Video summarization using a machine learning approach

Harikrushna Vanpariya

Abstract— In this paper, we describe machine learning approach for summarizing video. It will describe steps to summarize video with audio and create summary video with meaningful sentences.

Index Terms— Machine Learning, Video summarization, Speech recognition, Speech summarization.

1 Introduction

More than 1.9 Billion logged-in users visit YouTube each month. Everyday people watch more than a billion hours [1]. More than 70% of the total video viewers are watching in mobile devices [1]. These statistics are only for YouTube there are other video channels also available. With increasing number of the mobile devices, the video watchers are increasing day by day. There are different kind of videos available and people face difficulties to identify whether the video contains useful content or not. With the help of Machine Learning, we can summarize a video in short video and by watching that summary video, we can decide to watch complete video or not. This will also help in creating Ad of video tutorial or channel.

2 Machine Learning Tools

In video summarization, various tools like, Sphinx, MEAD and ffmpeg are used. Each of these tools will be explained in more detail in this section.

2.1 Sphinx-4 - Speech recognition

Sphinx-4 is built on Java, due to which it supports high portability. In Java, once the code is compiled it can be executed on any platform, which gives Sphinx-4 an advantage of executing on different platforms. Sphinx-4 supports different kinds of feature streams, language models , grammars and types of acoustic models. The decoder of Sphinx consists of main three modules: search manager, linguist, and acoustic scorer.

1. Search Manager: The search manager creates a tree of possibilities based on best hypothesis. The search manager required acostic scores which can be fetched from acostic scorer [2]. A token tree will be used by search manager. The token will also be used in other speech recognition systems [3].

   The search algorithm has an active list. The active list will hold a set of active tokens. Each token is assigned scores based on acoustic and language scores by SentenceHMM. During search, based on the score low scoring branches are pruned. Active list will be updated with the remaining tokens post the pruning. Depth-first search and breadth-first search are the two ways the search will be performed in sentenceHMM. Viterbi algorithm and Bush-derby algorithms are used to perform breadth-first search in Sphinx-4. Directed acyclic graph, which has a source and a sink, is used to represent each competing unit during search.

   In Viterbi algorithm, based on the probability of the best path each competing unit is assigned a score. The competing unit with maximum best score wins [2]. For example. If probabilities on ABC the egdes are (0.8,0.03,0.02) and for DEF the edges are (0.2,0.5,0.4) then the score for ABC would be 0.8 and score for DEF would be 0.5 and ABC will win. If the probability of entire path would be considered, then DEF will win [2].

   In Bush-derby algorithm, an η-score will be calculated for each directed acyclic graph. In compare to Viterbi algorithm, here all the paths will contribute to score. At a max node the sink nodes of competing unites will be united. Out of all, the one with the highest score will wins.

   To calculate the η-scores, the source node assined with score one and the score of other nodes will be calculated recursively from the predecessors and the probabilities of the edges in the directed acyclic graph [2].

2. Linguist: The linguist converts the linguistic constraints to the grammer. The grammer is the internal data structure and used by search manager. Each node of the grammar is the set of words which are spoken at perticular time. The directed graph of these type of nodes creates grammar. A SentenceHMM, in which the grammar will further be compiled. SentenceHMM is a directed state graph.In SentenceHMM, each state represents a unit of speech [2].

3. Acoustic Scorer: All information related to the state output densities will be retained by the acoustic scorer.The duty of the acoustic scorer is to compute state output probability or density values for the different states, for any given input vector. The serach modul get the score from acoustic scorer as needed. Acoustic scorer generates the score based on semi-continuous , discrete HMMs or continuous. Any heuristic algorithms incorporated into the scoring procedure for

IJSER

speeding it up can be executed locally within the search module [2].

2.2 MEAD

MEAD is a summarization and evaluation toolkit. It supports multi-lingual. Position-based, Centroid, TF*IDF, and query-based methods are the various summarization algorithms supported by MEAD. University of Michigan developed MEAD. Johns Hopkins University held eight-week summer workshop on Text Summarization in which MEAD v3.01 – v3.06 were developed [4].

The document will be processed sentence by sentence with basic method of MEAD. MEAD will calculate importance weight of each sentence, sort it and prepare the summary with the sentences with the highest importance weight. Before adding new sentence to the summary, it will also check if there is overlap with existing sentences in summary. If there is no overlap, then it will add new sentence to the summary.

To generate importance score information which is considered is the sentence length, position of sentence in the document and the word frequencies [4].

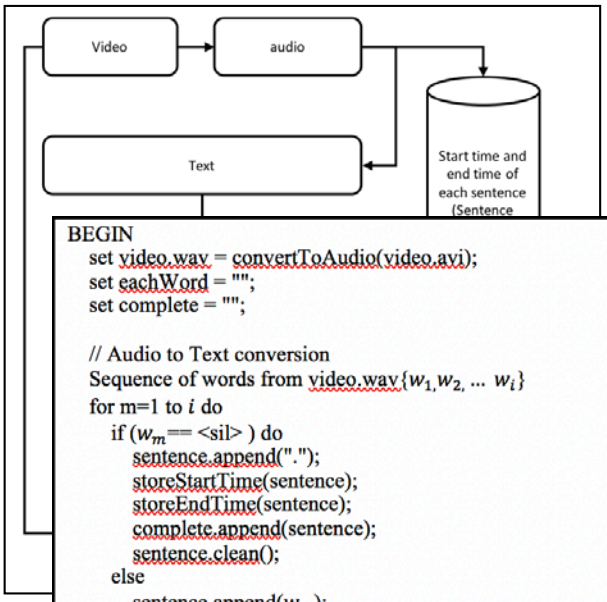
2.3 FFMPEG

To process video and audio files ffmpeg is very quick. It can also process live video/audio source. Resizing and transforming video is also supported by ffmpeg [5].

ffmpeg has one of the features which joins audio/video streams, together one after the other. The filter works on sections of synchronized video and audio streams [5].
ffmpeg also has feature to split the file from the given position to the audio/video file. It will search for the closest seek point from the position and split the video/audio from that position [5].

3 SUMMARIZE VIDEO

In this section, all the steps involved in summarizing video with Machine Learning is explained. Fig. 1. Shows the complete flow of video summarization. Fig. 2. depicts the complete video summarization process. The completed problem is divided into following major steps: Video to Text conversion, generate text summary and Summary to build Video. Video to text conversion



BEGIN
set video.wav = convertToAudio(video.avi);
set eachWord = "";
set complete = "";

// Audio to Text conversion
Sequence of words from video.wav $\{w_1, w_2, \ldots w_i\}$
for m=1 to i do
if (w_m == <sil>) do
sentence.append(".");
storeStartTime(sentence);
storeEndTime(sentence);
complete.append(sentence);
sentence.clean();
else
sentence.append(w_m);
done
done

// Predict summary
set summarySenteces[] = summary sentences $\{s_1, s_2 \ldots s_j\}$
set video[];
set n=0;
for m=1 to j do
set $time_{start}$ = getStartTime(s_m);
set $time_{end}$ = getEndTime(s_m);
video $_{n++}$ = splitVideo($time_{start}$, $time_{end}$);
done
$completeVideoClip = \sum_{k=0}^{n} video_k$
END.

Fig. 2. Steps to follow for Video Summary with ML

1. Video to Text conversion: As a first steps of this solution the video content is converted into an audio file. To convert video to audio file, ffmpeg will be used. Once ffmpeg will convert the file into audio, SPHINX – speech recognition is used to identify the content and convert into the text.
   An additional step is performed when SPHINX will convert speech to text, is to store the start time and end time of each sentence and store it. We call that storage as "sentence timing storage". So, with the help of ffmpeg and SPHINX, video file will be converted into the text content.

1. Generate text Summary: In this step, the summary will be generated from the text. To generate the summary,

MEAD is used to derive the summary from the complete text of the video.

2. Summary to build video: The summary text would contain multiple sentences. For each summary sentence, start time and end time will be fetched from "sentence timing storage" (which has stored the sentence timing in "Video to Text conversion" phase).
   Once all start time and end time of the video will be fetched from the "sentence timing storage", as per summary sentence, the video will be sliced. All the sliced frame of the video from the summary sentence will be re-joined and final summary video will be created.

   Table 1. depicts the results of various summary video performed. The experiment is performed on different tutorial videos . It summaries the video in 3 or 4 sentence depends on the configuration in text summarization phase.

TABLE 1

Results of different Summary Videos

Video Length	Sentences	Sentences in Summary	Summary Video Length
4 minutes	82	3	8 seconds
9 minutes	179	3	7 seconds
30 minutes	610	4	11 seconds

IJSER

4 Conclusion

With the help of Machine Learning the video with the voice can be summarize and it will help viewer to identify if the video is interested to watch. Here with the help of Sphinx, MEAD and ffmpeg, which uses different algorithms to perform functionality, video summarization is achieved.

Video summarization can be enhanced to support different languages, as Sphinx provides support to other languages as well.

References

[1] YouTube for Press. Retrieved 31st January 2019, from https://www.youtube.com/intl/en-GB/yt/about/press/

[2] The CMU Sphinx-4 Speech Recognition System, Lamere et al. - http://www.cs.cmu.edu/~rsingh/homepage/papers/icassp03-sphinx4_2.pdf

[3] S.J. Young, N.H.Russel, and J.H.S. Russel (1989). "Token passing: A simple conceptual model for connected speech recognition systems," Technical Report, Cambridge University Engineering Dept.

[4] Dragomir Radev, Timothy Allison, Sasha Blair-goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Adam Winkel, Zhu Zhang MEAD - a platform for multidocument multilingual text summarization, In: Proceedings of LREC, Lisbon, Portugal, 2004

[5] F. Bellard and M. Niedermayer. The FFmpeg Project, June 2008. http://ffmpeg.org