CS410 Text Information Systems
Final Project Report

1. Team Members:

Name	Netid	
Manasa Gangaiah	manasag3@illinois.edu	Captain
Sudhir Ponnachana	sudhirp2@illinois.edu	

2. Overview:

Our project is "***SeekFrame***". In this project we plan to build a search engine that can seek to right video segment/location based on query words. This would allow user to type in a query and provide a ranked list of videos and when user selects the link, the video will be seeked to the right segment/location based on the query words.

3. Source Code:

File Structure:

```
Project/
|--Data/
|     |--Subtitles/
|     |        |--1.vtt
|     |        |--2.vtt
|     |        |--3.vtt
|     |        |--4.vtt
|     |        |--5.vtt
|     |        |--6.vtt
|     |--Videos/                    → These videos are not on git repo due to size from the link below*
|     |        |--1.mp4
|     |        |--2.mp4
|     |        |--3.mp4
|     |        |--4.mp4
|     |        |--5.mp4
|     |        |--6.mp4
|--website/
|     |--images/
|     |        |--bg.jpg
|     |        |--logo.png
|     |        |--search_glass.jpg
|     |--index.html
|     |--search.html
```

|--generatecorpus.py
|--index.py
|--Readme.md
|--CS410ProgressReport.pdf
|--CS410_TextInformationSystems_ProjectProposal.docx

***Video link:** https://drive.google.com/drive/folders/1ReH5kzwQUITEsrJIviIb8G7C0M42eUHU

File Description:

- **Data Folder:** It contains 2 folders “Subtitles” and “Videos”.
  - **Subtitles:** It contains .vtt files for corresponding videos in the Videos folder. These files are named numerically “1”, “2”, and so on.
  - **Videos:** It contains mp4 files for corresponding .vtt file in Subtitles. These files are named numerically “1”, “2”, and so on.

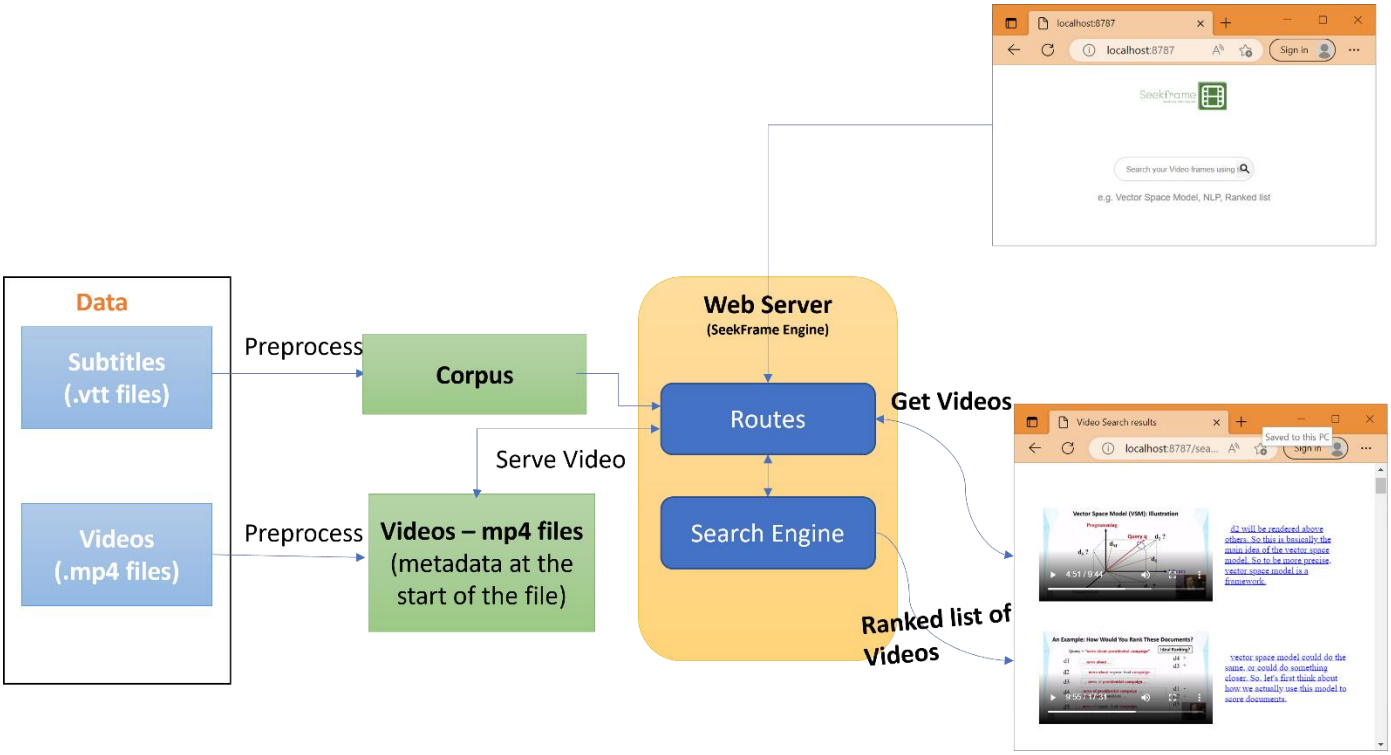
- **website:**
  - **images/**
    - **bg.jpg:** Background image
    - **logo.png:** Project logo image
    - **search_glass.jpg:** Search glass image
  - index.html: Main webpage where in user can enter query words.
  - search.html: Second webpage to display ranked videos based on query words.

- **generatecorpus.py:** This file is used for preprocessing the data. Each .vtt files in the Subtitles folder is read and parsed to generate a common corpus.txt file.

- **index.py:** This file is server file. This file reads the corpus.txt file generated and creates data frame to maintain Video id, Video Seek Time, Text. This file receives the query string from the website (front end used by user) and runs BM25 algorithm to get scores. Sends the results to the query to the front end(website) to display.

- **README.md:** Description of the project proposal.
- **CS410ProgressReport.pdf:** Description of the project progress.
- **CS410_TextInformationSystems_ProjectProposal.docx:** Description of the project proposal.

4. Architecture Block Diagram:



5. Implementation Details:

- **Data Description:**

  Two types of data are used for this project:

  i. Subtitle files that are associated with each video. These files are available in .vtt format.

  Snapshot of the data from .vtt file,

```
4
00:00:15.210 --> 00:00:17.860
In particular we're going to talk
about the using machine learning
```

  Data in vtt file corresponds to "Line number, Start Time→EndTime, Text Description."

  ii. MP4 videos (associated with the .vtt files are used).

  For easy maintenance, the files and videos are named numerically.

  For example, 1.vtt and 1.mp4. You can add any number of vtt files and its corresponding video in mp4 format.

- **Data Preprocessing:**
  - **Data Cleaning:**

    Words like " WEBVTT", "[MUSIC]", "[INAUDIBLE]", "[webvtt]" are removed.

    Each line number in the file is not required, so it removed.
  - **Parsing:**

Five lines from each vtt is parsed to make a line in corpus.txt. Start time in seconds for this line is saved. File number is used as Video id.
Format of the corpus.txt,

```
VideoID Time String
```

Snapshot of the corpus.txt,

```
1 7  This lecture is about the Learning to Rank. In this lecture, we are going to continue talking about web search. In particular we're going to talk
```

- **Video:** Download the code from [GitHub - danielgtaylor/qtfaststart: Quicktime atom positioning in Python for fast streaming] and run this command from command prompt for each video to make sure the metadata is placed at the beginning.

```
python -m qtfaststart 6.mp4
```

```
Removing free atom at 891192 (8 bytes)
Patching stco with 31499 entries
Patching stco with 31499 entries
Writing output...
```

- **Libraries and Utilities:**
  - **Python libraries:**
    requests_html, sys, pandas, os, numpy, genism, bottle, BM25Okapi
  - **Utilities:**
    qtfaststart to make sure videos have metadata at the beginning.

- **Implementation and Algorithm:**

corpus.txt file is read and dataframe is created with these columns
'Index': Vid from the corpus.txt file
'Time': Start time in seconds for the String in the line
'Strvalue': Original String(line) in corpus.txt
'CleanStrvalue': String(line) with stop words removed.
'relevance': Relevance value for the video for the line. For future enhancement.

Stop words: Standard stop words from genism package is used.

BM25 algorithm is used for calculating scores for the query search. Scores are sorted and the top 5 results are used for displaying the videos as the search results.
Whenever a user plays a video, the click count is added to the relevance in the dataframe. This value will be used to adjust the BM25 scores for future scoring(reranking).

- **Web Server:**

Bottle is used for running the backend server. Bottle routes are used for communication between the front end and back-end logic.

- **UI Details:**

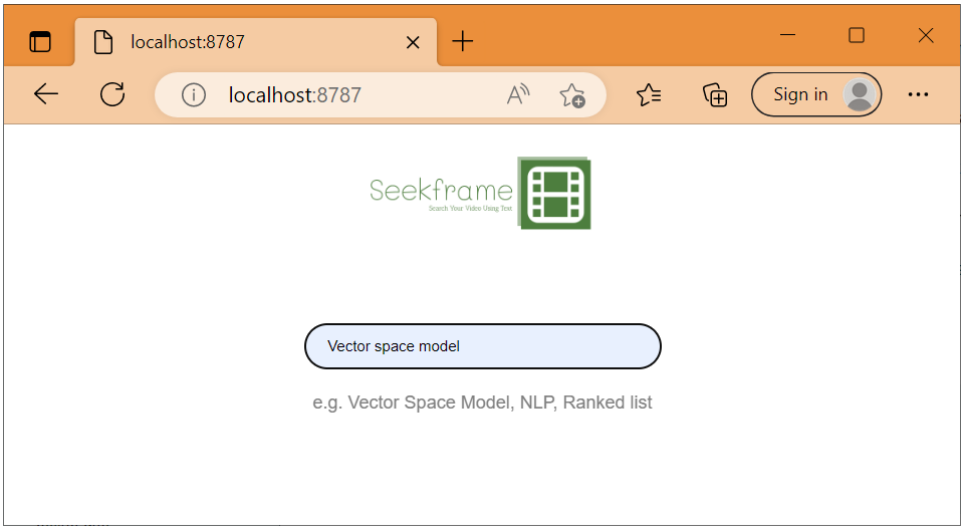
  In index.html file, search input text is used to provide user to enter the text to search.
  Once the results are obtained from the server, the videos are displayed with appropriate seek location of the videos to be played along with the original string for the video frame seeked.

6. Test Cases:

These test cases were run,

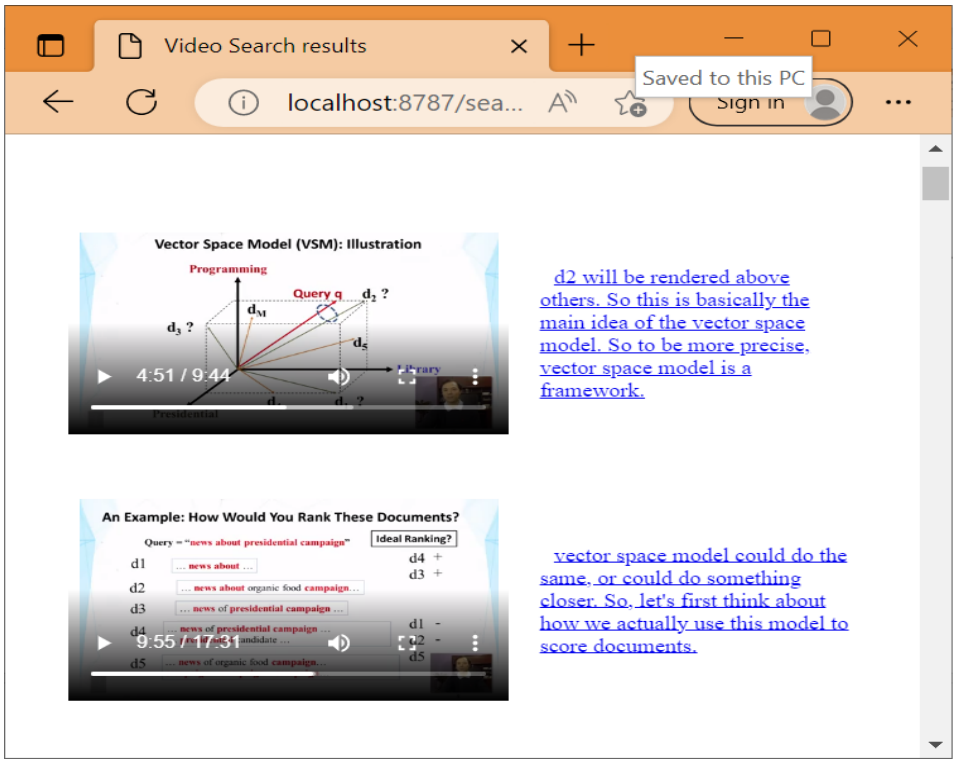
1. Enter query words in the search box. It should list all the videos with matching query words with proper seek time

Input:



Output:

You can see below that the video is offset to the location close to where the query word was specified

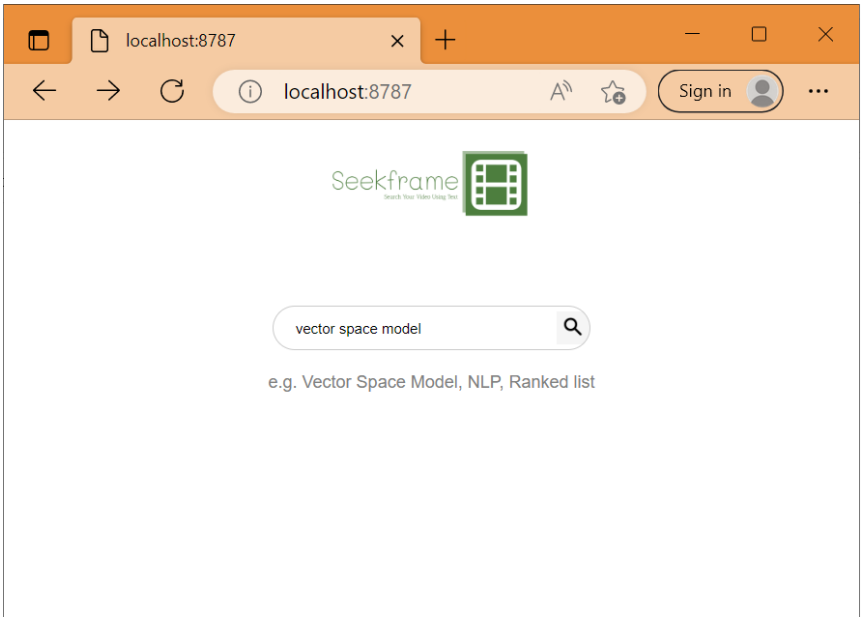


d2 will be rendered above others. So this is basically the main idea of the vector space model. So to be more precise, vector space model is a framework.

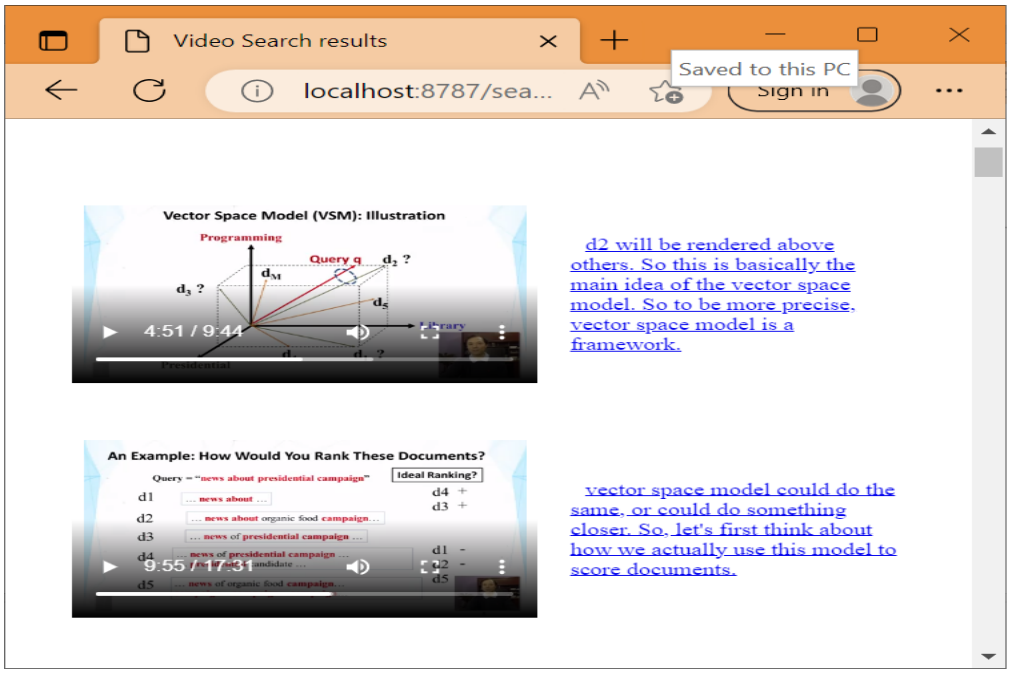
vector space model could do the same, or could do something closer. So, let's first think about how we actually use this model to score documents.

2. Enter a query in upper case/lower case. Search will result in the same search items with proper seek time.

Input:



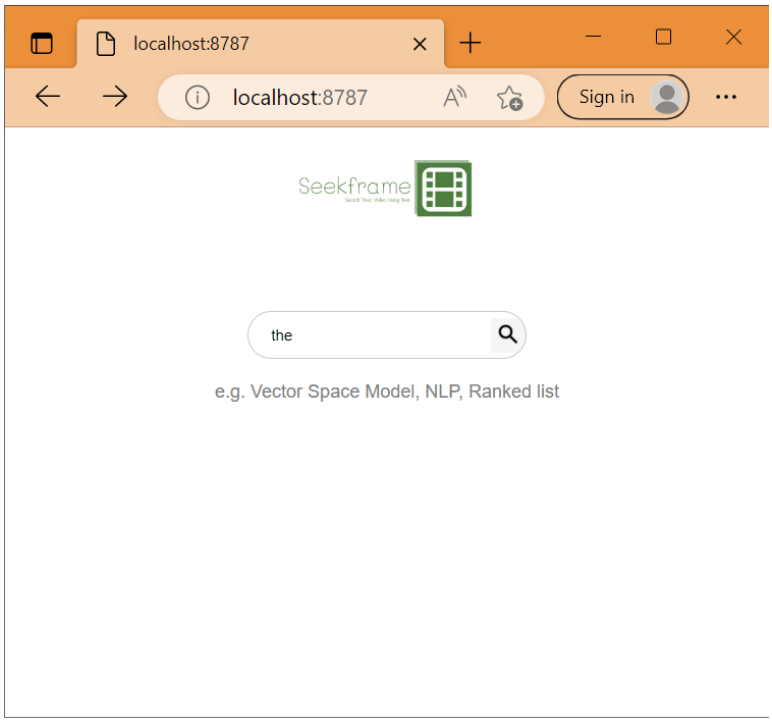
Output:



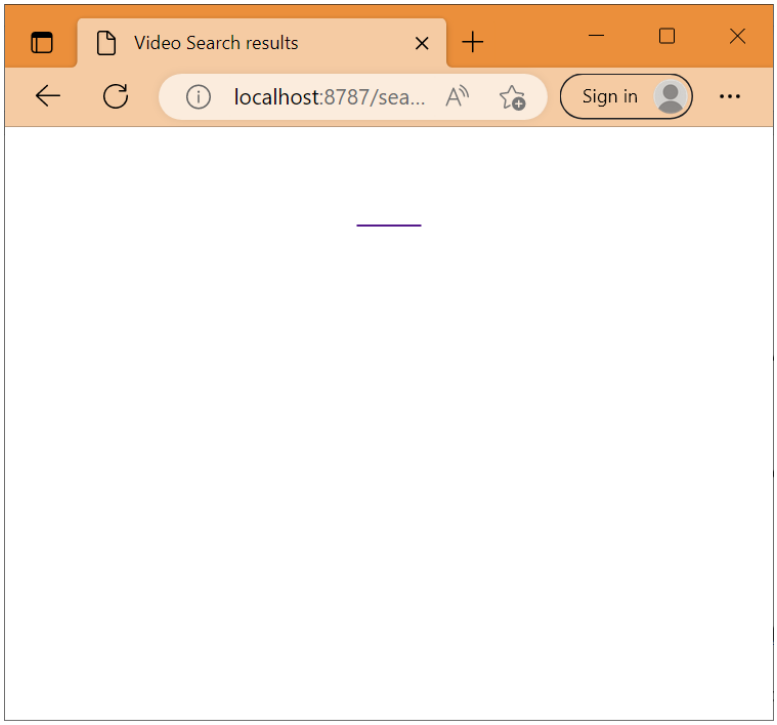
3. Enter a stop word. Since we are filtering out the stop words, we don't show any videos for that.

Input:

:



Output: Since we don't allow search based on stop words, the search result comes out empty



7. Setup:

Build Environment: Tested on Windows laptop with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz RAM 16.0 GB
Software Environment: Windows 10, Python 3.7 or Anaconda with python 3.7 can be used.

Steps:

1. Download code from **https://github.com/mtest-oss/CourseProject**

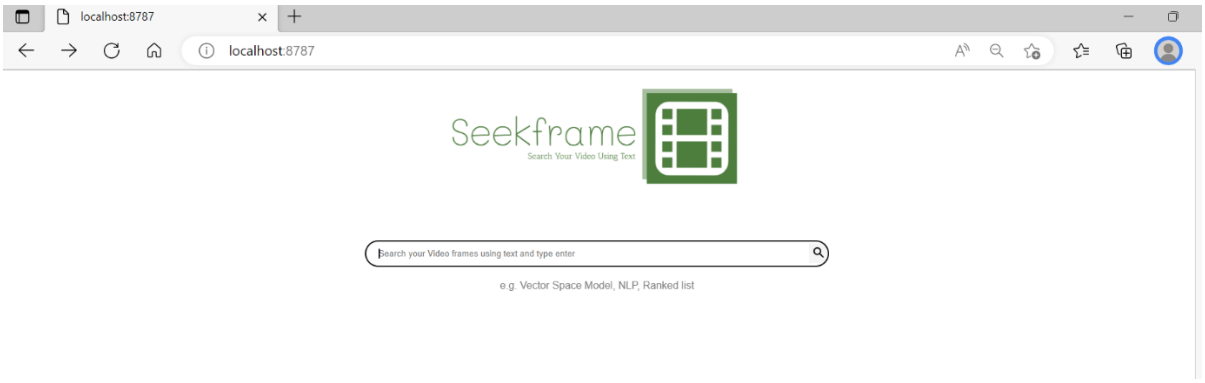
2. Copy all the videos from
   https://drive.google.com/drive/folders/1ReH5kzwQUITEsrJIvilb8G7C0M42eUHU
   into Data/Videos folder.

3. In command prompt,
   Install libraries if not installed already,

```
sudo pip3 install pandas
sudo pip3 install requests_html
sudo pip3 install gensim
sudo pip3 install bottle
sudo pip3 install rank_bm25
```

4. In command prompt,
   To generate corpus : python generatecorpus.py
   To launch the server: python index.py

5. Open Microsoft edge with this address http://localhost:8787/ to launch search web



   Enter the query words in the search text box.

6. Ranked list of videos will be listed. Each of those videos will be seeked to the query words.
   Todo: Put picture.

Contributions:

No.	Task	Team Member
1.	Preprocess the web.vtt (video subtitles) files and create a corpus data. Create a framework such that all the files stored under Data/Subtitles will be properly parsed into a corpus data.	Manasa Gangaiah
2.	Added logic to get the right seek time for each video based on the text words.	Manasa Gangaiah
3.	In-memory database for accessing the preprocessed data design and implementation.	Manasa Gangaiah
4.	Backend server code – Data cleaning, BM25 Search Engine, Basic Relevance logic implementation.	Manasa Gangaiah
5.	Front end Implementation (web interface)	Sudhir Ponnachana
6.	Integration and Testing	Manasa Gangaiah, Sudhir Ponnachana
7.	Documentation	Manasa Gangaiah, Sudhir Ponnachana
8.	Presentation	Sudhir Ponnachana
9.	Video	Manasa Gangaiah, Sudhir Ponnachana

References:

- CS 410 Text Information Systems MP 2.4 - Generic BM25 idea
- GitHub - danielgtaylor/qtfaststart: Quicktime atom positioning in Python for fast streaming

To preprocess the videos to have metadata at the beginning.

- Tutorial — Bottle 0.13-dev documentation (bottlepy.org)
- Python Programming Simple Web Server using Bottle Framework (Day 5) - YouTube
- BM25 | Build your Own NLP Based Search Engine Using BM25 (analyticsvidhya.com)