

index

December 15, 2015

1 CS 194-26: Video Supercuts

1.1 Welcome to our Interactive IPython Notebook

1.1.1 Find Phrases Demo

Our main program dynamically creates a supercut from a video and subtitle pairing. Users are able to view the occurrences of key clips from a large video file by extracting key words and phrases. As a bells & whistle, we made a feature that allows users to force the speaker to say a fake speech using his own words!

In this mini-demo, we will be showing off our find phrases feature on the video <https://youtu.be/TMDV3VY0tPA>, where Obama Delivers a Statement on the Shooting in Oregon.

Steps:

Process Command Line Input (Not Shown)

```
<li>Load Video and Subtitles </li>
<li>Process Subtitles to Determine Video Bounds </li>
<li>Slice the Video</li>
<li>Save the Video and Admire Results</li>
</ol>
```

```
In [33]: from config import INPUT_FOLDER, OUTPUT_FOLDER, LOG_FILE, TIMESTAMP_SET
        # Get the configuration variables from where our videos are stored
        from utils import parseSRT, flatten, testUserInput, listVideoFiles
        # Add a few utilities for ease of use
        from moviepy.editor import VideoFileClip, AudioFileClip, concatenate, ipython_display
        # Bring in the MoviePy Package for video splicing
        from IPython.display import YouTubeVideo, HTML
        import re, os, sys
        import argparse, datetime
```

After loading our utilities, we will now begin to pass in our video and subtitle data.

```
In [2]: VERBOSE = False
        video = VideoFileClip(INPUT_FOLDER+'1.mp4')
        subs = parseSRT('1')
```

Using MoviePy we load in our video, and process our subtitles with our parseSRT method from our internal utilities. Normally, we would process command line input, but for this demo will be just using the phrase method.

For reference, here is the sample video in question:

```
In [3]: YouTubeVideo("TMDV3VY0tPA", start=0, autoplay=0, theme="light", color="red")
```

```
Out[3]: <IPython.lib.display.YouTubeVideo at 0x1071b8050>
```

We can use MoviePy to see the first 10 seconds of the video file to make sure we are looking at the right file.

```
In [4]: ipython_display(video.subclip(0, 10), width=300)
```

```
Out[4]: <moviepy.video.io.html_tools.HTML2 object>
```

With our video and subtitles loaded, we are now ready to search for our favorite words. But first, let's take a look at what we have in our data!

```
In [5]: print(subs)
print 'Total Number of words: %d | Uniques: %d' % (subs.wordCount, subs.uniqueWordCount() )
fd = subs.freqDist
fdout = ['%s: %s' % (k, v) for k, v in sorted(fd.items(), key = lambda x: x[1], reverse = True)
        if not ('~' in k or '--' in k)] # take the most common words for show
print 'Common Words:', fdout[:20] # let's take a look
```

```
<SubtitleObj: 1 | lines: 260 | words: 2127>
```

```
Total Number of words: 2127 | Uniques: 719
```

```
Common Words: ['to: 67', 'the: 67', 'and: 63', 'of: 50', 'that: 42', 'we: 35', 'our: 33', 'in: 30', 'a:
```

Here we can see that a little more information about which words are commonly said, and since we are like totally rooting for America, we will examine just these time ranges.

```
In [27]: words = ['America', 'terrorist']
segmentedWordList = subs.words
times = subs.times
occurrences = []
for idx, wordList in enumerate(segmentedWordList):
    for word in words:
        if word.upper() in map(str.upper, wordList):
            timeRanges = [times[idx]]
            occurrences += timeRanges
            break
if len(occurrences) == 0:
    print "\nThe word(s) " + str(words) + " were not in the video. No supercut could be made.\n"
```

With a quick for loop, we are ready to extract the time ranges from our video

```
In [28]: print 'Time ranges:', occurrences
```

```
Time ranges: [['00:00:49,482', '00:00:52,385'], ['00:01:05,397', '00:01:07,399'], ['00:01:45,972', '00:
```

```
In [29]: slices = concatenate([video.subclip(start, end) for (start, end) in occurrences])
```

With our slices prepared, we join our key words to create our output file name and write the video to disk.

```
In [30]: keywordString = '-' + '-'.join([w.upper() for w in words ])
pathname = OUTPUT_FOLDER + '1' + '-' + 'phrases' + keywordString
slices.write_videofile(pathname + ".mp4", fps=video.fps,
                       codec='libx264', audio_codec='aac',
                       temp_audiofile= 'output/temp-audio.m4a',
                       remove_temp=True, audio_bitrate="1000k", bitrate="4000k")
```

```
[MoviePy] >>>> Building video output/1-phrases-AMERICA-TERRORIST.mp4
[MoviePy] Writing audio in output/temp-audio.m4a
```

```
[MoviePy] Done.
[MoviePy] Writing video output/1-phrases-AMERICA-TERRORIST.mp4
```

```
[MoviePy] Done.
[MoviePy] >>>> Video ready: output/1-phrases-AMERICA-TERRORIST.mp4
```

1.2 Let's see what we got!

```
In [41]: def playvideo(fname, mimetype):
        """Load the video in the file 'fname', with given mimetype, and display as HTML5 video.
        """
        from IPython.display import HTML
        from base64 import b64encode
        with open(fname, "rb") as f:
            video_encoded= b64encode(f.read())
            video_tag= """
        <center><video controls style='max-width:100%'>
        <source src='data:{mimetype};base64,{b64}' type='video/{mimetype}' loop=1 autoplay>
        Your browser does not support the video tag.
        </video><center/>""".format(mimetype=mimetype, b64=video_encoded)
            return HTML(data=video_tag)
        name = pathname + '.mp4'
        playvideo(name, name.split('.')[1])

Out[41]: <IPython.core.display.HTML object>
```

1.3 Thanks for reading!

1.3.1 Authors: Zach Zeleznick & Ollie O'Donnell

```
In [ ]: import urllib2
        # HTML(urllib2.urlopen('http://bit.ly/1Bf5Hft').read())
        # uncomment above line for magic CSS styling
```