Project

In this Project, you will bring together many of the tools and techniques that you have learned throughout this course into a final project. You can choose from many different paths to get to the solution.

Business scenario

You work for a training organization that recently developed an introductory course about machine learning (ML). The course includes more than 40 videos that cover a broad range of ML topics. You have been asked to create an application that will students can use to quickly locate and view video content by searching for topics and key phrases.

You have downloaded all of the videos to an Amazon Simple Storage Service (Amazon S3) bucket. Your assignment is to produce a dashboard that meets your supervisor's requirements.

Project steps

To complete this project, you will follow these steps:

1. Viewing the video files
2. Transcribing the videos
3. Normalizing the text
4. Extracting key phrases and topics
5. Creating the dashboard

Useful information

The following cell contains some information that might be useful as you complete this project.

In [1]:
```
bucket = "c56161a939430l3396553t1w744137092661-labbucket-rn642jaq01e9"
job_data_access_role = 'arn:aws:iam::744137092661:role/service-role/c56161a939430l3396553t1w7-ComprehendDataAccessRole-1P24MSS
```

1. Viewing the video files

(Go to top)

The source video files are located in the following shared Amazon Simple Storage Service (Amazon S3) bucket.

In [2]:
```
!aws s3 ls s3://aws-tc-largeobjects/CUR-TF-200-ACMNLP-1/video/
```

```
2021-04-26 20:17:33  410925369 Mod01_Course Overview.mp4
2021-04-26 20:10:02   39576695 Mod02_Intro.mp4
2021-04-26 20:31:23  302994828 Mod02_Sect01.mp4
2021-04-26 20:17:33  416563881 Mod02_Sect02.mp4
2021-04-26 20:17:33  318685583 Mod02_Sect03.mp4
2021-04-26 20:17:33  255877251 Mod02_Sect04.mp4
2021-04-26 20:23:51   99988046 Mod02_Sect05.mp4
2021-04-26 20:24:54   50700224 Mod02_WrapUp.mp4
2021-04-26 20:26:27   60627667 Mod03_Intro.mp4
2021-04-26 20:26:28  272229844 Mod03_Sect01.mp4
2021-04-26 20:27:06  309127124 Mod03_Sect02_part1.mp4
2021-04-26 20:27:06  195635527 Mod03_Sect02_part2.mp4
2021-04-26 20:28:03  123924818 Mod03_Sect02_part3.mp4
2021-04-26 20:31:28  171681915 Mod03_Sect03_part1.mp4
2021-04-26 20:32:07  285200083 Mod03_Sect03_part2.mp4
2021-04-26 20:33:17  105470345 Mod03_Sect03_part3.mp4
2021-04-26 20:35:10  157185651 Mod03_Sect04_part1.mp4
2021-04-26 20:36:27  187435635 Mod03_Sect04_part2.mp4
2021-04-26 20:36:40  280720369 Mod03_Sect04_part3.mp4
2021-04-26 20:40:01  443479313 Mod03_Sect05.mp4
2021-04-26 20:40:08  234182186 Mod03_Sect06.mp4
2021-04-26 20:40:33  207718047 Mod03_Sect07_part1.mp4
2021-04-26 20:42:07  125592110 Mod03_Sect07_part2.mp4
2021-04-26 20:45:10  508500301 Mod03_Sect07_part3.mp4
2021-04-26 20:46:16  320126756 Mod03_Sect08.mp4
2021-04-26 20:46:43   41839508 Mod03_WrapUp.mp4
2021-04-26 20:46:55   34148489 Mod04_Intro.mp4
2021-04-26 20:48:24   84959465 Mod04_Sect01.mp4
2021-04-26 20:48:25  345182970 Mod04_Sect02_part1.mp4
2021-04-26 20:51:34  218661651 Mod04_Sect02_part2.mp4
2021-04-26 20:53:32  430140637 Mod04_Sect02_part3.mp4
2021-04-26 20:56:03   22036605 Mod04_WrapUp.mp4
2021-04-26 20:57:18   49187118 Mod05_Intro.mp4
2021-04-26 20:58:19  245798071 Mod05_Sect01_ver2.mp4
2021-04-26 20:58:50  233314835 Mod05_Sect02_part1_ver2.mp4
2021-04-26 20:59:14  348545306 Mod05_Sect02_part2.mp4
2021-04-26 20:59:17  239142711 Mod05_Sect03_part1.mp4
2021-04-26 21:06:04  267533559 Mod05_Sect03_part2.mp4
2021-04-26 21:06:06  212502220 Mod05_Sect03_part3.mp4
2021-04-26 21:06:48  206317022 Mod05_Sect03_part4_ver2.mp4
2021-04-26 21:06:48   60361230 Mod05_WrapUp_ver2.mp4
```

```
2021-04-26 21:09:14   35397860 Mod06_Intro.mp4
2021-04-26 21:09:24  845633599 Mod06_Sect01.mp4
2021-04-26 21:10:47  326126684 Mod06_Sect02.mp4
2021-04-26 21:12:26   19790740 Mod06_WrapUp.mp4
2021-04-26 21:12:56  131249036 Mod07_Sect01.mp4
```

Install and configure the required Libraries and Packages

In [3]:

```
# FFmpeg installation
!wget https://johnvansickle.com/ffmpeg/builds/ffmpeg-git-amd64-static.tar.xz
!tar -xf ffmpeg-git-amd64-static.tar.xz
%cd ffmpeg-git-20240629-amd64-static
!sudo mv ffmpeg /usr/local/bin/
!sudo mv ffprobe /usr/local/bin/
!ffmpeg -version
```

```
--2025-04-02 09:14:17--  https://johnvansickle.com/ffmpeg/builds/ffmpeg-git-amd64-static.tar.xz
Resolving johnvansickle.com (johnvansickle.com)... 107.180.57.212
Connecting to johnvansickle.com (johnvansickle.com)|107.180.57.212|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 41964060 (40M) [application/x-xz]
Saving to: ‘ffmpeg-git-amd64-static.tar.xz.2’

100%[=====================================>] 41,964,060  35.0MB/s   in 1.1s

2025-04-02 09:14:19 (35.0 MB/s) - ‘ffmpeg-git-amd64-static.tar.xz.2’ saved [41964060/41964060]

/home/ec2-user/SageMaker/ffmpeg-git-20240629-amd64-static
ffmpeg version N-71064-gd5e603ddc0-static https://johnvansickle.com/ffmpeg/  Copyright (c) 2000-2024 the FFmpeg developers
built with gcc 8 (Debian 8.3.0-6)
configuration: --enable-gpl --enable-version3 --enable-static --disable-debug --disable-ffplay --disable-indev=sndio --disabl
e-outdev=sndio --cc=gcc --enable-fontconfig --enable-frei0r --enable-gnutls --enable-gmp --enable-libgme --enable-gray --enab
le-libaom --enable-libfribidi --enable-libass --enable-libvmaf --enable-libfreetype --enable-libmp3lame --enable-libopencore-
amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-librubberband --enable-libsoxr --enable-libspeex --enable-libs
rt --enable-libvorbis --enable-libopus --enable-libtheora --enable-libvidstab --enable-libvo-amrwbenc --enable-libvpx --enabl
e-libwebp --enable-libx264 --enable-libx265 --enable-libxml2 --enable-libdav1d --enable-libxvid --enable-libzvbi --enable-lib
zimg
libavutil      59. 27.100 / 59. 27.100
libavcodec     61.  9.100 / 61.  9.100
libavformat    61.  4.100 / 61.  4.100
libavdevice    61.  2.100 / 61.  2.100
libavfilter    10.  2.102 / 10.  2.102
libswscale      8.  2.100 /  8.  2.100
libswresample   5.  2.100 /  5.  2.100
libpostproc    58.  2.100 / 58.  2.100
```

In [4]:

```
# Install necessary Python libraries
# These are used for audio processing, transcription, and keyword/topic extraction

!pip install pydub                       # For handling audio segments
!pip install git+https://github.com/openai/whisper.git  # OpenAI's Whisper for transcription
!pip install rake_nltk                   # RAKE algorithm for key phrase extraction
!pip install gensim                      # For LDA topic modeling
```

```
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
Collecting git+https://github.com/openai/whisper.git
  Cloning https://github.com/openai/whisper.git to /tmp/pip-req-build-ju_xlhkq
  Running command git clone --filter=blob:none --quiet https://github.com/openai/whisper.git /tmp/pip-req-build-ju_xlhkq
  Resolved https://github.com/openai/whisper.git to commit 517a43ecd132a2089d85f4ebc044728a71d49f6e
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: more-itertools in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from ope
nai-whisper==20240930) (10.6.0)
Requirement already satisfied: numba in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from openai-whisp
er==20240930) (0.61.0)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from openai-whisp
er==20240930) (1.26.4)
Collecting tiktoken (from openai-whisper==20240930)
  Downloading tiktoken-0.9.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.7 kB)
Collecting torch (from openai-whisper==20240930)
  Downloading torch-2.6.0-cp310-cp310-manylinux1_x86_64.whl.metadata (28 kB)
Requirement already satisfied: tqdm in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from openai-whispe
r==20240930) (4.67.1)
Collecting triton>=2 (from openai-whisper==20240930)
  Downloading triton-3.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.4 kB)
Requirement already satisfied: llvmlite<0.45,>=0.44.0dev0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packag
es (from numba->openai-whisper==20240930) (0.44.0)
Requirement already satisfied: regex>=2022.1.18 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from t
iktoken->openai-whisper==20240930) (2024.11.6)
Requirement already satisfied: requests>=2.26.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from t
iktoken->openai-whisper==20240930) (2.32.3)
Requirement already satisfied: filelock in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from torch->op
enai-whisper==20240930) (3.17.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-package
s (from torch->openai-whisper==20240930) (4.12.2)
Requirement already satisfied: networkx in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from torch->op
enai-whisper==20240930) (3.4)
Requirement already satisfied: jinja2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from torch->open
ai-whisper==20240930) (3.1.5)
Requirement already satisfied: fsspec in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from torch->open
```

```
ai-whisper==20240930) (2025.2.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->openai-whisper==20240930)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->openai-whisper==20240930)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->openai-whisper==20240930)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->openai-whisper==20240930)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->openai-whisper==20240930)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->openai-whisper==20240930)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->openai-whisper==20240930)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->openai-whisper==20240930)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->openai-whisper==20240930)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparselt-cu12==0.6.2 (from torch->openai-whisper==20240930)
  Downloading nvidia_cusparselt_cu12-0.6.2-py3-none-manylinux2014_x86_64.whl.metadata (6.8 kB)
Collecting nvidia-nccl-cu12==2.21.5 (from torch->openai-whisper==20240930)
  Downloading nvidia_nccl_cu12-2.21.5-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.4.127 (from torch->openai-whisper==20240930)
  Downloading nvidia_nvtx_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.7 kB)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->openai-whisper==20240930)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting sympy==1.13.1 (from torch->openai-whisper==20240930)
  Downloading sympy-1.13.1-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
sympy==1.13.1->torch->openai-whisper==20240930) (1.3.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages
(from requests>=2.26.0->tiktoken->openai-whisper==20240930) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from reque
sts>=2.26.0->tiktoken->openai-whisper==20240930) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.26.0->tiktoken->openai-whisper==20240930) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
requests>=2.26.0->tiktoken->openai-whisper==20240930) (2025.1.31)
Requirement already satisfied: MarkupSafe>=2.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from ji
nja2->torch->openai-whisper==20240930) (3.0.2)
```

```
Downloading triton-3.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (253.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 253.1/253.1 MB 60.9 MB/s eta 0:00:0000:0100:01
Downloading tiktoken-0.9.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 79.3 MB/s eta 0:00:00
Downloading torch-2.6.0-cp310-cp310-manylinux1_x86_64.whl (766.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 766.7/766.7 MB 19.2 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 35.3 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 35.2 MB/s eta 0:00:00:00:01
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 111.4 MB/s eta 0:00:0000:01
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 62.8 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 21.3 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 59.7 MB/s eta 0:00:0000:0100:01
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 133.9 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 55.0 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 44.9 MB/s eta 0:00:0000:0100:01
Downloading nvidia_cusparselt_cu12-0.6.2-py3-none-manylinux2014_x86_64.whl (150.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 150.1/150.1 MB 63.9 MB/s eta 0:00:0000:0100:01
Downloading nvidia_nccl_cu12-2.21.5-py3-none-manylinux2014_x86_64.whl (188.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.7/188.7 MB 46.3 MB/s eta 0:00:0000:0100:01
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 133.7 MB/s eta 0:00:00
Downloading nvidia_nvtx_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (99 kB)
Downloading sympy-1.13.1-py3-none-any.whl (6.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.2/6.2 MB 51.1 MB/s eta 0:00:00
Building wheels for collected packages: openai-whisper
  Building wheel for openai-whisper (pyproject.toml) ... done
  Created wheel for openai-whisper: filename=openai_whisper-20240930-py3-none-any.whl size=803707 sha256=fe058ed8fb4929f618a3
47ad60da61fdd2112d71e605b9573fa4f0ab5838bdf0
  Stored in directory: /tmp/pip-ephem-wheel-cache-47zkpatx/wheels/8b/6c/d0/622666868c179f156cf595c8b6f06f88bc5d80c4b31dccaa03
Successfully built openai-whisper
Installing collected packages: triton, nvidia-cusparselt-cu12, sympy, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu
12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-c
```

```
ublas-cu12, tiktoken, nvidia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cusolver-cu12, torch, openai-whisper
  Attempting uninstall: sympy
    Found existing installation: sympy 1.13.3
    Uninstalling sympy-1.13.3:
      Successfully uninstalled sympy-1.13.3
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cud
a-runtime-cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-cu12-10.3.5.147 nvidia-cusolver-c
u12-11.6.1.9 nvidia-cusparse-cu12-12.3.1.170 nvidia-cusparselt-cu12-0.6.2 nvidia-nccl-cu12-2.21.5 nvidia-nvjitlink-cu12-12.4.
127 nvidia-nvtx-cu12-12.4.127 openai-whisper-20240930 sympy-1.13.1 tiktoken-0.9.0 torch-2.6.0 triton-3.2.0
Collecting rake_nltk
  Downloading rake_nltk-1.0.6-py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: nltk<4.0.0,>=3.6.2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
rake_nltk) (3.9.1)
Requirement already satisfied: click in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from nltk<4.0.0,>
=3.6.2->rake_nltk) (8.1.8)
Requirement already satisfied: joblib in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from nltk<4.0.0,
>=3.6.2->rake_nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from nl
tk<4.0.0,>=3.6.2->rake_nltk) (2024.11.6)
Requirement already satisfied: tqdm in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from nltk<4.0.0,>=
3.6.2->rake_nltk) (4.67.1)
Downloading rake_nltk-1.0.6-py3-none-any.whl (9.1 kB)
Installing collected packages: rake_nltk
Successfully installed rake_nltk-1.0.6
Collecting gensim
  Downloading gensim-4.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.2 kB)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from
gensim) (1.26.4)
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
Collecting smart-open>=1.8.1 (from gensim)
  Downloading smart_open-7.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: wrapt in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from smart-open>=
1.8.1->gensim) (1.17.2)
Downloading gensim-4.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.5/26.5 MB 145.4 MB/s eta 0:00:00
Downloading scipy-1.13.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 135.7 MB/s eta 0:00:0000:01
Downloading smart_open-7.1.0-py3-none-any.whl (61 kB)
Installing collected packages: smart-open, scipy, gensim
  Attempting uninstall: scipy
```

```
    Found existing installation: scipy 1.15.1
    Uninstalling scipy-1.15.1:
      Successfully uninstalled scipy-1.15.1
Successfully installed gensim-4.3.3 scipy-1.13.1 smart-open-7.1.0
```

In [5]:

```python
# Import standard libraries
import os
import io
import re
import json
import logging
import subprocess
from concurrent.futures import ThreadPoolExecutor

# AWS SDK
import boto3
from botocore.exceptions import ClientError

# Audio transcription and NLP tools
import whisper
from rake_nltk import Rake
from gensim import corpora
from gensim.models import LdaModel

# NLTK for tokenization, lemmatization, and stopword filtering
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Download required NLTK resources
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/ec2-user/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /home/ec2-user/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /home/ec2-user/nltk_data...
[nltk_data] Downloading package punkt_tab to
[nltk_data]     /home/ec2-user/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

Out[5]: True

2. Transcribing the videos

(Go to top)

Use this section to implement your solution to transcribe the videos.

In [6]:
```python
# ------------------- Copy video to my S3 bucket -------------------
print("Copy videos to my S3 bucket...")

# Initialize AWS S3 client
s3_client = boto3.client('s3')

# Source and destination bucket details
source_bucket_name = 'nlpprojectamirhossein'
destination_bucket_name = 'nlpprojectamirhossein'
destination_prefix = 'datasource/'

# Track file status
existing_files = []
new_files_uploaded = False

# Check if a file exists in the destination bucket
def check_if_file_exists(destination_key):
    try:
        s3_client.head_object(Bucket=destination_bucket_name, Key=destination_key)
        return True
    except ClientError as e:
```

```
        if e.response['Error']['Code'] == '404':
            return False
        print(f'Error checking {destination_key}: {e}')
        return False

# Upload a file from source to destination bucket
def upload_file_to_destination(file_key, destination_key):
    global new_files_uploaded
    try:
        video_buffer = io.BytesIO()
        s3_client.download_fileobj(Bucket=source_bucket_name, Key=file_key, Fileobj=video_buffer)
        video_buffer.seek(0)
        s3_client.upload_fileobj(video_buffer, Bucket=destination_bucket_name, Key=destination_key)
        print(f'Uploaded {file_key} to s3://{destination_bucket_name}/{destination_key}')
        new_files_uploaded = True
    except ClientError as e:
        print(f'Error uploading {file_key}: {e}')

# Process each object in the source bucket
def process_files(response):
    global new_files_uploaded, existing_files
    for obj in response.get('Contents', []):
        file_key = obj['Key']
        if file_key.endswith('.mp4'):
            print(f'⚙️ Processing file: {file_key}')
            destination_key = os.path.join(destination_prefix, os.path.basename(file_key))
            if check_if_file_exists(destination_key):
                existing_files.append(destination_key)
            else:
                upload_file_to_destination(file_key, destination_key)

# Main function to list and process MP4 files
def main():
    response = s3_client.list_objects_v2(Bucket=source_bucket_name)
    if 'Contents' in response:
        process_files(response)

    if not new_files_uploaded and existing_files:
        print("✅ All MP4 files already exist in the destination bucket.")
    elif new_files_uploaded:
        print("✅ New MP4 files have been successfully copied to the destination bucket.")
```

```
    else:
        print("⚠️ No MP4 files were found in the source bucket.")

# Run main function
if __name__ == "__main__":
    main()

# ------------------- Convert MP4 to WAV and store in S3 -------------------
print("_____________________________________________")
print("Convert MP4 to WAV and store in S3...")

# Initialize S3 client
s3_client = boto3.client('s3')

# Define bucket names and destination path
source_bucket_name = 'nlpprojectamirhossein'
destination_bucket_name = 'nlpprojectamirhossein'
destination_prefix = 'converted_files/'

# Collect already existing WAV files in destination bucket
existing_files = set()
response = s3_client.list_objects_v2(Bucket=destination_bucket_name, Prefix=destination_prefix)
if 'Contents' in response:
    existing_files = {obj['Key'] for obj in response['Contents']}

# List MP4 files in the source bucket
response = s3_client.list_objects_v2(Bucket=source_bucket_name)

if 'Contents' in response:
    for obj in response['Contents']:
        file_key = obj['Key']
        if not file_key.endswith('.mp4'):
            continue

        # Define WAV filename and full S3 destination key
        wav_filename = os.path.basename(file_key).replace('.mp4', '.wav')
        wav_s3_path = os.path.join(destination_prefix, wav_filename)

        # Skip if WAV already exists
        if wav_s3_path in existing_files:
            print(f"✅ {file_key} WAV is already exists.")
```

```
            continue

        print(f"⚙️ Processing: {file_key}")

        # Define local file paths
        local_video_path = f"/tmp/{os.path.basename(file_key)}"
        output_wav_path = f"/tmp/{wav_filename}"

        try:
            # Download MP4 from S3
            s3_client.download_file(Bucket=source_bucket_name, Key=file_key, Filename=local_video_path)
            print(f"Downloaded to {local_video_path}")

            # Convert MP4 to WAV using FFmpeg
            ffmpeg_cmd = [
                "ffmpeg",
                "-i", local_video_path,
                "-vn", "-acodec", "pcm_s16le",
                "-ar", "16000", "-ac", "1",
                output_wav_path
            ]
            process = subprocess.Popen(ffmpeg_cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
            _, error = process.communicate()

            if process.returncode != 0:
                print(f"❌ Conversion error for {file_key}: {error.decode()}")
                continue

            print(f"✅ Conversion complete: {output_wav_path}")

            # Upload WAV to S3
            s3_client.upload_file(output_wav_path, Bucket=destination_bucket_name, Key=wav_s3_path)
            print(f"✅ Uploaded to s3://{destination_bucket_name}/{wav_s3_path}")

        except ClientError as e:
            print(f"❌ AWS error for {file_key}: {e}")
        except Exception as e:
            print(f"❌ General error for {file_key}: {e}")
        finally:
            # Cleanup local files
            for path in [local_video_path, output_wav_path]:
```

```
                if os.path.exists(path):
                    os.remove(path)
else:
    print("⚠️ No MP4 files found in the source bucket.")

# ------------------- Transcribe audio files and save to JSON in S3 -------------------
print("_____________________________________________________")
print("Transcribing audio files and save to JSON in S3...")

# Initialize AWS S3 client
s3_client = boto3.client('s3')

# Bucket configuration
source_bucket_name = 'nlpprojectamirhossein'      # Where .wav files are stored
destination_bucket_name = 'nlpprojectamirhossein' # Where the output JSON will be stored
destination_prefix = 'json/'                      # Folder for JSON output in destination
json_s3_key = os.path.join(destination_prefix, "transcriptions.json")

# Check if transcription JSON already exists
def check_json_exists():
    try:
        s3_client.head_object(Bucket=destination_bucket_name, Key=json_s3_key)
        print(f"✅ Transcription JSON is already exists at s3://{destination_bucket_name}/{json_s3_key}")
        return True
    except ClientError as e:
        if e.response['Error']['Code'] == '404':
            return False
        print(f"⚠️ Error checking JSON file: {e}")
        return False

# Skip transcription if JSON exists
if check_json_exists():
    print("🛑 Skipping transcription; JSON file already exists.")
else:
    # Load Whisper model for transcription
    model = whisper.load_model("base")  # You can change to "small", "medium", "large"

    # List .wav files in source bucket
    response = s3_client.list_objects_v2(Bucket=source_bucket_name)
    transcriptions = {}
```

```python
if 'Contents' in response:
    for obj in response['Contents']:
        file_key = obj['Key']
        if not file_key.endswith('.wav'):
            continue

        print(f"⚙️ Transcribing: {file_key}")
        base_filename = os.path.basename(file_key)
        audio_name = os.path.splitext(base_filename)[0]
        local_audio_path = audio_name

        try:
            # Download WAV to local file
            s3_client.download_file(Bucket=source_bucket_name, Key=file_key, Filename=local_audio_path)
            print(f"⬇️ Downloaded: {local_audio_path}")

            # Transcribe with Whisper
            result = model.transcribe(local_audio_path)
            transcriptions[audio_name] = result["text"]

        except Exception as e:
            print(f"❌ Error transcribing {file_key}: {e}")
        finally:
            if os.path.exists(local_audio_path):
                os.remove(local_audio_path)

# Save transcription JSON to S3
if transcriptions:
    json_buffer = io.BytesIO()
    json_buffer.write(json.dumps(transcriptions, indent=4).encode('utf-8'))
    json_buffer.seek(0)

    try:
        s3_client.upload_fileobj(json_buffer, Bucket=destination_bucket_name, Key=json_s3_key)
        print(f"✅ Uploaded transcription JSON to s3://{destination_bucket_name}/{json_s3_key}")
    except ClientError as e:
        print(f"❌ Upload failed: {e}")
else:
    print("⚠️ No transcriptions were created.")
```

```
Copy videos to my S3 bucket...
⚙ Processing file: datasource/Mod01_Course Overview.mp4
⚙ Processing file: datasource/Mod02_Intro.mp4
⚙ Processing file: datasource/Mod02_Sect01.mp4
⚙ Processing file: datasource/Mod02_Sect02.mp4
⚙ Processing file: datasource/Mod02_Sect03.mp4
⚙ Processing file: datasource/Mod02_Sect04.mp4
⚙ Processing file: datasource/Mod02_Sect05.mp4
⚙ Processing file: datasource/Mod02_WrapUp.mp4
⚙ Processing file: datasource/Mod03_Intro.mp4
⚙ Processing file: datasource/Mod03_Sect01.mp4
⚙ Processing file: datasource/Mod03_Sect02_part1.mp4
⚙ Processing file: datasource/Mod03_Sect02_part2.mp4
⚙ Processing file: datasource/Mod03_Sect02_part3.mp4
⚙ Processing file: datasource/Mod03_Sect03_part1.mp4
⚙ Processing file: datasource/Mod03_Sect03_part2.mp4
⚙ Processing file: datasource/Mod03_Sect03_part3.mp4
⚙ Processing file: datasource/Mod03_Sect04_part1.mp4
⚙ Processing file: datasource/Mod03_Sect04_part2.mp4
⚙ Processing file: datasource/Mod03_Sect04_part3.mp4
⚙ Processing file: datasource/Mod03_Sect05.mp4
⚙ Processing file: datasource/Mod03_Sect06.mp4
⚙ Processing file: datasource/Mod03_Sect07_part1.mp4
⚙ Processing file: datasource/Mod03_Sect07_part2.mp4
⚙ Processing file: datasource/Mod03_Sect07_part3.mp4
⚙ Processing file: datasource/Mod03_Sect08.mp4
⚙ Processing file: datasource/Mod03_WrapUp.mp4
⚙ Processing file: datasource/Mod04_Intro.mp4
⚙ Processing file: datasource/Mod04_Sect01.mp4
⚙ Processing file: datasource/Mod04_Sect02_part1.mp4
⚙ Processing file: datasource/Mod04_Sect02_part2.mp4
⚙ Processing file: datasource/Mod04_Sect02_part3.mp4
⚙ Processing file: datasource/Mod04_WrapUp.mp4
⚙ Processing file: datasource/Mod05_Intro.mp4
⚙ Processing file: datasource/Mod05_Sect01_ver2.mp4
⚙ Processing file: datasource/Mod05_Sect02_part1_ver2.mp4
⚙ Processing file: datasource/Mod05_Sect02_part2.mp4
⚙ Processing file: datasource/Mod05_Sect03_part1.mp4
⚙ Processing file: datasource/Mod05_Sect03_part2.mp4
⚙ Processing file: datasource/Mod05_Sect03_part3.mp4
⚙ Processing file: datasource/Mod05_Sect03_part4_ver2.mp4
```

⚙️ Processing file: datasource/Mod05_WrapUp_ver2.mp4
⚙️ Processing file: datasource/Mod06_Intro.mp4
⚙️ Processing file: datasource/Mod06_Sect01.mp4
⚙️ Processing file: datasource/Mod06_Sect02.mp4
⚙️ Processing file: datasource/Mod06_WrapUp.mp4
⚙️ Processing file: datasource/Mod07_Sect01.mp4
✅ All MP4 files already exist in the destination bucket.

Convert MP4 to WAV and store in S3...
✅ datasource/Mod01_Course Overview.mp4 WAV is already exists.
✅ datasource/Mod02_Intro.mp4 WAV is already exists.
✅ datasource/Mod02_Sect01.mp4 WAV is already exists.
✅ datasource/Mod02_Sect02.mp4 WAV is already exists.
✅ datasource/Mod02_Sect03.mp4 WAV is already exists.
✅ datasource/Mod02_Sect04.mp4 WAV is already exists.
✅ datasource/Mod02_Sect05.mp4 WAV is already exists.
✅ datasource/Mod02_WrapUp.mp4 WAV is already exists.
✅ datasource/Mod03_Intro.mp4 WAV is already exists.
✅ datasource/Mod03_Sect01.mp4 WAV is already exists.
✅ datasource/Mod03_Sect02_part1.mp4 WAV is already exists.
✅ datasource/Mod03_Sect02_part2.mp4 WAV is already exists.
✅ datasource/Mod03_Sect02_part3.mp4 WAV is already exists.
✅ datasource/Mod03_Sect03_part1.mp4 WAV is already exists.
✅ datasource/Mod03_Sect03_part2.mp4 WAV is already exists.
✅ datasource/Mod03_Sect03_part3.mp4 WAV is already exists.
✅ datasource/Mod03_Sect04_part1.mp4 WAV is already exists.
✅ datasource/Mod03_Sect04_part2.mp4 WAV is already exists.
✅ datasource/Mod03_Sect04_part3.mp4 WAV is already exists.
✅ datasource/Mod03_Sect05.mp4 WAV is already exists.
✅ datasource/Mod03_Sect06.mp4 WAV is already exists.
✅ datasource/Mod03_Sect07_part1.mp4 WAV is already exists.
✅ datasource/Mod03_Sect07_part2.mp4 WAV is already exists.
✅ datasource/Mod03_Sect07_part3.mp4 WAV is already exists.
✅ datasource/Mod03_Sect08.mp4 WAV is already exists.
✅ datasource/Mod03_WrapUp.mp4 WAV is already exists.
✅ datasource/Mod04_Intro.mp4 WAV is already exists.
✅ datasource/Mod04_Sect01.mp4 WAV is already exists.
✅ datasource/Mod04_Sect02_part1.mp4 WAV is already exists.
✅ datasource/Mod04_Sect02_part2.mp4 WAV is already exists.
✅ datasource/Mod04_Sect02_part3.mp4 WAV is already exists.
✅ datasource/Mod04_WrapUp.mp4 WAV is already exists.

```
✅ datasource/Mod05_Intro.mp4 WAV is already exists.
✅ datasource/Mod05_Sect01_ver2.mp4 WAV is already exists.
✅ datasource/Mod05_Sect02_part1_ver2.mp4 WAV is already exists.
✅ datasource/Mod05_Sect02_part2.mp4 WAV is already exists.
✅ datasource/Mod05_Sect03_part1.mp4 WAV is already exists.
✅ datasource/Mod05_Sect03_part2.mp4 WAV is already exists.
✅ datasource/Mod05_Sect03_part3.mp4 WAV is already exists.
✅ datasource/Mod05_Sect03_part4_ver2.mp4 WAV is already exists.
✅ datasource/Mod05_WrapUp_ver2.mp4 WAV is already exists.
✅ datasource/Mod06_Intro.mp4 WAV is already exists.
✅ datasource/Mod06_Sect01.mp4 WAV is already exists.
✅ datasource/Mod06_Sect02.mp4 WAV is already exists.
✅ datasource/Mod06_WrapUp.mp4 WAV is already exists.
✅ datasource/Mod07_Sect01.mp4 WAV is already exists.
```

```
Transcribing audio files and save to JSON in S3...
✅ Transcription JSON is already exists at s3://nlpprojectamirhossein/json/transcriptions.json
🛑 Skipping transcription; JSON file already exists.
```

3. Normalizing the text

(Go to top)

Use this section to perform any text normalization steps that are necessary for your solution.

In [7]:
```python
# ------------------- Normalize transcript text using NLTK -------------------

# Initialize NLTK tools
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# S3 bucket and file paths
s3_client = boto3.client('s3')
bucket_name = 'nlpprojectamirhossein'
json_key = 'json/transcriptions.json'
normalized_json_key = 'json/normalized_transcripts.json'

# Text normalization pipeline
def normalize_text_nltk(text):
```

```
    """Lowercase, remove punctuation, filter stopwords, and lemmatize."""
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = word_tokenize(text)
    normalized_tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    return ' '.join(normalized_tokens)

# Download transcription JSON from S3
def download_json_from_s3():
    try:
        json_buffer = io.BytesIO()
        s3_client.download_fileobj(bucket_name, json_key, json_buffer)
        json_buffer.seek(0)
        print(f"✅ Transcription JSON is downloaded from s3://{bucket_name}/{json_key}")
        return json_buffer
    except ClientError as e:
        print(f"❌ Download error: {e}")
        return None

# Save normalized result locally
def save_normalized_json_to_local(normalized_data):
    try:
        path = '/tmp/normalized_transcripts.json'
        with open(path, 'w') as file:
            json.dump(normalized_data, file, indent=4)
        print(f"📄 Normalized transcription JSON is saved locally at {path}")
        return path
    except Exception as e:
        print(f"❌ Local save error: {e}")
        return None

# Upload normalized JSON to S3
def upload_json_to_s3(local_path):
    try:
        s3_client.upload_file(local_path, bucket_name, normalized_json_key)
        print(f"✅ Normalized transcription JSON is uploaded to s3://{bucket_name}/{normalized_json_key}")
    except ClientError as e:
        print(f"❌ Upload error: {e}")

# Main process
def process_transcriptions():
```

```
    json_buffer = download_json_from_s3()
    if not json_buffer:
        return

    transcriptions = json.load(json_buffer)
    normalized_transcriptions = {
        key: normalize_text_nltk(value) for key, value in transcriptions.items()
    }

    local_path = save_normalized_json_to_local(normalized_transcriptions)
    if local_path:
        upload_json_to_s3(local_path)

# Run normalization
process_transcriptions()
```

```
✅ Transcription JSON is downloaded from s3://nlpprojectamirhossein/json/transcriptions.json
📄 Normalized transcription JSON is saved locally at /tmp/normalized_transcripts.json
✅ Normalized transcription JSON is uploaded to s3://nlpprojectamirhossein/json/normalized_transcripts.json
```

4. Extracting key phrases and topics

(Go to top)

Use this section to extract the key phrases and topics from the videos.

In [23]:
```
# ------------------- SETUP -------------------

s3_client = boto3.client('s3')
bucket_name = 'nlpprojectamirhossein'

# Input and output keys
raw_json_key = 'json/transcriptions.json'
normalized_json_key = 'json/normalized_transcripts.json'
output_key = 'json/extracted_key_phrases_and_topics.json'

# Logging configuration
logging.basicConfig(
    filename='project.log',
```

```
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

# ------------------- TEXT PROCESSING -------------------

def normalize_text_for_rake(text):
    """Lightly clean text and chunk it to simulate sentence-like structure for RAKE."""
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s\.\,\?\!]', '', text)
    words = text.split()
    chunks = [' '.join(words[i:i + 20]) + '.' for i in range(0, len(words), 20)]
    return ' '.join(chunks)

# ------------------- NLP FUNCTIONS -------------------
def extract_key_phrases(text, top_n=5, min_score=11, min_words=2):
    try:
        rake = Rake()
        rake.extract_keywords_from_text(text)
        phrases_with_scores = rake.get_ranked_phrases_with_scores()

        seen = set()
        filtered_phrases = []

        for score, phrase in phrases_with_scores:
            phrase = phrase.strip()
            if phrase in seen:
                continue
            if score < min_score or len(phrase.split()) < min_words:
                continue
            seen.add(phrase)
            filtered_phrases.append({
                "text": phrase,
                "score": round(score, 2)
            })
            if len(filtered_phrases) == top_n:
                break

        return filtered_phrases

    except Exception as e:
```

```
        logging.error(f"Error extracting key phrases: {e}")
        return []

# Define a set of filler/noise words to exclude from topics
FILLER_WORDS = {
    "youll", "section", "module", "topic", "video", "lesson", "introduction",
    "overview", "slide", "course", "content", "example", "use", "identify", "set", "well", "one",
}

def identify_topics(text, num_topics=3):
    """Identify topics from normalized text using LDA."""
    try:
        words = text.split()
        if len(words) < 5:
            logging.warning("Text too short for LDA topic modeling.")
            return []
        dictionary = corpora.Dictionary([words])
        corpus = [dictionary.doc2bow(words)]
        lda_model = LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=15)

        # Get the top topic by weight
        topic_distribution = lda_model.get_document_topics(corpus[0])
        top_topic_id = max(topic_distribution, key=lambda x: x[1])[0]

        # Return only the top topic
        return [lda_model.show_topic(topicid=top_topic_id, topn=5)]
    except Exception as e:
        logging.error(f"Error extracting topics: {e}")
        return []

def clean_lda_topics(lda_output):
    """Clean and normalize the most relevant LDA topic into one list."""
    if not lda_output:
        return []

    topic = lda_output[0]  # only one topic
    words = [
        re.sub(r'[^a-z]', '', item[0].strip().lower())  # Clean word
        for item in topic
    ]
```

```python
    # Filter out filler/noise words
    words = [w for w in words if len(w) > 2 and w not in FILLER_WORDS]

    # Deduplicate and sort for consistency
    return [sorted(set(words))] if words else []

# ------------------- S3 HELPERS -------------------

def check_file_exists(bucket, key):
    try:
        s3_client.head_object(Bucket=bucket, Key=key)
        return True
    except s3_client.exceptions.ClientError as e:
        if e.response['Error']['Code'] == '404':
            return False
        logging.error(f"Error checking {key}: {e}")
        return False

def download_file_from_s3(bucket, key, local_filename):
    if not check_file_exists(bucket, key):
        raise FileNotFoundError(f"{key} not found in {bucket}")
    try:
        s3_client.download_file(bucket, key, local_filename)
        logging.info(f"Downloaded {key}")
    except Exception as e:
        logging.error(f"Error downloading {key}: {e}")
        raise

# ------------------- PROCESSING FUNCTION -------------------

def process_transcription(key, raw_text, normalized_text):
    try:
        key_phrases = extract_key_phrases(normalize_text_for_rake(raw_text))
        raw_topics = identify_topics(normalized_text)
        topics = clean_lda_topics(raw_topics)
        return key, {
            "key_phrases": key_phrases,
            "topics": topics
        }
    except Exception as e:
        logging.error(f"Error processing {key}: {e}")
```

```
        return key, {
            "key_phrases": [],
            "topics": []
        }

# ------------------- MAIN SCRIPT -------------------

# Step 1: Download files from S3
download_file_from_s3(bucket_name, raw_json_key, 'raw_transcriptions.json')
download_file_from_s3(bucket_name, normalized_json_key, 'normalized_transcriptions.json')

# Step 2: Load JSON contents
with open('raw_transcriptions.json', 'r') as f:
    raw_transcriptions = json.load(f)

with open('normalized_transcriptions.json', 'r') as f:
    normalized_transcriptions = json.load(f)

# Step 3: Process each transcript in parallel
results = {}
with ThreadPoolExecutor() as executor:
    futures = {
        executor.submit(
            process_transcription,
            key,
            raw_transcriptions[key],
            normalized_transcriptions[key]
        ): key
        for key in raw_transcriptions if key in normalized_transcriptions
    }
    for future in futures:
        try:
            key, result = future.result()
            results[key] = result
        except Exception as e:
            logging.error(f"Failed to process key {futures[future]}: {e}")

# Step 4: Save processed data locally
with open('extracted_key_phrases_and_topics.json', 'w') as f:
    json.dump(results, f, indent=4)
```

```
# Step 5: Upload results to S3
try:
    s3_client.upload_file('extracted_key_phrases_and_topics.json', bucket_name, output_key)
    print(f"✅ Extracted key phrases and topics JSON are uploaded to s3://{bucket_name}/{output_key} successfully.")
    logging.info(f"Uploaded to s3://{bucket_name}/{output_key}")
except Exception as e:
    print(f"❌ Upload failed: {e}")
    logging.error(f"Upload failed: {e}")
```

```
✅ Extracted key phrases and topics JSON are uploaded to s3://nlpprojectamirhossein/json/extracted_key_phrases_and_topics.jso
n successfully.
```

5. Creating the dashboard

(Go to top)

Use this section to create the dashboard for your solution.

In [9]:
```
# ------------------- Flask Dashboard Deployed on EC2 (Ubuntu) -------------------

from IPython.display import HTML, display

url = "http://54.162.24.54:5000/"

display(HTML(f"""
<h5>⚠️ Flask Dashboard should be run first in EC2, Then you can open dashboard by clicking on below link </h5>
<a href="{url}" target="_blank" style="font-size: 16px; color: blue;">Click here to open the dashboard</a>
"""))
```

⚠️ **Flask Dashboard should be run first in EC2, Then you can open dashboard by clicking on below link**

Click here to open the dashboard