PDF TO VIDEO CONVERTER

Data Mining (CSE3019)
J-Component Final Review

Submitted by

Vineet Nalawade (15BCE0284)

Pragadeesh Dharsha(15BCE0707)

Under Prof. **VAIRAMUTHU S**
SCOPE, VIT Vellore

SCHOOL OF COMPUTING SCIENCE AND

ENGINEERING

APRIL 2018

TABLE OF CONTENTS

1. ABSTRACT

We communicate with pictures and images. But, in many cases, the images alone may not tell the whole story. That's where narration fits in. When used well, narration adds depth and harmony to your production. When used poorly, narration can ruin an otherwise perfect project. Narration, often called voice over (VO), is the off-camera voice that imparts the

important information that the video footage itself doesn't. Carefully constructing good, crisp narration is critical to the success of your video. If you don't start with well-written narration the end result will still sound flat and out of tune, regardless of how well the narration is delivered or recorded. Fortunately, writing effective narration isn't as difficult as it may seem. Use the software which we have created can be useful and makes learning interesting. This software which we have developed is used to convert the pdfs into a narrated video format. This makes learning simpler and interesting for the students. We have used natural language processing and google image scrapping for the video purpose. This is the first software to convert the pdf into a narrated video with pictures for each important words when it is narrated.

2. INTRODUCTION

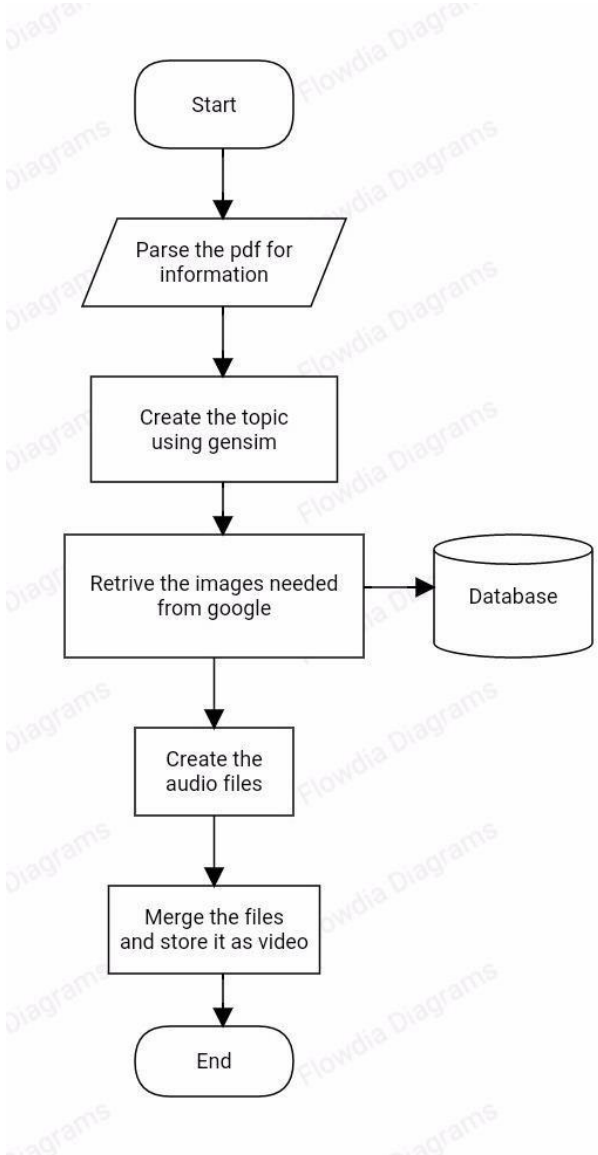
Decide how you want to use narration in your project before you start writing. Narration can do many things. How you use the VO depends upon the video itself. A typical travel video can highlight examples of some common uses of narration. Picture a small Hawaiian beach surrounded by hotels. The narrator can introduce the subject: "To some people, the place where the surf meets the sand at Waikiki Beach is a piece of heaven on earth." Narration can also impart information not obvious to the viewer. "With millions of visitors each year, Waikiki is perhaps the most famous beach in the world." Or you might use narration as a bridge between segments. "Only miles away from Waikiki is a vision few people have seen up close. Fiery balls of lava explode into the sea at the most active volcano in the world, Kilauea." Narration does many other things, too. It can tell the viewer what to look for, or it can summarize the video in a few words. Make certain that the narration matches the tone of your project. For example, if you're producing a comedic short, try using the narrator as the straight man. If you're creating a video on how to bake a wedding cake, it might be appropriate to give your VO a motherly tone. Don't wait until you've finished shooting the video to flesh out the narration. Write in advance. Put your narration into words at the same time you are scripting your visuals. A good narration written ahead of time can actually help improve your shooting

Carefully constructing good, crisp narration is critical to the success of your video. If you don't start with well-written narration the end result will still sound flat and out of tune, regardless of how well the narration is delivered or recorded. Fortunately, writing effective narration isn't as difficult as it may seem. Use the software which we have created can be useful and makes learning interesting. This software which we have developed is used to convert the pdfs into a narrated video format. This makes learning simpler and interesting for the students. We have used natural language processing and google image scrapping for the video purpose. This is the first software to convert the pdf into a narrated video with pictures for each important words when it is narrated.

3. PROPOSED WORK

The main purpose of this project is to make learning interesting for students. Maximum number of students' dislike subjects like history, geography, etc. because it is completely theory and only visualization is their imagination. So to make those students bring own interest into the education and the subject this software is created. This can help the students even in at the last moment. Even though this might not give all the details needed, this will surely help in revising the complete subject before exam.

4. ARCHITECTURE



5. PYTHON PACKAGES USED

 StringIO

This module implements a file-like class, **StringIO**, that reads and writes a string buffer (also known as *memory files*).

 Pdfminer

PDFMiner is a tool for extracting information from PDF documents. Unlike other PDFrelated tools, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.

 Moviepy

MoviePy is a Python module for video editing, which can be used for basic operations (like cuts, concatenations, title insertions), video compositing (a.k.a. non-linear editing), video processing, or to create advanced effects. It can read and write the most common video formats, including GIF.

 Gtts

This module is used to convert the text to speech using Google Text to speech convertor

 Nltk

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries

 Gensim

Gensim is a robust open-source vector space modeling and topic modeling toolkit implemented in Python. It uses NumPy, SciPy and optionally Cython for performance.

Gensim is specifically designed to handle large text collections, using data streaming and efficient incremental algorithms, which differentiates it from most other scientific software packages that only target batch and in-memory processing.

Shutil

The shutil module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal. For operations on individual files System Features.

BeautifulSoup

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Urllib

A package that collects several modules for working with URLs:

- urllib.request for opening and reading URLs
- urllib.error containing the exceptions raised by urllib.request
- urllib.parse for parsing URLs
- urllib.robotparser for parsing files

PIL

The **Python Imaging Library** adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

tKinter

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don't need to write Tcl code, but you will need to consult the Tk documentation, and

occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module tkinter provides a threadsafe mechanism which allows Python and Tcl to interact.

6. Algorithm Features

- Select the pdf to be extracted ○ Extract the pdf using pdfminer module ○ Use stemming technique to remove the stop words
- Convert the words into their real form ○ Use genism module to get the topics for the group of words ○ Send the important words and download the image and store it ○ Convert the text form the pdf to speech and store it as separate files for each line
- Create a video with all the file downloaded and the audio

7. CODE AND LIBRARIES USED

7.1 Libraries Used

- BeautifulSoup
- Requests
- Re
- Urlopen
- Request
- Os
- Cookiejar
- Json
- Image
- from io import StringIO
- io
- PDFResourceManager
- PDFPageInterpreter
- TextConverter
- LAParams
- PDFPage
- moviepy.editor
- gTTS
- nltk.corpus
- WordNetLemmatizer
- gensim
- shutil
- ffmpy

- ssl

7.2 Code:

```
from io import StringIO import

io

from     pdfminer.pdfinterp     import       PDFResourceManager,
PDFPageInterpreter from pdfminer.converter

import TextConverter from pdfminer.layout import

LAParams from pdfminer.pdfpage import PDFPage

import os import re from moviepy.editor import *

from gtts import gTTS from nltk.corpus import

stopwords  from nltk.stem.wordnet import

WordNetLemmatizer import gensim from gensim

import corpora import google_image_download as

image_downloader import shutil import ffmpy


  stop =

set(stopwords.words('english')) lemma

= WordNetLemmatizer()

_FPS=24
 f_dict = open('dictionary.txt')# from scrabble words

list scrabble_list = [] for line in f_dict:

    scrabble_list.append(str(line.replace('\n','').lower()))

f_dict.close() # print scrabble_list print ("scrabble_list

loaded!")




# PDF TO STRING
```

```
 def pdf_to_txt(fname,
pages=None):     if not pages:
        pagenums = set()
else:
        pagenums = set(pages)
     output = StringIO()
manager = PDFResourceManager()
    converter       =       TextConverter(manager,       output,
laparams=LAParams())                              interpreter     =
PDFPageInterpreter(manager, converter)
     infile = open(fname, 'rb')     for page in
PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
infile.close()     converter.close()
text = output.getvalue()
output.close     print ("conversion
from PDF to TXT done!\n")     return
text
 def
txt_to_clean(fname):
    fr = open('./txt/'+fname[:-4]+'.txt')
full_text = ''
         for line in fr:                 if line == '\n':
full_text += line + '\n'              continue
line = line.replace('\\n','')          line =
re.sub(r"[^a-zA-Z0-9'\".,!-]+", ' ', line)         if
len(line) > 3:
            full_text += line+' '

fr.close()
```

```
    #CLNING D FILE N STORING     fw =
open('./txt/'+fname[:-4]+'_clean.txt', 'w')
         fw.write(full_text.replace('Fig.', 'Figure'))
fw.close()     print ("conversion from TXT to Clean
TXT done!\n")

#BRKING TO FIT d SCRN def
format_text(string):
    words=string.split()
output=''
buffer_string=''     for
w in words:
if(len(buffer_string)<50
):
buffer_string+=w+' '
else:
            output+=buffer_string+'\n'
buffer_string=w+' '
output+=buffer_string        return
output
 def clean_txt_to_clean_words(doc):
global scrabble_list     doc =
doc.replace(',', ' ')     propernouns
= doc.lower().split()
    propernouns_clean = [word for word in propernouns if (word
not   in   scrabble_list)]            propernouns_string   =   '
'.join(propernouns_clean)
    stop_free = " ".join([i for i in
propernouns_string.split() if i not in stop])
    normalized  =  "  ".join(lemma.lemmatize(word)  for  word  in
stop_free.split())     return normalized
```

```
  def get_topics_from_text(line):
doc_complete = line.split('.')
    doc_clean = [clean_txt_to_clean_words(doc).split() for doc
in doc_complete]# ignore if length of docs for topic analysis
is less than 3            doc_clean_empty = True     all_topics
= []     for doc in doc_clean:         if len(doc) > 0:
            doc_clean_empty = False     if
len(doc_clean) >=1 and doc_clean_empty == False:
dictionary = corpora.Dictionary(doc_clean)
        doc_term_matrix = [dictionary.doc2bow(doc) for doc in
doc_clean]
        Lda = gensim.models.ldamodel.LdaModel
num_topics = 3
        ldamodel = Lda(doc_term_matrix, num_topics=num_topics,
id2word = dictionary, passes=25)
                for i in
range(0,num_topics):
            topic = ldamodel.get_topic_terms(i, topn=2)
topic_list = []             for word in topic:
                word_name = dictionary.get(word[0])
if len(word_name) > 1:
                    topic_list.append(word_name)
topic_list.sort()             topic_name = "
".join(topic_list)             add = False
for ch in topic_name:                 #NMBRS R IGNORED
if ch in r"[abcdefghijklmnopqrstuvwxyz]":
                    add = True             if
add:                 if topic_name not in
all_topics:
                    all_topics.append(str(topic_name))
     return
all_topics
```

```
 def get_topics_from_text1(line):
doc_complete = line.split('.')
    doc_clean = [clean_txt_to_clean_words(doc).split() for doc
in doc_complete]
    doc_clean_empty = True
doc_total_list = []
all_topics = []     for
doc in doc_clean:
if len(doc) > 0:
            doc_clean_empty = False     if
len(doc_clean) >=1 and doc_clean_empty == False:
for doc in doc_clean:
            doc_total_list = doc_total_list + doc
     print (" important word list: ",
doc_total_list)     for i in
range(0,len(doc_total_list),2):         if
i+2<len(doc_total_list):
            if (str(doc_total_list[i])  ==
str(doc_total_list[i+1])) and (str(doc_total_list[i+2]) ==
str(doc_total_list[i+1])) :                 topic_name =
(doc_total_list[i+2])
            elif    str(doc_total_list[i])                  ==
str(doc_total_list[i+1]):
                topic_name                  =               ('
'.join([doc_total_list[i],doc_total_list[i+2]]))
            elif    str(doc_total_list[i+1])                ==
str(doc_total_list[i+2]):
                topic_name                  =               ('
'.join([doc_total_list[i],doc_total_list[i+1]]))
            elif    str(doc_total_list[i])                  ==
str(doc_total_list[i+2]):
                topic_name                  =               ('
'.join([doc_total_list[i],doc_total_list[i+1]]))
else:
                topic_name                  =               ('
```

```
'.join([doc_total_list[i],doc_total_list[i+1],doc_total_list[i
+2]]))             add
= False             for
ch in topic_name:#
ignore numerical topics
if ch in
r"[abcdefghijklmnopqrst
uvwxyz]":
                    add = True             if
add:                 if topic_name not in
all_topics:
                    all_topics.append(str(topic_name))
         elif
i+1<len(doc_total_list):
            if      str(doc_total_list[i])                    ==
str(doc_total_list[i+1]):
                topic_name = (doc_total_list[i])
else:
                topic_name                  =                 ('
'.join([doc_total_list[i],doc_total_list[i+1]]))
            add = False             #NMBRS R ELIMINATED
for ch in topic_name:                 if ch in
r"[abcdefghijklmnopqrstuvwxyz]":
                    add = True             if
add:                 if topic_name not in
all_topics:
                    all_topics.append(str(topic_name))
     return
all_topics
  path = input("PATH TO PDF FILE:
") infile = path.split('/')[-1]
 full_text_messy      =      pdf_to_txt(infile)      fw      =
io.open('./txt/'+infile[:-4]+'.txt',      'w',encoding='utf-8')
```

```
fw.write(str(full_text_messy.encode('ascii','ignore')))
#print(full_text_messy.encode('ascii','ignore'))     fw.close()
txt_to_clean(infile)
 audio_dir = './audio/tmp'
picture_dir = './picture/tmp'
video_dir = './video/tmp' if
os.path.exists(audio_dir):
shutil.rmtree(audio_dir) if
os.path.exists(picture_dir):
shutil.rmtree(picture_dir) if
os.path.exists(video_dir):
shutil.rmtree(video_dir)
  fr = open('./txt/'+infile[:-
4]+'_clean.txt') count_lines = 1 for line in
fr:
    line = line.replace('\n','')
all_topics = get_topics_from_text1(line)
print ('\n\n',line,'\n')     print ("all
topics ", all_topics, '\n\n')
folder_names = []     for i in
range(0,len(all_topics)):         if
len(all_topics) > 4:
            image_downloader.download_images(all_topics[i],1)
else:
            image_downloader.download_images(all_topics[i],2)
folder_names.append(all_topics[i].replace(' ','_'))
text_sentences=[f for f in line.split('.') if len(f)>1]     if
len(text_sentences) <=0:
        continue
     if not
os.path.exists(audio_dir):
```

```
        os.mkdir(audio_dir)     if not
os.path.exists(picture_dir):
        os.mkdir(picture_dir)     if
not os.path.exists(video_dir):
        os.mkdir(video_dir)


    print ("creating "+str(len(text_sentences))+" audio files
")     for i in
range(0,len(text_sentences)):
        tts    =    gTTS(text=text_sentences[i],  lang='en',
slow=False)         tts.save(audio_dir+'/'+str(i)+'.mp3')
print ('\n',text_sentences[i],'\n')         print ("created "+
str(i)+ " audio file")
     text_clip_list=[]
audio_clip_list=[]
    silence                                                    =
AudioFileClip('./audio/silence.mp3').subclip(0,0.1)
audio_clip_list.append(silence)     for i in
range(0,len(text_sentences)):

sent_audio_clip=AudioFileClip(audio_dir+'/'+str(i)+'.mp3')
        print     ("length     of     audio:     "+str(i)+"      =
",sent_audio_clip.duration)
audio_clip_list.append(sent_audio_clip)
        sent_txt_clip                                          =
TextClip(format_text(text_sentences[i]),font='CourierBold',fon
tsize=200,color='yellow',bg_color='black',stroke_widt
h=30).set_pos('bottom').set_duration(sent_audio_clip.duration)
.resize(width=1000)
text_clip_list.append(sent_txt_clip)


audio_clip=concatenate_audioclips(audio_clip_list)
```

```
        file_names = []     for i in
range(0,len(folder_names)):
        files       =       (fn       for       fn       in
os.listdir(picture_dir+'/'+folder_names[i])                 if
fn.endswith('.jpg')      or      fn.endswith('.png')      or
fn.endswith('.PNG')      or      fn.endswith('.JPG')      or
fn.endswith('.jpeg') or fn.endswith('.JPEG'))
        for file in files:
            file_names.append(folder_names[i]+'/'+file)
    #   s_file_names   =   sorted(file_names,   key=lambda   x:
x.split('.')[0].split('/')[1])       s_file_names = file_names
number_of_images=len(s_file_names)     print (s_file_names)



    # minimum_image_size=1200
video_clip_list=[]

black_clip=ImageClip('./picture/black1.jpg').set_duration(0.1)
.set_fps(_FPS)
video_clip_list.append(black_clip)
black = './picture/black1.jpg'
title_clip_list = []     if
number_of_images > 0:         for f in
s_file_names:
temp_clip=ImageClip(picture_dir+'/'+f)
.set_duration(audio_clip
.duration/number_of_images).set_positi
on('center').set_fps(_FP
S).crossfadein(0.5)

            name_txt_clip       =         TextClip(format_text('
'.join([word[:1].upper()+word[1:]       for       word       in
f.split('/')[0].split('_')])),font='Courier-
Bold',fontsize=200,color='yellow',bg_color='black',stroke_widt
h=30).set_position('top').set_duration(audio_clip.duration/num
```

```
ber_of_images).resize(height=30)
title_clip_list.append(name_txt_clip)

            #                    temp_clip                          =
CompositeVideoClip([temp1_clip,name_txt_clip])

video_clip_list.append(temp_clip)

            #

minimum_image_size=min([minimum_image_size,temp_clip.size[0]])

print ('temp_clip width: ',temp_clip.size)     else:


temp_clip=ImageClip(black).set_duration(audio_clip.duration).s
et_fps(_FPS)          video_clip_list.append(temp_clip)

         #
minimum_image_size=min([minimum_image_size,temp_clip.size[0]])


    video_clip                                                    =
concatenate_videoclips(video_clip_list).set_position('center')

print("Video size:",video_clip.size)


    # print "Minimnum image size: "+str(minimum_image_size)

    # minimum_image_size = 0.98*minimum_image_size


    # resize_text_clip_list = []

    # for sent_txt_clip in text_clip_list:

    #
resize_text_clip_list.append(sent_txt_clip.resize(width=minimu
m_image_size))
txt_clip=concatenate_videoclips(text_clip_list).set_position('
bottom')     if len(title_clip_list) > 0:

         title_clip                                               =
concatenate_videoclips(title_clip_list).set_position('top')


result=CompositeVideoClip([video_clip,txt_clip,title_clip])

else:

         result=CompositeVideoClip([video_clip,txt_clip])
```

```
     print ("Composite video clip size:
",result.size)


result_with_audio=result.set_audio(audio_clip)


    # print txt_clip.duration
    # print txt_clip.fps     # print
audio_clip.duration     print ("audio duration:
"+str(audio_clip.duration))     print ("result
duration: "+str(result.duration))
    print           ("result           audio             duration:
"+str(result_with_audio.duration))
    # print video_clip.fps
    # print len(video_clip_list)

result_with_audio.write_videofile(video_dir+'/'+str(count_line
s)+'.mp4',codec ='libx264',fps=_FPS)
count_lines += 1
     shutil.rmtree(audio_dir)
shutil.rmtree(picture_dir)
fr.close()


video_files = [fn for fn in os.listdir(video_dir) if
fn.endswith('.mp4')]
video_files    =    sorted(video_files,      key=lambda     x:
int(x.split('.')[0])) video_clip_list = [] for video in
video_files:
    clip = VideoFileClip(video_dir+'/'+video).crossfadein(0.5)
video_clip_list.append(clip)
 video_clip=concatenate_videoclips(video_clip_list)
video_clip.write_videofile(infile[:4]+'.mp4',codec='libx264',f
ps=_FPS)
```

8. RESULTS AND DISCUSSION

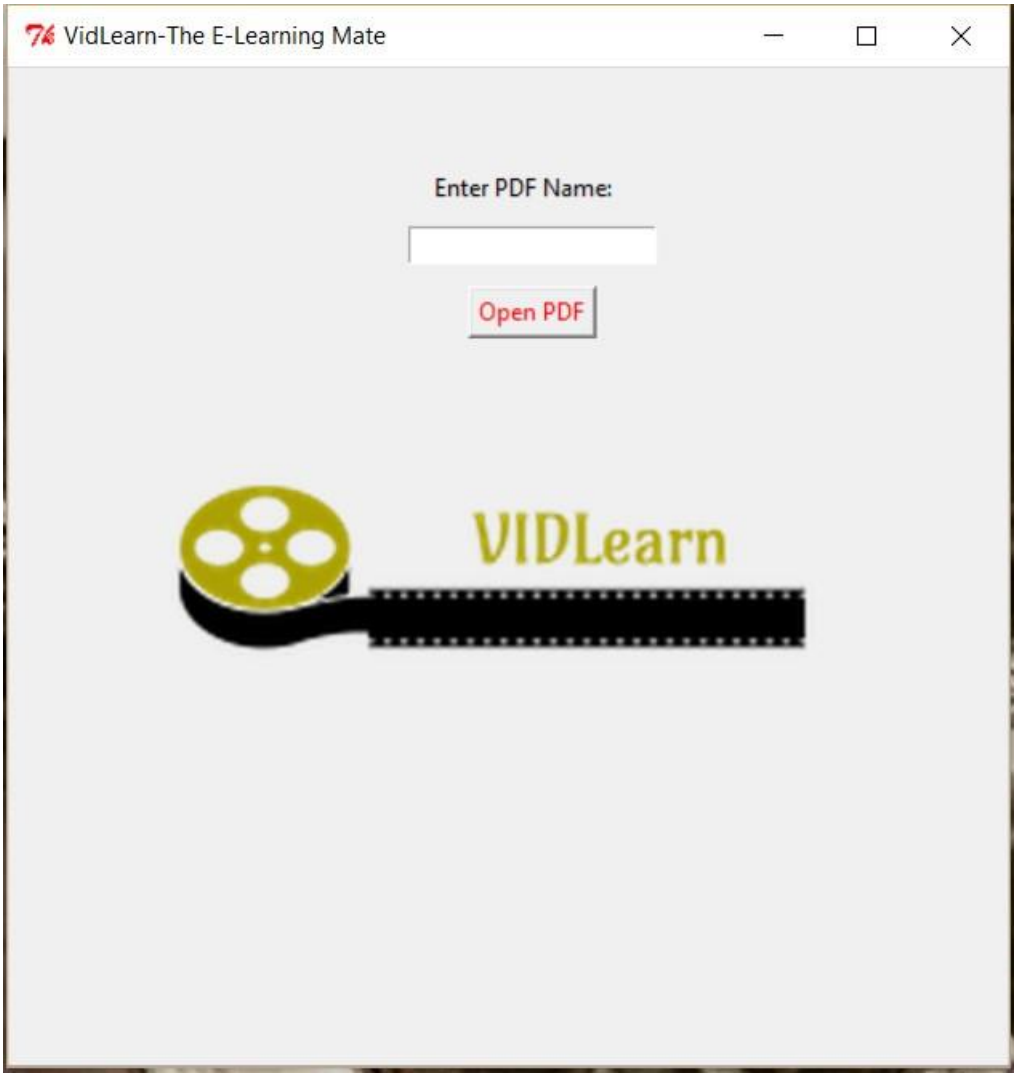


Figure 1 : GUI

Name	Date modified	Type	Size
abs.txt	3/19/2018 11:49 A...	Text Document	2 KB
abs_clean.txt	3/19/2018 11:49 A...	Text Document	2 KB
crow.1.txt	3/19/2018 5:33 PM	Text Document	2 KB
crow.1_clean.txt	3/19/2018 5:33 PM	Text Document	2 KB
crow.txt	3/19/2018 1:09 AM	Text Document	5 KB
crow_clean.txt	3/19/2018 1:09 AM	Text Document	5 KB
mak.1.txt	3/19/2018 5:05 PM	Text Document	2 KB
mak.1_clean.txt	3/19/2018 5:05 PM	Text Document	2 KB
pdf-sample.txt	3/19/2018 5:10 PM	Text Document	2 KB
pdf-sample_clean.txt	3/19/2018 5:10 PM	Text Document	2 KB
rani.txt	4/6/2018 2:42 PM	Text Document	4 KB
rani_clean.txt	4/6/2018 2:42 PM	Text Document	4 KB
rani1.txt	3/19/2018 6:09 PM	Text Document	4 KB
rani1_clean.txt	3/19/2018 6:09 PM	Text Document	4 KB
sample.txt	3/19/2018 4:54 PM	Text Document	8 KB
sample_clean.txt	3/19/2018 4:54 PM	Text Document	4 KB
shiv.txt	3/19/2018 12:47 A...	Text Document	1 KB
shiv_clean.txt	3/19/2018 12:47 A...	Text Document	1 KB
test.txt	3/19/2018 11:24 A...	Text Document	57 KB
test_clean.txt	3/19/2018 11:24 A...	Text Document	51 KB

Figure 2: Files created

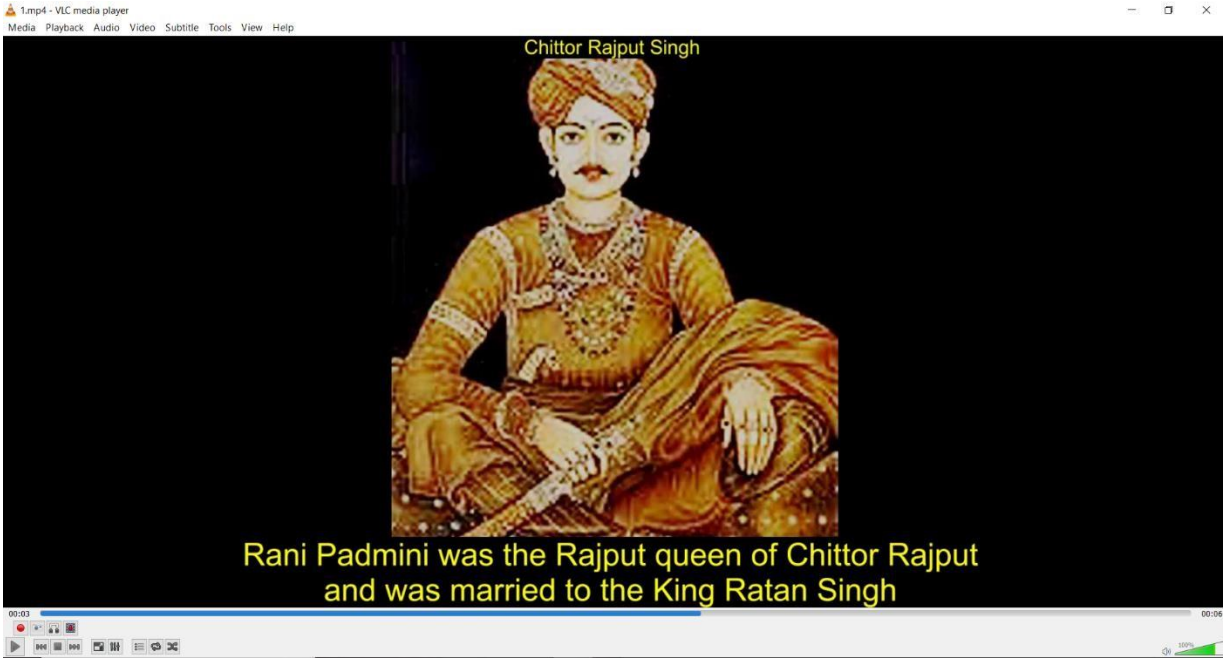


Figure 3: Video Created

9. CONCLUSION

This software was mainly created for the students with lack of interest in theory subjects. The results were as expected, scraping an image from google and adding the audio file for it while the video is played with the subtitles. This software also helps in remembering the chapters as a visual instead of only imagination. This helps in students easily understand and visualize the topics.