packages/torchlibrosa/stft.py:193: Fu
tureWarning: Pass size=1024 as keyword args. From version 0.10 passing thes
e as positional arguments will result in an error
  fft_window = librosa.util.pad_center(fft_window, n_fft)
```

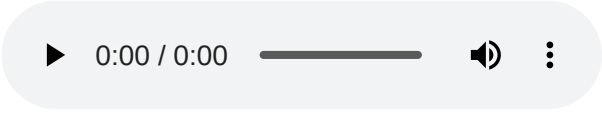
```
Using CPU.
Starting Insert
Insert Done
```

In [ ]:
```python
def get_embed(paths, at):
    embedding_list = []
    for x in paths:
        audio, _ = librosa.core.load(x, sr=32000, mono=True)
        audio = audio[None, :]
        try:
            _, embedding = at.inference(audio)
            embedding = embedding/np.linalg.norm(embedding)
            embedding_list.append(embedding)
        except:
            print("Embedding Failed: " + x)
    return np.array(embedding_list, dtype=np.float32).squeeze()

random_ids = [int(red.randomkey()) for x in range(2)]
search_clips = [x.decode("utf-8") for x in red.mget(random_ids)]
embeddings = get_embed(search_clips, at)
print(embeddings.shape)
```

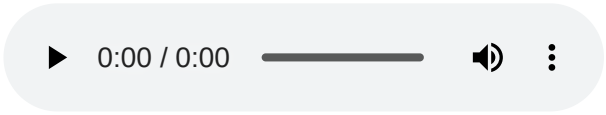
```
(2, 2048)
```

In [ ]:
```python
import IPython.display as ipd

def show_results(query, results, distances):
    print("Query: ")
    ipd.display(ipd.Audio(query))
    print("Results: ")
    for x in range(len(results)):
        print("Distance: " + str(distances[x]))
        ipd.display(ipd.Audio(results[x]))
    print("-"*50)

embeddings_list = embeddings.tolist()

search_params = {"metric_type": "L2", "params": {"nprobe": 16}}


try:
    start = time.time()
    results = collection.search(embeddings_list, anns_field="embedding", pa
    end = time.time() - start
    print("Search took a total of: ", end)
    for x in range(len(results)):
        query_file = search_clips[x]
        result_files = [red.get(y.id).decode('utf-8') for y in results[x]]
        distances = [y.distance for y in results[x]]
        show_results(query_file, result_files, distances)
except Exception as e:
    print("Failed to search vectors in Milvus: {}".format(e))
```

```
Search took a total of:  0.05125927925109863
Query:
```



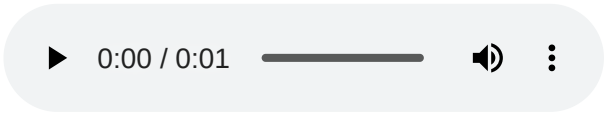

```
Results:
Distance: 0.0
```




```
Distance: 0.11495525389909744
```


```
Distance: 0.19858866930007935
```

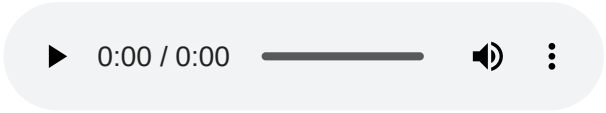

```
--------------------------------------------------
Query:
```




```
Results:
Distance: 0.0
```


```
Distance: 0.2286444902420044
```


```
Distance: 0.26576918363571167
```




```
--------------------------------------------------
```

In [ ]: