Video Summarization using NLP

Shivesh Meenakshi Murali
University of Southern California
smeenaks@usc.edu

Sukavanan Nanjundan
University of Southern California
snanjund@usc.edu

Harshitha Kurra
University of Southern California
kurra@usc.edu

Monali Rajendra Patil
University of Southern California
monalira@usc.edu

Ashutosh Gupta
University of Southern California
guptaash@usc.edu

Abstract

In the era of unlimited access to educational content and videos on the internet, it has become ever so important to present these content in a precise and concise manner. In this paper we present an approach to summarize videos by means of techniques used for extractive text summarization problems.

1 Introduction

With the rapid proliferation of electronic gadgets, there's a remarkable growth in the amount of textual and non-textual data generated by every individual. Further, with the sudden surge in the availability of multiple video-on-demand platforms like Netflix, Amazon Prime Video and YouTube has led to an exponential increase in the amount of visual content generated.

Video-sharing platforms like YouTube have advanced recommendation systems powered by the behavior of users and their dwell time on videos. However, this also means that users are becoming selective of what they see and prefer to know the substance of the content beforehand. This, paired with the fact that users are spending increasingly less time watching videos, it becomes important to convey the content quickly with as little filler content as possible. This means that there's a marked rise in approaches to "summarize" content, aiming to reduce the length of the video besides retaining the informativeness and perception quality of the video. Since lecture videos tend to run the duration of a class, anywhere from 45 minutes to 90 plus minutes, this problem is exacerbated. Summarizing a long sequenced educational video has many applications in scenarios where time is a critical resource. As summarized video is expected to contain the essence of the whole video, it is highly beneficial for academicians on educational platforms to revisit a topic. Further, summarizing a video ensures that the cognitive load on the minds of learners is minimized and can work with the attention span of specific learners.

Work done by (Brame, 2016) describes multiple types of mental effort necessary to assimilate video-form educational content. This paper discusses the additional unwanted cognitive impact called Extraneous load which makes video understanding harder due to malformed lessons, or poorly structured content. Another paper by (Guo et al., 2014) which analyzed over 6.9 million video sessions over four MOOC courses on eDX to map student engagement with length of the videos. Their findings also point to the fact that shorter videos are more engaging with normalized engagement time close to 100% till the 6 minutes to 9 minute range and progressively falling off thereafter. Thus, video summarization serves a very important purpose for educational video comprehension.

2 Experiments

As we know, there are Abstractive and Extractive Text Summarization. Extractive Text Summarization contains words from the original text only and abstractive summarisation may or may not contain the words from the original text while summarising.

We have experimented with Summarizations using:

1. **TF-IDF**: The initial approach utilized involved a module to rank subtitles based on a modified TF-IDF approach. By treating each subtitle entry as a document and the whole subtitle file as a corpus, we can rank each subtitle entry based on this score and use this to create a summary. This method also introduces a concept of flexibility parameter, discussed in Section 3.1.

2. **BERT Embeddings - Supervised approach**: In this method, we initially tokenize the data and generate the BERT embeddings of the subtitle sentences and pass these embeddings through a bi-LSTM layer and a fully-connected layer before passing it through a classifier layer. Scoring the sentence based on the number of important words in the sentence allows us to rank the subtitles and by extension, allows us to generate the summary of the video.

3. **BERT Embeddings - Unsupervised approach**: As discussed in the paper by (Miller, 2019), text summarization could be done in an unsupervised manner by getting either the sentence level or word embeddings and then performing K-Means on the embeddings and clustering them together. Now, the number of cluster could either be determined by the user or by an automated method such as the Silhouette method (Rousseeuw, 1987). Due to computational constraints we had to reduce the sequence length of the model and also the dimensions of the embeddings. This drastically reduced the unsupervised model's performance and made the results not worth mentioning here.

3 Implementation

Before we begin, the key tasks involve obtaining the video and the subtitle files needed to commence learning. Since the code uses a modular approach to summarization. The first module allows for downloading YouTube videos using the youtube-dl command line module. Next, subtitle files are collected and parsed into a common file format (Figure 2). This allows for easy extension into any subtitle format. Support for `.srt` and `.vtt` subtitle files are included with this module by implementing custom parsers. This platform agnostic format allows for different modules to be used for learning from text. One older implementation as a part of a bachelor's capstone project used a simple system based on sentence level tf-idf sums. This project uses a system based on NLP techniques.

We are modeling extractive text-summarization as a sequence-to-sequence prediction task by assigning 1 or 0 labels to embeddings of each word in the given sequence. 1 meaning it is important and present in the summary and 0 meaning that it is not selected as part of the summary. We use the DebateSum dataset by (Roush and Balaji, 2020) for training. We used Title based Video summarisation dataset for this model. TvSum dataset is an unsupervised video summarization framework that uses title-based image search results to find visually important shots. TVSum50 contains 50 videos and their shot- level importance scores annotated via crowdsourcing. The main genre of videos that we target is lecture videos, the structure of which closely resembles that of a formal structured conversation or explanation. To that end, the debate sum dataset is the only dataset that closely resembles that structure which has supervised examples for extractive text summarization.

By using BERT embeddings of the subtitle sentences. We generate the BERT embeddings of the subtitle sentences by first tokenizing them, then finding the embeddings for the tokens using the BERT base uncased model. We then pass the BERT embeddings through a bi-directional LSTM layer and a fully-connected layer before passing it through a classifier layer (2-nodes). We choose 50481 texts that are of 350 words or less in order to find the tokenized version of the text within input sequence length of the BERT model, which is 512 tokens. We then choose the final hidden state from the model as the embeddings for the tokens. Then we un-tokenize the tokens and convert them to the original words, which admittedly isn't a lossless transformation, in order to merge the embeddings of different tokens of the same word. For example, the word "embeddings" is tokenized as "em", "##bed", "##ding", "s". The final output is of the shape `Batch_size` $\times$ `Seq_Length` $\times$ 768(`hidden_dim`).

Due to computational constraints, we reduce the embedding dimension from 768 to 100 using PCA. Although, other word embedding vectors such as word2vec, GloVe etc., can be reduced to a lower dimension using PCA and other techniques, such as the ones presented by (Raunak et al., 2019) there isn't yet an efficient method to reduce the dimensionality of contextualized word embeddings such as BERT embeddings. PCA does produce good results albeit the loss during the reduction. When a list of sentences / subtitles is given, we first divide them into subdocuments that do not exceed 251 words and then find the important words for each sub-document. Then we assign an importance factor to each of the sentences in the sub-

```
bLSTM(
  (blstm): LSTM(100, 256, batch_first=True, dropout=0.33, bidirectional=True)
  (linear): Linear(in_features=512, out_features=128, bias=True)
  (dropout): Dropout(p=0.33, inplace=False)
  (classifier): Linear(in_features=128, out_features=2, bias=True)
)
```

Figure 1: Model Overview

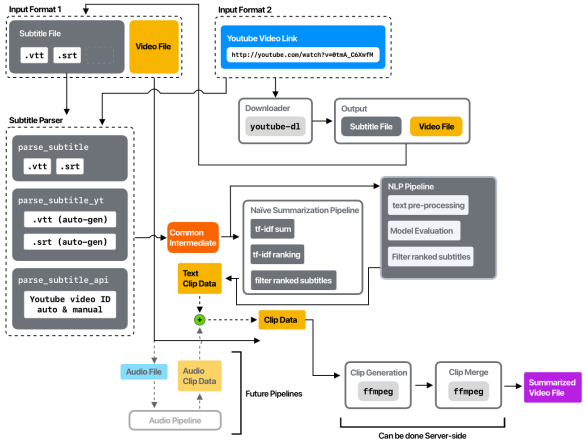

Figure 2: NLP Model Architecture

Categories	Documents
Train	40384
Dev	5048
Test	5049

Table 1: Document train-test Split

document based on the number of words that have been selected for the summary from that particular sentence.

3.1 Flexibility Parameter

Now the concept of a flexibility parameter is introduced. This parameter helps to introduce a ”temporal importance” to selected subtitles. Subtitles that occur close to selected subtitles are also deemed important. That is, if the subtitle indices, 1 and 3 are deemed important and if the flexibility parameter is 1, then at most one unselected subtitle index will also be flexibly selected in the summary under the assumption that spoken words in-between any selected subtitles might be important. For the purposes of this project and analysis, the flexibility parameter, F, has been set to 3.

The overall project architecture therefore looks has the following structure:

Finally, this NLP module returns another platform agnostic file called the clip list file. This file

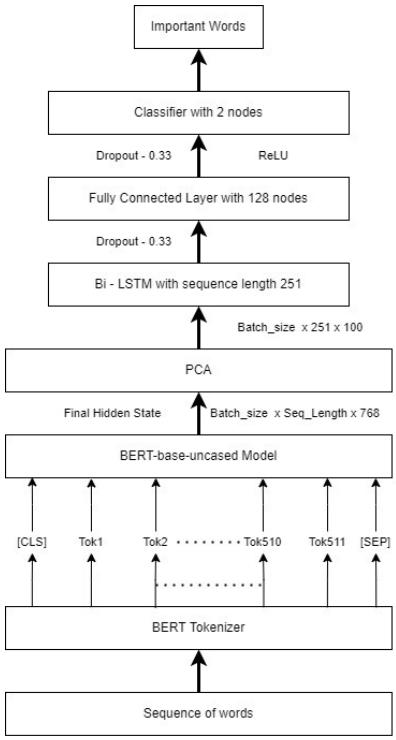

Figure 3: NLP Model Architecture

Parameter	Values
Batch size	256
Optimizer	AdamW
Epoch	10
Learning Rate	0.001
Weight Decay	0.01

Table 2: Final optimization parameters for the project

Hardware	Power
CPU	AMD Ryzen 9 5900HS
GPU	Nvidia GeForce RTX 3070
RAM	8GB

Table 3: Computational resources used for the project

contains the indices of the clips in the summary and the start and the end times of the clips. Another module loads in the clip file and appropriately cuts the input video mindful of the frame rate of the video.

4 Results and Discussions

As we can see from Table 4, the BERT-LSTM model outperforms the state of the art Longformer-Base model on all ROUGE metric on the debateSum dataset. While the fact that the Longformer-Base model is trained on a larger set of data of max sequence length 4096 makes it harder to compare our model, which on the other hand has a max sequence length of 251 and is trained on a considerably lesser amount of data, there is no denying the fact that the results of our model do look promising. Training on just 40k+ data while undergoing some lossy transformations such as untokenizing the texts and dimensionality reduction through PCA, our model is still able to achieve a competitive score. This proves that the proposed model architecture of using LSTM on top of BERT embeddings for extractive text summarization is indeed an improvement to the current state-of-the-art model.

Now, moving on to the video-summarization results, we can clearly see that the BERT-LSTM model is able to capture more important parts of the video than the TF-IDF model was able to, even with the same flexibility parameter. This shows that the BERT-LSTM model has succeeded in establishing some sort of relation between the words in the video and their importance in the context of the whole video. On the other hand, the TF-IDF model, which is comparatively very simple, isn't able to capture these dependencies and thus produces very short and incoherent summaries, as evidenced by the width of the continuous frames captured. One unique observation to notice is that the BERT-LSTM model seems to always capture some initial part of the video. This may be due to the nature of the video and the content present in the early parts of the video. For example, in the 6 majority of the captured frames are from the initial parts of the video. This is because the said video is an informational video about how to handle wasp nests aimed at gardeners and home-owners. In such informational or instructional videos the early parts of the video mention what the video is about and what they are going to demonstrate. This in itself becomes a small summary of the video, which is important, and the BERT-LSTM model captures it effectively.

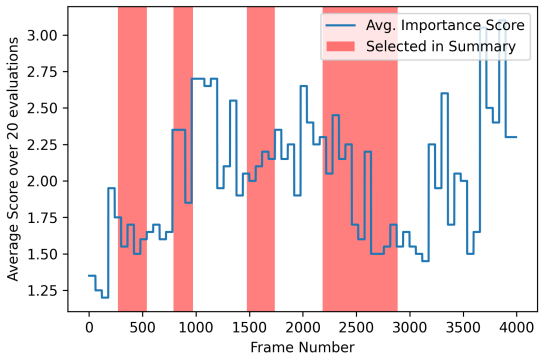

Figure 4: BERT-LSTM Result on Electric Cars Video - Informational

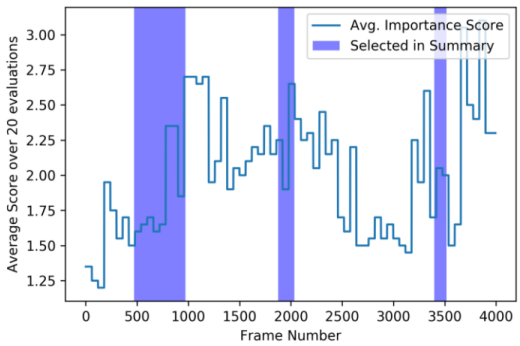

Figure 5: TF-IDF Result on Electric Cars Video - Informational

5 Conclusions

Even with training on just 40384 examples, the suggested BERT-LSTM model is able to produce results that are better than the SOTA model (Longformer-BASE). Not to mention, that the data undergoes some lossy transformations such as untokenization of the embeddings and dimensionality reduction of the embeddings. This shows that the model is both powerful and robust. Even

Model	ROUGE-1	ROUGE-2	ROUGE-L
BERT-Large (Devlin et al., 2018)	56.32	35.20	49.98
GPT2-Medium (Radford et al., 2019)	52.07	34.20	53.23
Longformer-Base-4096 (Beltagy et al., 2020)	60.21	38.53	57.21
Our Model	**60.57**	**47.83**	**59.17**

Table 4: Comparison of different token classification transformer models fine-tuned on the training split of the DebateSum dataset

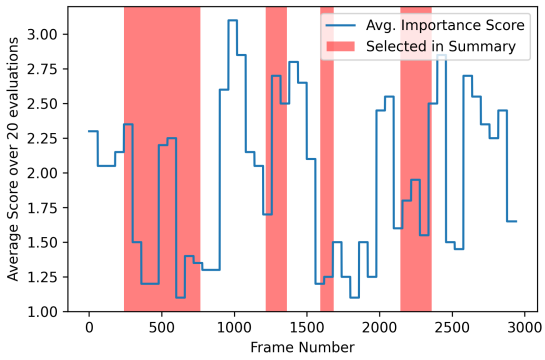

Figure 6: BERT-LSTM Result on Wasps Video - Educational

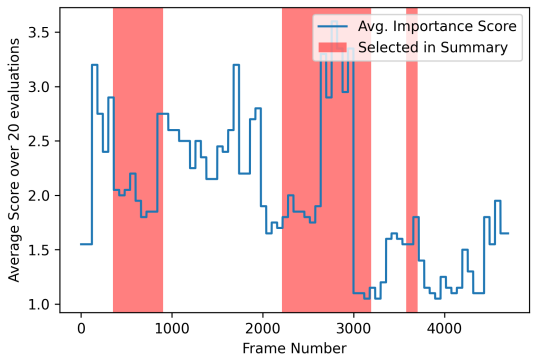

Figure 8: BERT-LSTM Result on Pet Spa Video - Informational

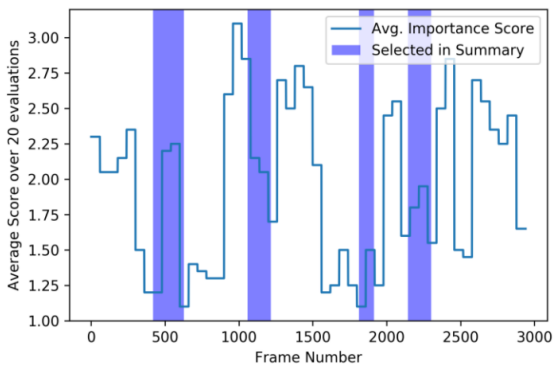

Figure 7: TF-IDF Result on Wasps Video - Educational

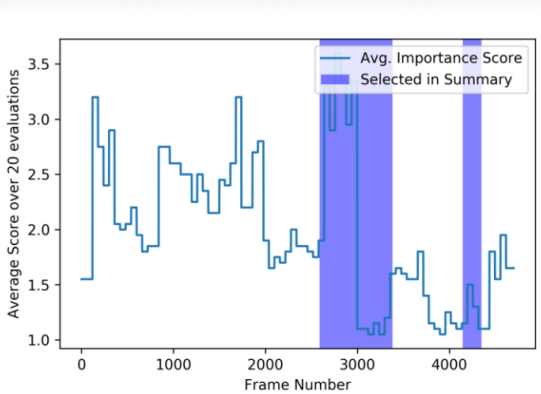

Figure 9: TF-IDF Result on Pet Spa Video - Informational

though a direct comparison between the SOTA model and the BERT-LSTM model cannot be drawn explicitly due to the differences in the training process, there is no denying that the BERT-LSTM model will produce better results if the same training process is followed between the two models, as evidenced by the ROUGE metric results. Adding to that, the BERT-LSTM model also outperforms the TF-IDF model by being able to capture longer and more coherent summaries of the videos.

6 Future Work

Future improvements to this model would be incorporation of an audio pipeline to handle auditory cues (not speech recognition). Some points of interest that we can glean by this methodology might include:

1. **Voice energy**: Energetic voices of the lecturer and/or students could form another datapoint that would let us evaluate for important segments of video.

2. **Tonal Changes**: Handling different voices and voice levels which could correspond to a student asking questions for instance. These form valuable portions of the video for inclusion into the summary.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Cynthia J Brame. 2016. Effective educational videos: Principles and guidelines for maximizing student learning from video content. *CBE—Life Sciences Education*, 15(4):es6.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Philip J Guo, Juho Kim, and Rob Rubin. 2014. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference*.

Derek Miller. 2019. Leveraging bert for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243.

Allen Roush and Arvind Balaji. 2020. Debatesum: A large-scale argument mining and summarization dataset. *arXiv preprint arXiv:2011.07251*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.