Timeline Binder: A Caption-based Approach for Creating Semantic Video Timeline

Muhammad Faran Majeed, Sadam Hussain, Abbasi Rehan Tariq
Department of Computer Science, Kohsar University Murree, Punjab, Pakistan
m.faran.majeed@kum.edu.pk sadamabbasi@kum.edu.pk rehantariq@kum.edu.pk

Alam Zeb, Khalid Saeed
Department of Computer Science, Shaheed Benazir Bhutto University Sheringal, Dir (U), Pakistan
alamzeb7@gmail.com khalidsaeed@sbbu.edu.pk

Rana Abu Bakar
Institute of Telecommunications, Informatics and Photonics (TeCIP), Scuola Superiore Sant'Anna, Pisa, Italy
engranaabubakar@gmail.com

Abstract

Video is an important multimedia type with myriad characteristics such as richer content and little prior structure. Major applications such as surveillance, medicine, education, entertainment, and sports heavily use videos. Internet protocol video is dominating the World's Internet traffic. A large number of videos are made and uploaded daily. Looking for topics of one's interest in these videos is a cumbersome task. Mostly, people are only interested in small portions of videos as opposed to the whole video. It is better to look for small but related segments of interest. Our proposed system Timeline Binder is an attempt to tackle this problem. For Timeline Binder, we have created our own dataset of online videos with subtitles. Based on those subtitles' files, a concept vector is obtained. If a concept vector has a match for a user's query tag, then timed segments for all related videos to that matched concept in the concept vector are combined into a single timeline thus making all related segments into a single video. This research presents performance measurements between Timeline Binder and other available applications such as Google2SRT and NR8. The results show that Timeline Binder performs better than Google2SRT and NR8.

Keywords— Captioning, Google2SRT, multimedia semantics, NR8, Timeline Binder.

I. INTRODUCTION

A) Overview

Multimedia is a blend of computer-controlled text, graphic art, photographs, sound, animation, and video elements. It becomes interactive multimedia when an end-user controls what and when the elements are delivered whereas when a structure of interconnected elements is provided to navigate through interactive multimedia, it becomes hypermedia [1]. In multimedia, a computer presents and combines text, graphics, audio, and video with links and tools that allow the user to navigate, interact and communicate [2]. The power of multimedia messages is due to the way people retain information. Researchers have found that people retain only 20% by hearing, 40% by seeing and hearing, and 75% by seeing, hearing, and practicing[3].

Multimedia unifies numerous separate products of human communication (the legacy media forms) that could not be combined, into a single channel of expression and delivery, therefore, it can be thought of as a super-medium of sorts [4]. Multimedia is applicable whenever someone is using any type of electronic information at a given interface. Multimedia is used in businesses, schools, homes, public places, and virtual reality. Digital video is an important element of multimedia, and it acts as a powerful tool to get computer users closer to the real world as with all multimedia elements, it closely represents the real world [1]. Globally, Internet protocol (IP) video traffic was 73% in 2016 and now it is 90% of all IP traffic by 2022[5]. Video content equal to a million minutes will cross the network every second by 2025[6].

Theoretically, multimedia and the Semantic Web match each other well. On the one hand, Semantic Web helps by providing a bunch of languages and technologies to annotate web resources thus enabling machine processing of metadata which describes the semantics of web content. On the other hand, multimedia applications need metadata descriptions of their media items to help in the search and retrieval, processing, and efficient presentation of multimedia information[7].

Annotation is a note added as a comment or explanation which helps the user's readability. Semantic annotation is a specific metadata generation and usage schema used to enable new methods for accessing information and to extend the existing ones by attaching additional information to different concepts such as people, places, things, and organizations. As opposed to classic text annotations, which are used for the reader's reference, machines make use of semantic ‍[8]. To manually add annotations is a costly process and needs a considerable amount of time. Also, a problem arises due to different interpretations of annotations when these are added by different people such as authors, editors, publishers, or end users. The primary drawback of such annotations is the lack of

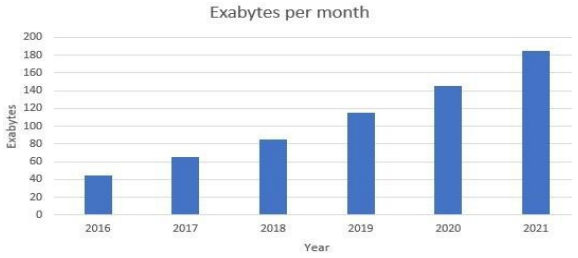

Figure 1:Global Internet Video [5]

proper syntax and semantics, as a result, computers often hardly interpret such information. Semantic Web technologies such as extensible markup language (XML), resource description framework (RDF), and ontologies are needed to create new and enrich existing annotations as there is a huge number of multimedia metadata standards and formats, and they are incompatible[9].

Video is a significant type of multimedia information that has myriad characteristics such as richer content, a huge amount of raw data, and very little prior structure which makes the indexing and retrieval of the videos difficult [10]. Major applications such as surveillance, medicine, education, entertainment, and sports heavily consume video. Searching for the segments required in this large data on the Internet can be challenging. Therefore, numerous video retrieval systems have been introduced for this intent. Semantic-based video retrieval is the new trend in video retrieval systems that intend to retrieve video clips based on semantic content[11].

B) Importance of Multimedia in Web 2.0

Web 2.0 is the second generation of Web which is more interactive and dynamic as opposed to static HTML pages. It is also referred to as the wisdom Web, participative Web, and people-centric Web. Web 2.0 utilizes the WWW by making it more interactive and collaborative. It encourages social interaction and collective wisdom. It empowers WWW and engages users more effectively. Web 2.0, being an umbrella term refers to multiple Web technologies such as wikis, blogs, really simple syndication (RSS), and mashups[12].

The main feature of Web 2.0 is its dynamic platform which enables users to generate their own experience as opposed to static pages which can only be viewed. Users' experience is enriched by the contents they supply and the interaction they make with the site. Another feature of Web 2.0 is to present the Web as a platform that enables Web-based applications to run on the Web, independent of the actual underlying operating system. Web as a platform is a top trend of Web 2.0. However, the growth of multimedia content is also a developing trend. Technological advancement has caused increased media content. One of the future challenges for WWW is how to organize a large amount of online media content and how to deliver it to the user[13].

As digital content is being created at a rapid speed by individuals and organizations through the use of different technologies, multimedia roles on the Web are guaranteed to become significant. With the convergence of rich media with Web 2.0, new solutions are needed for newly arising requirements. A key challenge for the richer media Web platforms is how to present media. Rich content on the Web platform along with rich content sources accessible through Web platform APIs is the key to discovering new value through the collaboration of both media and content stream [13].

With the arrival of new technologies, the amount of data generated is enormous on the world wide web (WWW). Large numbers of videos are made and uploaded daily to WWW servers. Looking for topics of one's interest in these videos is a cumbersome task. Moreover, mostly we are only interested in small portions of video segments as opposed to the whole video. Apropos of user query which contains a tag, our proposed system the Timeline Binder will generate different timelines from online videos matching those tags and will blend those smaller portions of videos into a single fused timeline thus populating a single timeline video.

C) Scope and Limitations

- Our proposed system will help users find chunks of videos along with captions thus saving them time.
- Once the proposed system is implemented for query tags in the English language, it may be extended to other languages.
- Once established for a small domain of videos, our proposed system can scale to larger domains.
- It will make watching online videos more meaningful and a time-saving process.
- Our proposed system will work with online videos to save video download time.
- Our proposed system only works with videos that are already annotated.
- To annotate videos, a huge amount of human work is required.
- Initially, our proposed system will work with a limited number of videos.

D) Our Contributions

- A dataset consisting of YouTube videos' URLs is manually defined.
- A dictionary of compound words is created to differentiate compound words in the concept vector.
- To develop the proposed system, algorithms are developed.
- A web application is developed for achieving the goals of the proposed system.

Our proposed system is based on a web application that

supports multiple technologies such as Java Servlets, JSP, and JavaScript. The code is open source, enabling YouTube videos search, searching across many videos, making one video out of many videos, and can also work as a stand-alone application.

II. RELATED WORK

Due to the enormous amount of multimedia available today, the need to retrieve multimedia according to one's needs has always been there. The most relevant and important applications and browser extensions presented are related multimedia retrival.

A) Google2SRT

There is no simple way to download YouTube subtitles as they are encoded in Google's format. Why would one want to download subtitles from YouTube? There may be a video whose subtitles are not available or difficult to find or even if they are available, they are in a different language, but the same subtitles are available on YouTube. Or maybe, there is a video that one likes and is only available on YouTube with subtitles as "not embedded". Google2SRT is an open source and stand-alone tool which is used to download subtitles i.e., closed captions (CC) from YouTube as well as Google Video and change them to SubRip Subtitle (SRT) which is a standard format and is supported by most media players. By using other software, SubRip Subtitle is converted to other formats such as SubStation Alpha (SSA) or MicroDVD (SUB).

Google2SRT is short for "Google to SRT", which means, from Google's format to SRT format. Since YouTube is owned by Google, therefore, it has also inherited the subtitles format. Google2SRT is free software that is licensed under the GNU public license (GPL). This can be copied, modified, and downloaded as long as the source code is distributed and, if the Google2SRT source code is modified or reused, also the new source code and the same conditions may still apply[14].

B) YouTube Subtitle Search

At times people watch videos on YouTube such as lectures or documentaries but afterward, forget what the professor said about some topic or what was that specific detail in the documentary. As a result, one skim through hour-long videos to look for that topic which results in time wastage and at times cause frustration. YouTube subtitles search can help in this regard, one can effortlessly find the spot in the video with its help. It is a Google Chrome extension that searches YouTube subtitles and shows the exact spot in the video. It works with all languages supported by subtitles. It works on YouTube videos with Subtitles enabled, if no subtitles for the video are found, a notification is shown accordingly. The quality of the search results depends on the quality of the subtitles attached to the video. Since all YouTube videos are not captions enabled, therefore, it does not work with all YouTube videos[15].

C) Invideo for YouTube

Invideo for YouTube helps users to search inside a YouTube video and skip right to one's favorite parts. Invideo is here to save one's time by jumping to the exact part one is interested in. It only works with the "new YouTube" which is accessed at "https://youtube.com /new". It is a Google Chrome extension that enables one to search through YouTube videos. It searches YouTube Videos such as lectures, movies, or talks by skipping to specific parts thus saving one's time as well as energy[16].

Table 1. Comparison of the applications and browser extensions used in multimedia content and video captions retrieval.

S.#	Application / Extension	Architecture / Tools	Pros.	Cons.
1.	Google2SRT [14]	Java	Open source, stand-alone	Only downloads Closed Captions (CC) from YouTube
2.	BriefTube [15]	Google Chrome Extension	Instant summary, word cloud, video search	Works only with English videos having subtitles and HTML5 format, only the first half of every video is free
3.	YouTube SubtitlesSearch [16]	Google Chrome Extension	Search across YouTube videos	Does not yet work with YouTube material de-sign, only searches through YouTube subtitles Works only with "new YouTube" https://youtube.com/new
4.	Invideo for YouTube [17]	Google Chrome Extension	Search across YouTube videos	Used only for subtitles addition
5.	Substital [18]	Google Chrome, FirefoxExtension	Let's easily add subtitles to videos online, supports most videos found online	Only search through YouTube subtitles
6.	YouTube CaptionsSearch [19]	Google Chrome Extension	Allows search for any word/phrase within a YouTube video	Videos segments need to be manually selectedto be spliced
7.	NR8 [20]	Web application	Splice together any clips from one or more YouTube videos	Works with already annotated videos currently works with a limited number of videos
8.	Our proposed system (Timeline Binder)	Web application, Java, Servlets, JSP, Javascript	Open source, enabling YouTube videos search, search across many videos, making one video out of many videos, stand-alone	

D) BriefTube

BriefTube is a Google Chrome ex- tension that not only offers the ability to search through a video based on its transcript but also generates a word cloud based on the frequency of words in that video, the more frequent words are shown bigger as compared to the less frequent ones. It also shows an instant summary such as how many times a specific word has appeared in a video. For BriefTube to work, the video's length must be longer than 5 minutes. It only supports videos that have English subtitles. Also, it only works with videos that have HTML5 format [14].

E) Substital

Substital is a browser extension that lets one make videos more accessible online by adding subtitles. Whether it is for entertainment, language learning, or making videos accessible to people with hearing problems, Substital is agood solution. It works with both Google Chrome and Firefox browsers. It is available on the Chrome Web Store as well as Mozilla Addons. It integrates well with the major video platfroms such as YouTube, Netflix (only is Chrome for now). Dailymotion, and Vimeo just like if it was a native feature of these platforms, still, it is not limited to these major video platforms. It supports all videos found online, either as a custom video player or as a partial integration. The automatic character encoding detection makes sure characters always display nicely even in Greek or Arabic[17].

F) YouTube Captions Search

YouTube captions search is like “CTRL + F” on documents but for YouTube. It is a Google Chrome extension that searches words or phrases within YouTube videos. It only searches through those YouTube videos that are provided with subtitles. While searching for some specific word or phrase, a userenters that word into a search box and as a result, a complete list of that word along with its corresponding timeline in the video is presented. Clicking on any link in the list, a video is played from the corresponding timeline, thus ithelps users by saving their time as well as energy[18].

G) Nr8

Nr8 is a proprietary Web application that splices together any clips from one or more YouTube videos into a single video known as "Narrative" which can be shared instantly. To make a Narrative, one has to manually select portions of videos. Alongside video clips, one can also add images so the result as a Narrative is a combination of video clips and images. One can add up to a maximum of 8 individual videos in order to make a single video as a narrative[19].

These applications and browser extensions are further elaborated by comparing them against each other in Table 1 which compares them from different perspectives.

III. PROBLEM STATEMENT

One of the popular types of entertainment in recent years is watching online videos. Online video-on-demand (VoD) services allow subscribers to watch different types of video content such as TV shows, music, live streams, movies, and user-generated videos. It has been observed that in the past few years, most of the Internet's traffic is caused by video

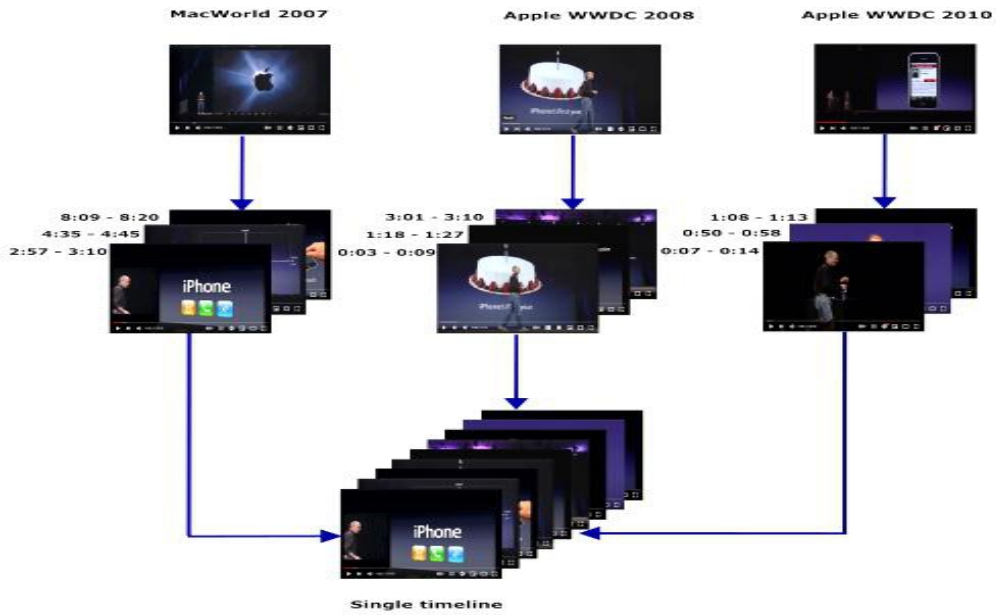

Fig. 2. Segments from different videos are extracted and combined into a single timeline.

streaming and it keeps on increasing every year [20]. With the rise of the Internet and increase in user-generated data, and the commonness of tools and technologies such as cell phones, digital cameras, and social media, a significant amount of multimedia data is being regularly generated. In recent times, the prevalence of affordable commodity digital cameras and camcorders has increased user-generated media content. In the early times, media storage was used on a small scale, usually in kilobytes. Now, the scale has gone up and data is being used at the scale of a terabyte or even petabyte [14].

Since the first video on YouTube "Me at the Zoo", uploaded on April 23rd, 2005, there are over 1.9 billion logged-in users that visit YouTube per month, and every day over a billion hours of video are watched and billions of views are generated [21]. In 2021, the amount of video content crossing global IP networks each month will be so high that it would require 5 million years to watch it. Video content equal to a million minutes will cross the network every second in 2021. Figure 1 shows how much video content is crossed through the Internet monthly [7]. Keeping in view the incredibly large number of online videos, one would naturally be choosy while selecting videos. Our proposed system Timeline Binder is an attempt to address this problem.

Going through a whole video just because one is interested in a few seconds' segment results in a waste of time and energy. To further elaborate, imagine someone is looking for a topic or issue of interest. One would go through different videos one by one and would manually adjust the timeline of each video according to one's needs. Ideally, one would wish to see segments of interest in a single place instead of being scattered across different videos. Thus, making it possible to watch a single video of full interest rather than 9, or 10 videos of partial interest which would end up saving time and energy.

Our research is an attempt to combine related segments across different online videos into a single timeline. Based on the user's query tags, our proposed system TimeLine Binder will select all videos containing those tags as captions in a specific domain. Consequently, segments matching captions will be extracted and added into a single timeline thus making a single video of interest. Our dataset consists of Apple's launch events. We have uploaded the source code to GitHub [22].

IV. METHODOLOGY

This section describes the methodology for our proposed system. It also discusses an overview and design of the proposed system. In the end, the tools used are mentioned.

A) System Overview

Most online videos consist of poorly filmed, long-running, and unedited content. Semantic content is generally scattered across different online videos. It is palpable that most online videos contain little semantic contents in which one is interested while the rest of the video is skimmed to instantly get to the semantic content or skipped entirely. One way to address this problem is to summarize the video through the provision of a short video summary of the whole video [23]. Another approach would be to combine video segments with similar semantic content across many online videos. It will make it possible for users to watch all video segments of their interest in one place thus eliminating the need to skim through long videos or skip them entirely. This is exactly what our proposed system tries to accomplish.

Our research is a step forward toward multimedia retrieval which will combine small but related chunks of multiple videos into a single timeline thus eliminating the need to skim through the video. Figure 2 shows the formation of a single timeline from different segments.

B) System Design

We are going to design a web-based application that will target online videos to achieve the functionality i.e., combining different online video segments into a single fused timeline. For this to work, a target server such as YouTube or Daily- motion needs to be defined. Once a target server is defined, a dataset containing specific domains of videos needs to be specified. The target server, dataset, and other related topics are discussed in detail next.

1) Target Server

Our proposed system uses YouTube. As it is a big name in online videos. Also, YouTube offers flexibility through the IFrame player application programming interface (API). YouTube video player is embedded in websites and is controlled using JavaScript functions through the usage of IFrame player API. By using JavaScript functions of the API, videos can be queued for playback, played, paused, or stopped. One can also adjust the player volume or retrieve information about the video being played using the API. Event listeners can also be added which are executed in response to certain player events such as player state change [5]. Our proposed system uses IFrame player API for slicing clips from different videos and then combining them into a single fused timeline.

Algorithm 1: SRT file modification pseudocode.

Input: A list of video IDs as videoIDs
Output: A modified SRT file as the file
1: define List of Strings as videoIDs
2: videoIDs ? GetVideoIDs
3: **for all** videoID ? videoIDs **do**
4: file ? SRTFile(videoID) {SRT file name is based on video ID}
5: remove style tags from the file
6: remove special characters from the file
7: remove stopwords from the file
8: save the file
9: **end for**

YouTube presents a rich platform for video captions or subtitles. Also, YouTube provides auto subtitle functionality for videos in supported videos. Our proposed system works only with those videos that have captions, captions that are explicitly provided for the videos as well as automatically generated by YouTube. The YouTube features make it an easy choice to select as the target server for our proposed system.

2) *Videos Domain*

Our proposed system is in its early stage; therefore, it is self-evident that it will work with a limited number of YouTube videos. Currently, a limited number of videos specific to Apple's products such as iPhones, iPads, and Macs launching events have been selected which makes a dataset for the proposed system.

3) *Concept Vector*

Based on the dataset, a list of concepts or keywords is obtained from subtitle files mapped to different videos. Thus, the concept vector stores all these retrieved keywords along with their related timed-segments which means that each concept is mapped with the video within the dataset that has this concept and further with the timed-segment where exactly the concept has occurred or mentioned in the video. The concept vector is further optimized by removing unnecessary characters and stop words.

4) *Dictionary*

After removing stop words, the concept vector is further optimized by extracting compound words from single words. For this purpose, a dictionary is defined as one which contains compound words optimized for the dataset. The concept vector is compared with the dictionary, if there is a match, the matched compound word is added to the concept vector. To further elaborate, two consecutive words in the concept vector are compared with the compound words in the dictionary, if they match, the two consecutive words are added as a single compound word in the concept vector otherwise they are added as two separate words.

5) *Algorithms*

To achieve the desired functionality, algorithms have been developed. These algorithms perform various operations such as Algorithm 1 presents pseudocode for SRT file modification to optimize it for retrieving concept words, Algorithm 2 presents pseudocode for retrieval of keywords along with its related timestamp from videos in a given dataset, and Algorithm 3 presents pseudocode for picking up compound nouns from an array of available nouns.

Algorithm 2: Obtaining concepts list from SRT file pseudocode.

Input: A list of video IDs as videoIDs
Output: A list of concepts as concepts

```
1: for all videoID ∈ videoIDs do
2:         fileName ← GetSRTFileName(videoID) {Get file name based on
           video ID}
3:         scan ← file(fileName) {Get file based on the file name and scan
           it}
4:         while scan.hasNext do
5:         line ← scan.nextLine
6:         if line 6= null then
7:         if the line matches the timestamp format then
8:         timestamp ← line
9:         else
10:        line ← line.removeWhiteSpaces
11:        declare an array aKeywords
12:        aKeywords ← line.split
13:        declare a List aListKeywords
14:        aListKeywords ← GetCompoundNouns(aKeywords) {if
           aKeywords contains compound nouns specific to our domain, get
           them}
15:        for all keyword in aListKeywords do
16:        if timestamp 6= empty then
17:        if concepts contains keyword then
18:        for all concept ∈ concepts do
19:        if concept.GetName = keyword then
20:        if concept has video with id = videoID then
21:        get list from concept where id = videoID
22:        add timestamp to the list
23:        update concept with modified list
24:        else
25:        add timestamp to new list
26:        add new list to concept for id = videoID
27:                                  end if
28:                                  end if
29:                        end for
30:                        else
31:        create new Concept as concept with name keyword
32:        create list as newList
33:        add timestamp to newList
34:        create HashMap as newMap
35:        map newList and id using newMap
36:        update concept with newMap
37:        add concept to concepts
38:                                  end if
39:                        end if
40:                        end for
41:                         end if
42:                        end if
43:        end while
44: end for
45: return concepts
```

These algorithms are thoroughly discussed in Section IV.

6) *Tools*

We are going to develop an open-source Web application. The following tools are used in the development of our proposed system.

JavaScript: JavaScript is the programming language of HTML and WWW. It is a front-end language which means it

Algorithm 3: Compound words extraction pseudocode.

Input: An array arg[] of concepts.
An array compoundWords of compound words.
Output: A list of Strings having both single and compound words as newWords

```
1: define array of compound words as compoundWords
2: define list of Strings as newWords
3: for all compoundWord ∈ compoundWords do
4:         for i ← 0 to arg.lenght − 1 do
5:         if compoundWord = arg[i] + arg[i + 1] then
6:         if newWords doesn't contain compoundWord then
7:         add compoundWord to newWords
8:         end if 9:    else
10:        if newWords doesn't contain arg[i] then
11:        add arg[i] to newWords
12:        end if 13:  end if 14:  end for 15: end for
16: return newWords
```

works on the client side. jQuery: jQuery is a JavaScript library that is small but rich in features. It is used for the traversal and manipulation of HTML documents, animation, event handling, and AJAX. It makes things simple with an easily accessible API that works across most browsers.

Java: Java, owned by Oracle, is a well-known programming language, released in 1995. It is used in the development of ndroid applications, Web applications, desktop applications, Web servers, and games development. Java is used for the back-end development of our proposed system.

Servlets: Servlet is a small program written in Java that runs within a Web server. Servlet gets requests from Web clients across hypertext transfer protocol (HTTP) and responds accordingly. JSP: Java server pages (JSP) is a web technology that helps developers to create dynamic web pages based on HTML, XML, and other document types.

Apache NetBeans: NetBeans is an open-source, integrated development environment (IDE). It runs on several platforms such as Windows, macOS, Linux, and Solaris. Applications are developed from a set of modular software components known as modules.

Apache Tomcat 9: For our proposed system, Apache Tomcat 9 is used as a server.

Inkscape: Inkscape is an open-source, vector graphics software. It runs on different platforms such as Linux, Mac OS X, and Windows. It is used to design figures for our proposed system's documentation.

A) Proposed Scheme

Our scheme works as follows.

- First, a domain of videos is selected as a dataset.
- Subtitles file for each video in the dataset is extracted.
- Subtitles files are optimized by removing stop words and unnecessary characters.
- Concepts are obtained from optimized subtitles files and are added to a list as concept vectors.
- In concept vector, compound words are identified with the help of a dictionary which is discussed in Section III-B4.
- Each concept is mapped with its related video(s) and timed segments.
- If the user query tag(s) gets a match in the concept vector, a list of corresponding videos along with timed segments for the matched concept is obtained.
- All selected segments are populated into a single timeline.
- In the end, a single video is composed of many semantically related segments and is presented as an output to the user.

This is also represented in Figure 3 which shows the flow diagram of the proposed system.

V. RESULTS & DISCUSSION

This section represents an implementation of the proposed system through a stepwise approach. The implementation work has been divided into a couple of modules i.e., user-end and back-end. Each module is further elaborated. This section also represents a comparison between our developed system with other already available applications.

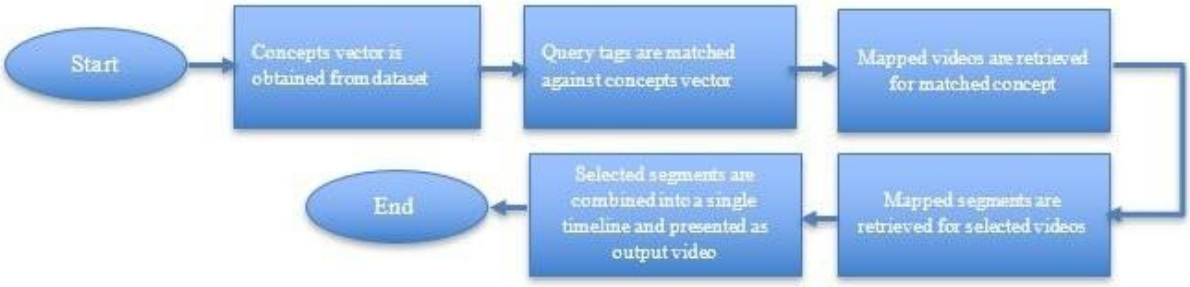

Fig. 3. Flow diagram of the proposed system.

Fig. 4. Web Application Homepage

A) Module 1: Back End

The back-end module defines the business logic of our developed system. It is further divided into sub-modules as follows.

I) Selection of Dataset

Selecting and preparing a dataset is the first step in our developed system. A uniform resource locator (URL) for YouTube videos is manually added to a list that makes the desired dataset. Based on these URLs, subtitles files are extracted.

2) Extraction and Modification of Subtitle (SRT) Files

Once a dataset is defined, for each video URL a subtitle file in SRT format is extracted. To extract subtitle files based on URL, Google2SRT code is used. It is an open-source, standalone application which is discussed in detail in Section II. Once an SRT file for a given YouTube video, based on its URL, is obtained, it is further modified to make it suitable for keyword extraction by removing stop words, extra white spaces, and other unnecessary characters. Algorithm 1 presents pseudocode for SRT file modification to optimize it for retrieving concept words.

Table II Comparison between applications in terms of total video support and access time.

S. No	Application	No. of Videos	Dataset Access Time	Videos Access Time (seconds)
1	Google2SRT	Tested up to 20	36	N/A
2	NR8	Maximum 8 videos	N/A	116
3	Timeline Binder	Tested up to 20	34	15

3) Extraction of Keywords (Concepts)

Once an SRT file is modified as discussed in Section IV-A2, keywords are extracted from already optimized SRT files. Finally, a list of concepts is obtained, and for each concept, a list of videos that have the concept. Also, for each video, a list of "start" and "end" times is obtained according to the timed segments corresponding to the concept. Algorithm 2 presents pseudocode for retrieval of keywords along with its related timestamp from videos in each dataset.

Once a list of concepts is obtained as a concept vector, it is further optimized by identifying compound words. Algorithm 3 presents pseudocode for picking up compound words from an array of available words.

4) Searching Video Segments

Once a final list of concepts is obtained, a user query in the form of a tag is searched in the concepts list and if a match is found, segments containing the searched keyword across all videos in the dataset are combined into a single timeline thus forming a single video. The video is then presented to the user. In the final video, the user can still search through the navigation bar of the video player.

B) Module 2: User-End

While the back end contains the business logic of our developed system, it is the user end that enables users to search for video segments based on query tags. The user interface consists of webpages such as the homepage, player page, dataset page, about page, and contact us page.

Fig. 5. Web Application Player Page

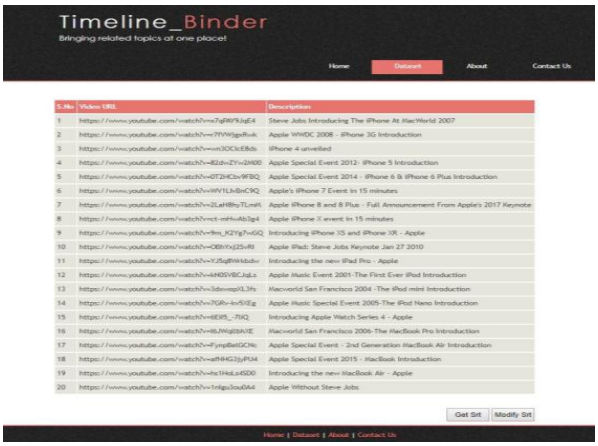

Fig. 6. Web Application Database Page

1) Home Page

The home page is the main page which has a text box and a button. A user writes a query tag into the text box, if the query tag is not found, a corresponding message is shown otherwise the user is taken to the Player page where the user is presented with a video based on the searched query tag. Figure 4 shows Home Page.

2) Player Page

The player page contains a media player which shows a video based on the query tag. The player page gives the ability to play and pause the video. Also, one can search inside that video by moving the seek bar. Figure 5 shows the Player page.

3) Dataset Page

The Dataset page shows URLs of YouTube videos used as a dataset. This page enables users to obtain SRT files based on provided URLs. Once SRT files are obtained, these can further be modified through this page. Figure 6 shows the dataset page.

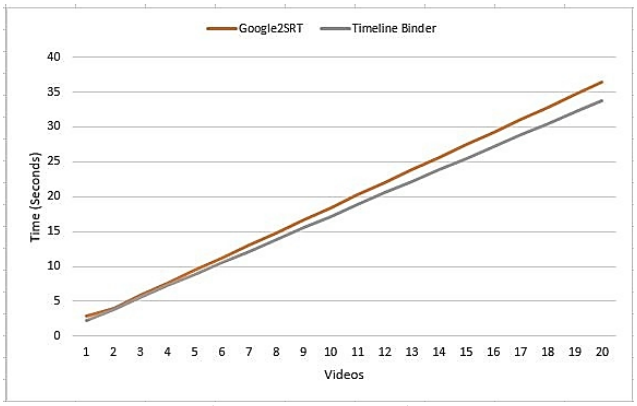

Fig. 7. Dataset Access Time.

C) Performance Measurement

We have other architectures that include invideo for YouTube, YouTube captions search, YouTube subtitles search, BriefTube, and Substital but no comparison could be made since they work on different parameters. Table 2 shows a comparison between applications in terms of dataset access time and video access time.

Timeline Binder could broadly be divided into two modules: 1) retrieval of caption files based on the dataset and 2) timeline creation from the dataset based on user tags. Google2SRT also retrieves caption files based on video URLs [14] while NR8 creates a single timeline called "Narrative" from multiple videos [20]. This means that module 1 works like Google2SRT while module 2 works like NR8, therefore, these applications cannot be directly compared however Timeline Binder is compared separately against Google2SRT as well as NR8. In Section IV-C1, Timeline Binder is compared against Google2SRT while in Section IV-C2 it is compared against Nr8.

1) Dataset Access Time

Figure 7 compares our developed system with Google2SRT in terms of dataset access time for a given set of video URLs. According to Figure 8, access time for Google2SRT and Timeline Binder is 2.8 and 2.2 seconds respectively when the dataset consists of only one video URL and 36 and 34 seconds respectively when the dataset consists of 20 video URLs.

2) Video Access Time

Figure 8 compares our developed system with NR8 in terms of timeline access time from an already-defined dataset. According to Figure 8, video access time for NR8 is 26 and 116 seconds when the dataset consists of only one and 8 videos respectively. Also, according to Figure 8 video access time for Timeline Binder is 12 and 15 seconds when the dataset consists of only one and 20 videos respectively.

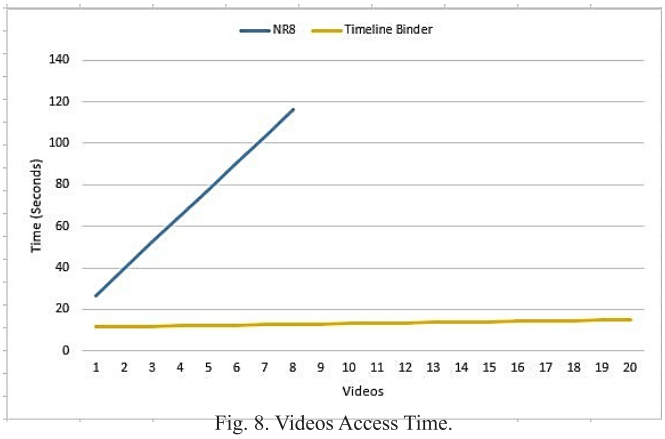

Fig. 8. Videos Access Time.

VI. CONCLUSION & FUTURE WORK

Multimedia usage is everywhere from home, school, business, and public places to VR [1]. Video is heavily used for major applications such as surveillance, medicine, education, entertainment, and sports. Searching for the elements required in this large data on the Internet can be challenging. Therefore, numerous video retrieval systems have been introduced for this intent [11]. Due to the enormous number of online videos, it will be better to look for small but related segments of interest as opposed to a whole video[14][17][18].

Our proposed system Timeline Binder is an attempt to tackle this problem. For Timeline Binder a dataset is first defined and based on the dataset a concept vector is obtained. If a concept vector has a match for the user's query tag, then timed segments for all related videos to that matched concept are combined into a single timeline thus making all related segments in one place.

Timeline Binder is just the beginning, more work needs to be done to make it a more relative and general-purpose application. The biggest challenge is to increase the dataset i.e., the number of videos on which it operates efficiently so that system performance doesn't deteriorate, one way to achieve this is to include a database in our web application. Also, for proper modification and optimization of subtitle files i.e., extraction of only nouns from other words, natural language processing (NLP) techniques need to be implemented.

REFERENCES

[1] T. Vaughan, Multimedia: Making it work. McGraw-Hill, Inc., 2006.

[2] A. Mamasoliyev, "Types, Functions, Role, Application of Multimedia Technologies," Eurasian Journal of Media and Communications, vol. 10, pp. 1-3, 2022.

[3] K. L. Valijonovna, "Multimedia Technologies and Their Use in the System of Preschool Education," Journal of Ethics and Diversity in International Communication, vol. 2, no. 4, pp. 62-66, 2022.

[4] V. Costello, Multimedia foundations: Core concepts for digital design. CRC Press, 2016.

[5] S. Zhang, X. Wang, J. Lv, and M. Huang, "Routing and content delivery for in-network caching enabled IP network," Multimedia Tools and Applications, pp. 1-21, 2022.

[6] C. V. N. Index, "The zettabyte era: trends and analysis," ed: CISCO White Paper, 2017, p. 32.

[7] J. van Ossenbruggen, G. Stamou, and J. Z. Pan, "Multimedia annotations and the Semantic Web," in Workshop on Semantic Web Case Studies and Best Practice for eBusiness, International Semantic Web Conference, 2005, p. 8.

[8] L. Eadicicco, "Watch Steve Jobs launch the iPhone at Apples 2007 keynote," ed, 2017.

[9] T. Sjekavica, G. Gledec, and M. Horvat, "Advantages of semantic web technologies usage in the multimedia annotation and retrieval," International journal of computers and communications, vol. 8, pp. 41-48, 2014.

[10] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 797-819, 2011.

[11] M. E. Abdulmunem and E. Hato, "Semantic based video retrieval system: survey," Iraqi Journal of Science, pp. 739-753, 2018.

[12] S. Murugesan, "Understanding Web 2.0," IT professional, vol. 9, no. 4, pp. 34-41, 2007.

[13] L. J. B. Nixon, "Multimedia, web 2.0 and the semantic web: A strategy for synergy," in First International Workshop on Semantic Web Annotations for Multimedia (SWAMM), as a part of the 15th World Wide Web Conference, 2006, p. 15.

[14] "BriefTube, Instant Video Summarizer," ed: Online, 2022.

[15] "YouTube Subtitles Search - Google Chrome Extension," ed, 2022.

[16] "Invideo for YouTube - Google Chrome Extension," ed, 2022.

[17] "Substital: Add subtitles to videos online," ed, 2022.

[18] "YouTube Captions Search - Google Chrome Extension," ed, 2022.

[19] "NR8," ed, 2022.

[20] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," IEEE Communications Surveys \& Tutorials, vol. 18, no. 1, pp. 401-418, 2015.

[21] "YouTube for Press," ed, 2022.

[22] W. Zhu, P. Cui, Z. Wang, and G. Hua, "Multimedia big data computing," IEEE Multimedia, vol. 22, no. 3, pp. 96-c3, 2015.

[23] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid, "Category-specific video summarization," in Computer Vision--ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13, 2014, pp. 540-555.