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

Alam Zeb a, Muhammad Faran Majeed b,* Khalid Saeed c and Muhammad Faisal Abrar d

a Department of Computer Science, Shaheed Benazir Bhutto University, Sheringal, Pakistan
E-mail: alamzeb7@gmail.com
b Department of Computer Science, Shaheed Benazir Bhutto University, Sheringal, Pakistan
E-mail: m.faran.majeed@ieee.org
c Department of Computer Science, Shaheed Benazir Bhutto University, Sheringal, Pakistan
E-mail: khalidsaeed@sbbu.edu.pk
d Department of Computer Software Engineering, University of Engineering and Technology, Mardan, Pakistan
E-mail: abrarfaisal49@gmail.com

Abstract. Video is an important type of multimedia which has 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. Large number of videos are made and uploaded on daily basis. 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 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. In this research, we present 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: Timeline Binder, captioning, multimedia, Semantic Web, Google2SRT, NR8.

1. Introduction

This section describes the overview of the multimedia and it’s different types. It also describes problem statement and objectives for the proposed system. Scope and limitations of the proposed system are also covered in this section while research outlines are presented at the end.

1.1. 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 a hypermedia [1]. In multimedia, a computer is used for presenting and combining text, graphics, audio, video having links, and tools that allow the user to navigate, interact, and communicate [2]. The power of multimedia message 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 business, schools, home, 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 real world as of all multimedia elements, it closely represents real world [1]. Globally, Internet protocol (IP) video traffic was 73% in 2016 and it will be 82% of all IP traffic by 2021. Video contents equal to a million minutes will cross the network every second by 2021 [5].

Theoretically, multimedia and the Semantic Web make a good match for each other. On the one hand, Semantic Web helps in providing a bunch of languages and technologies in order to annotate Web resources thus enabling machine processing of metadata which describes the semantics of Web content. One the other hand, multimedia applications need metadata descriptions of their media items to help in search and retrieval, processing and efficient presentation of multimedia information [6].

Annotation is a note added as a comment or explanation which helps 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 annotations [7]. 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 missing of a proper syntax and semantics, as a result, often computers 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 are huge numbers of multimedia metadata standards and formats, and they are incompatible [8].

Video is a significant type of multimedia information that has myriad characteristics such as richer content, huge amount of raw data and very little prior structure which makes the indexing and retrieval of the videos difficult [9]. Major applications such as surveillance, medicine, education, entertainment and sports heavily consume video. Searching for the segments required in these 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 of video retrieval systems which intents to retrieve video clips based on semantic content [10].

1.2. 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 in a more effective way. Web 2.0, being an umbrella term refers to multiple Web technologies such as wikis, blogs, really simple syndication (RSS) and mashups [11].

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 which 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 to increase media content. One of the future challenges for WWW is how to organize the large amount of online media content and how to deliver it to the user [12].

As digital content is being created at a rapid speed by individuals and organizations through the use of different technologies, multimedia role on the Web is guaranteed to become significant. With the convergence of rich media with Web 2.0, new solutions are needed for newly arising requirements. 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 discover new value through collaboration of both media and content stream [12].

With the arrival of new technologies, the amount of data generated is enormous on the world wide web (WWW). Large number of videos are made and uploaded on daily basis 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 to user query which contains a tag, our proposed system the Timeline Binder will generate different timelines from online videos matching those tags and would blend those smaller portions of videos into a single fused timeline thus populating a single timeline video.

1.3. Objectives

1. To explore multimedia content retrieval based on user query tags from the WWW.
2. To develop a Web application based on the Semantic Web.
3. To combine segments from different videos specific to a topic and compile a single timeline video.

1.4. Scope and limitations

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

1.5. Our contributions

– A dataset consisting of YouTube videos URLs is manually defined.
– A dictionary of compound words is created in order to differentiate compound words in the concept vector.
– To develop the proposed system, algorithms are developed.

Fig. 1. Steve Jobs during the original iPhone launch event - Macworld San Francisco 2007 [13].

– A Web application is developed for achieving goals of the proposed system.

Rest of the paper is organized as follows. In Section 2, relevant work is discussed while in Section 3, methodology is proposed. Section 4 presents experimental results and finally, in Section 5, research is concluded.

2. Related work

Due to the enormous amount of multimedia available today, the need to retrieve multimedia according to one’s need has always been there. The most relevant and important applications and browser extensions are presented below that are related to multimedia retrieval;

1. Goog2SRT: There is no simple way to download YouTube subtitles as they are encoded in Google’s own 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 may be, 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 subtitles format. Google2SRT is free software which is licensed under the GNU general public license (GPL). This can be copied, modified, and downloaded as long as original source code is distributed and, if Google2SRT source code is modified or reused, also the new source code and the same conditions may still apply [14].

2. YouTube Subtitles Search: At times people watch videos on YouTube such as lectures or documentaries but afterwards forget what the professor said about some topic or what was that specific detail in the documentary? As a result, one skims 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 which searches YouTube subtitles and shows the exact spot in the video. It works with all languages supported by the subtitles. It works on YouTube videos with Subtitles enabled, if no subtitles for the video are found, a notification is shown accordingly. 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].

3. 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 as “https://youtube.com/new”. It is a Google Chrome extension which 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].

4. BriefTube: BriefTube is a Google Chrome extension 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, videos 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 [17].

5. 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 to make the videos accessible to people with hearing problems, Substital is a good solution. It works with both Google Chrome and Firefox browsers. It is available on the Chrome Web Store as well as Mozilla Add-ons.

It integrates well with the major video platforms such as YouTube, Netflix (only in 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 [18].

6. YouTube Captions Search: YouTube captions search is like “CTRL + F” on documents, but for YouTube. It is a Google Chrome extension which searches word or phrase within YouTube videos. It only searches through those YouTube videos that are provided with subtitles.

While searching for some specific word or phrase, a user enters 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, video is played from the corresponding timeline, thus it helps users by saving their time as well as energy [19].

7. NR8: NR8 is a proprietary Web application that splice together any clips from one or more YouTube videos into a single video known as “Narrative” which can be shared instantly. In order to make a Narrative, one has to manually select portions of videos. Alongside video clips, one can also add images so the end result as a Narrative is a combination of video clips and images. One can add up to maximum of 8 individ-
Table 1: Comparison of the applications and browser extensions used in multimedia content and videos captions retrieval

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Goog2SRT [14]</td>
<td>Java</td>
<td>Open source, stand-alone</td>
<td>Only downloads Closed Captions (CC) from YouTube</td>
</tr>
<tr>
<td>2</td>
<td>BriefTube [17]</td>
<td>Google chrome Extension</td>
<td>Instant summary, word cloud, video search</td>
<td>Works only with English videos having subtitles and HTML5 format, only first half of every video is free</td>
</tr>
<tr>
<td>3</td>
<td>YouTube Subtitles Search [15]</td>
<td>Google chrome Extension</td>
<td>Search across YouTube videos</td>
<td>Does not yet work with YouTube material design, only searches through YouTube subtitles</td>
</tr>
<tr>
<td>5</td>
<td>Substital [18]</td>
<td>Google chrome, Firefox extension</td>
<td>lets one easily add subtitles to videos online, supports most videos found online</td>
<td>Used only for subtitles addition</td>
</tr>
<tr>
<td>6</td>
<td>YouTube Captions Search [19]</td>
<td>Google chrome Extension</td>
<td>Allows search for any word/phrase within a YouTube video</td>
<td>Only search through YouTube subtitles</td>
</tr>
<tr>
<td>7</td>
<td>NR8 [20]</td>
<td>Web application</td>
<td>Splice together any clips from one or more YouTube videos</td>
<td>Videos segments need to be manually selected in order to be spliced</td>
</tr>
<tr>
<td>8</td>
<td>Our proposed system (Timeline Binder)</td>
<td>Web application, Java, Servlets, JSP, Javascript</td>
<td>Open source, enabling YouTube videos search, search across many videos, making one videos out of many videos, stand-alone</td>
<td>Works with already annotated videos, currently works with limited number of videos</td>
</tr>
</tbody>
</table>

Since the first video on YouTube “Me at the zoo”, uploaded on April 23rd, 2005, there are over 1.9 Billion number of logged-in users that visit YouTube per month, and every day over a billion hours of video is watched and billions of views are generated [23]. In 2021, the amount of video content crossing global IP networks each month will be so much high that it would require 5 million years to watch it. Video contents equal to a million minutes will cross the network every second in 2021. Figure 2 shows how much video content is crossed through the Internet on monthly basis [5].

Keeping in view the incredible 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 wastage of time and energy. To further elaborate, imagine someone is looking for a topic or issue of his interest. One would go through different videos one by one and would manually adjust timeline of each video according to one’s need.

Ideally, one would wish to see segments of interest at 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, 10 videos of
Our research is an attempt to combine related segments across different online videos into a single timeline. Based on user’s query tags, our proposed system TimeLine Binder would select all videos containing those tags as captions in a specific domain. Consequently, segments matching captions would be extracted and added into a single timeline thus making a single video of interest. Our dataset would consist of Apple’s launching events. We have uploaded the source code to GitHub.

3. Methodology

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

3.1. System overview

Most online videos consist of poorly-filmed, long-running, and unedited content. Semantic contents are generally scattered across different online videos. It is obvious that most online videos contain little semantic contents in which one is interested while 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 video through provision of a short video summary of the whole video [24]. Another approach would be to combine video segments with similar semantic contents across many online videos. It would make it possible for users to watch all video segments of their interest at 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 towards multimedia retrieval which would combine small but related chunks of multiple videos into a single timeline thus eliminating the need to skim through the video. Figure 3 shows formation of a single timeline from different segments.

3.2. System design

We are going to design a Web application which would target online videos in order to achieve the functionality i.e., combining different online videos segments into a single fused timeline. For this to work, a target servers such as YouTube or Dailymotion needs to be defined. Once a target server is defined, a dataset containing specific domain of videos needs to be specified. Target server, dataset and other related topics are discussed in detail as next.

3.2.1. Target Server

Our proposed system uses YouTube for online videos. YouTube is a big name in online videos. Also, YouTube offers flexibility through 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 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 [25]. Our proposed system uses IFrame player API for slicing clips from different videos and then combining them into a single fused timeline.

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 aforementioned YouTube features make it an easy choice to select it as target server for our proposed system.

3.2.2. Videos domain

Our proposed system is in its early stage, therefore, it’s obvious that it would work with limited number of YouTube videos. Currently, a limited number of videos

---

1 https://github.com/alamz7/Timeline-Binder
specific to Apple’s products such as iPhones, iPads, and Macs launching events have been selected which makes dataset for the proposed system.

3.2.3. Concept vector
Based on dataset, a list of concepts or keywords are obtained from subtitle files mapped to different videos. Thus, concept vector stores all these retrieved keywords along with its 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. Concept vector is further optimized by removing unnecessary characters and stopwords.

3.2.4. Dictionary
After removing stopwords, concept vector is further optimized by extracting compound words from single words. For this purpose, a dictionary is defined 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.

3.2.5. Algorithms
To achieve desired functionality, algorithms have been developed. These algorithms perform various operations such as Algorithm 1 presents pseudocode for SRT file modification in order 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. These algorithms are thoroughly discussed in Section 4.

3.2.6. Tools
We are going to develop an open source Web application. Following tools are used in the development of our proposed system;
1. **JavaScript**: JavaScript is the programming language of HTML and the WWW. It is a front-end language which means it works on the client side.

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

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

4. **Servlets**: Servlet is basically a small program written in Java which runs within a Web server. Servlet gets requests from Web clients across hypertext transfer protocol (HTTP) and respond accordingly.

5. **JSP**: Java server pages (JSP) is a Web technology which helps developers to create dynamic web-pages based on HTML, XML, and other document types.

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

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

8. **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.

### 3.3. Proposed scheme

Our scheme works as follows;

1. First a domain of videos is selected as a dataset.
2. Subtitles file for each video in the dataset is extracted.
3. Subtitles files are optimized by removing stopwords and unnecessary characters.
4. Concepts are obtained from optimized subtitles files and are added to a list as concept vector.
5. In concept vector, compound words are identified with the help of dictionary which is discussed in Section 3.2.4.
6. Each concept is mapped with its related video(s) and timed-segments.
7. If user query tag(s) gets a match in the concept vector, a list of corresponding videos along with timed-segments for the matched concept are obtained.
8. All selected segments are populated into a single timeline.
9. At 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 4 which shows flow diagram of the proposed system.

### 4. Results and Discussion

This section represents implementation of the proposed system through step-wise 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.
4.1. 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.

4.1.1. Selection of dataset

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

4.1.2. Extraction and modification of subtitle (SRT) files

Once dataset is defined, for each video URL a subtitle file in SRT format is extracted. In order to extract subtitle file based on URL, Google2SRT code is used. It is an open source, standalone application which is discussed in detail in Section 2.

Once SRT file for a given YouTube video, based on its URL, is obtained, it is further modified in order to make it suitable for keyword extraction by removing stopwords, extra white spaces, and other unnecessary characters. Algorithm 1 presents pseudocode for SRT file modification in order to optimize it for retrieving concept words.

Algorithm 1: SRT file modification pseudocode

Input: A list of video IDs as videoIDs
Output: A modified SRT file as file

1: define List of Strings as videoIDs
2: videoIDs ← GetVideoIDs
3: for all videoID ∈ videoIDs do
4: file ← SRT File(videoID) (SRT file name is based on video ID)
5: remove style tags from file
6: remove special characters from file
7: remove stopwords from file
8: save file
9: end for

4.1.3. Extraction of keywords (concepts)

Once SRT file is modified as discussed in Section 4.1.2, keywords are extracted from already optimized SRT files. Finally, a list of concepts is obtained and for each concept, a list of videos which have the concept. Also, for each video a list of “start” and “end” time 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 a given dataset.

4.1.4. Searching video segments

Once a final list of concepts is obtained, 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, user can still search through navigation bar of the video player.

4.2. Module 2: User-end

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

4.2.1. Homepage

Homepage is the main page which has a textbox and a button. User writes a query tag into the textbox, if query tag is not found, a corresponding message is shown otherwise user is taken into the Player page where user is presented video based on searched query tag. Figure 5 shows Homepage.

4.2.2. Player page

Player page contains a media player which shows video based on query tag. Player page gives the ability to play and pause video. Also, one can search video by moving seekbar. Figure 6 shows Player page.
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 file name and scan it}
4:   while scan.hasNext do
5:     line ← scan.nextLine
6:     if line ≠ null then
7:       if line matches timestamp format then
8:         timestamp ← line
9:       else
10:          line ← line.trimWhiteSpaces
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 ≠ 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 if
41:   end while
42: end for
43: return concepts

4.2.3. Dataset page

Dataset page shows URLs of YouTube videos used as 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 7 shows Dataset page.

4.2.4. About and Contact Us pages

About page shows short description of the developed system. Contact Us page enables users to contact. It provides contact information such as phone number and email address.

4.3. 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.

Fig. 7. Web application Dataset page.
Table 2

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

Algorithm 3: Compound words extraction pseudocode

```
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.length − 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
```

4.3.1. Dataset access time

Figure 8 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 is 2.8 and 2.2 seconds respectively when dataset consists of only one video URL while 36 and 34 seconds respectively when dataset consists of 20 video URLs.

4.3.2. Video access time

Figure 9 compares our developed system with NR8 in terms of timeline access time from already defined dataset. According to Figure 9, video access time for NR8 is 26 and 116 seconds when dataset consists of only one and 8 videos respectively. Also, according to Figure 9, video access time for Timeline Binder is 12 and 15 seconds when dataset consists of only one and 20 videos respectively.

5. Conclusion and future work

Multimedia usage is everywhere from home, school, business and public places to VR [1]. Video, being the dominant part of multimedia is also, the dominant part of all IP traffic as IP video traffic in 2016 was 73% and expected to be 82% by 2021. Video contents equal to a million minutes will cross the network every second in 2021 [5]. Video is heavily used by major applications such as surveillance, medicine, education, entertainment, and sports. Searching for the elements required in these large data on the Internet can be challeng-
ing. Therefore; numerous video retrieval systems have been introduced for this intent [10]. Due to the enormous number of online videos, it would be better to look for small but related segments of interest as opposed to a whole video. Our proposed system Timeline Binder is an attempt to tackle this problem. For Timeline Binder a dataset is first defined and based on dataset a concept vector is obtained. If a concept vector has a match for 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 at one place.

Timeline Binder is just the beginning, more work needs to be done to make it more relative and general purpose application. The biggest challenge is to increase the dataset i.e., the number of videos on which it operates in an efficient way so that system performance doesn’t deteriorate, one way to achieve this is to include 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