CSCI 544 Project: Text Summarization of Lecture Video Transcripts

Linsheng Ji, Sylvia Li, Haonan Xu, Dalya Alqaseer, Ronet Swaminathan
{jilinshe, lisiwei, haonanxu, alqaseer, rswamina}@usc.edu
Univeristy of Southern California

1 Introduction

The expansion of the internet has resulted in an abundance of multi-sourced information being created in the form of videos available for public consumption in recent years. This includes the big supply of online educational videos varying in duration, content, and presentation style. Lecture videos usually have a long duration (more than 30 minutes), making it hard to grasp the main ideas or to choose suitable videos to watch without summary of the videos and thus there is a necessity for developing such summarization techniques. The problem of creating a summary of lecture videos has been less studied. While the available textual summarization methods are trained and built for written texts such as news and blog articles, this mismatch of data sources may lead to issues while making use of these techniques to spoken language as in videos.

This project is intended to summarize educational videos by experimenting with existing methods of long-text summarization for video transcript summarization. We used texts scraped from the Open Yale Courses website (https://oyc.yale.edu/courses); the corpus contains more than 1000 long transcripts and corresponding summaries from lectures of 41 courses. Our implementation takes over from fine-tuning BART model and compares performance of abstractive summarization alone and that of a combined approach of both extractive and abstractive summarization. We have made our code[1] publicly available with demo video[2].

2 Related Works

Textual summarization is classified based on the algorithm used for abstractive and extractive summarization(Liu, 2019). The abstractive summarization summarizes a document in a human-like manner using external vocabulary and paraphrasing, producing words and phrases in the target summary which were not in the original text. In contrast, extractive summaries are formed by copying and concatenating the most important spans, i.e. sentences, phrases, or words considered to be significant.

In the field of long text summarization, transformer-based models represented by BERT have the advantages over CNNs or RNNs of consuming less time for training as well as capability to utilize pre-trained models on a large corpus, maintaining state-of-the-art performance in different domains of text materials.

2.1 Sentence-BERT (SBERT)

This is an application of pretrained BERT network that generates semantically meaningful sentences that can be compared using cosine similarity. Common methods to map sentences to a vector space such that semantically similar sentences are close include averaging the BERT output layer embeddings and using the output of the first token (the [CLS] token) from BERT. This common practice yields rather bad sentence embeddings, often worse than averaging GloVe embeddings. SBERT uses siamese and triplet networks (Schroff et al., 2015) to fine-tune BERT/RoBERTa, updating the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

2.2 K-Means Clustering of BERT Embeddings

There have been few attempts to summarize lecture transcripts in the field of text summarization, among them the only one study we have found addressing this void creates summaries of lectures by K-means clustering of sentence embeddings in-

[1] https://github.com/siwei-li/NLP_summarization

[2] https://www.youtube.com/watch?v=6N7IAukfnkc

ferred from BERT model (Miller, 2019). Multiple layers of averaged word embeddings were taken into consideration to produce the best representation of sentences. From the clusters, the sentences closest to the centroids were selected for the final summary. Some of the weaknesses found in this study include insufficiency with summarizing long lectures, as well as difficulty dealing with conversational language over written transcripts. For long lectures, classified as those that have 100 or more sentences, the challenge was to have a small ratio of sentences be properly representative of the entire lecture. That is why in our project we consider it an applicable idea to first select a relatively larger amount of highlight sentences to include more information, and then feed them into abstractive summarization models.

2.3 VT-SSum: A Benchmark Dataset for Video Transcript Segmentation and Summarization

VT-SSum is a benchmark dataset with spoken language for video transcript segmentation and summarization, which includes 125K transcript-summary pairs from 9,616 videos. VT-SSum takes advantage of the videos from `http://videolectures.net/` by leveraging the content of the slides as the weak supervision to generate the extractive summary for video transcripts (Lv et al., 2021). The alignment of knowledge domains in this dataset makes it ideal for training extractive models in our project.

3 Methods

In this study we investigate if combining the two approaches of extractive and abstractive summarization further elevates the performance, as shown in the previous works (Gehrmann et al., 2018; Pilault et al., 2020). We first trained an extractive model to condense the articles into 1/3 of the original lengths. Then we feed the condensed texts into an abstractive model to produce the final summary. Our extractive model is a SentenceBERT-based classifier trained on contextual encodings of sentence-document pairs, while the baseline abstractive model is a BART model, which has a maximum length limitation on input texts of 1024 tokens. We also compared the performance of our extractive model with that of the one produced by Miller (2019).

3.1 Data Sources

Attempts to directly apply NLP methods on text decoded by speech recognition usually fail because of the poor quality of the detected text, which is reflected in a high word error rate and lack of punctuation (Taskiran et al., 2006). Thus, we try to limit the scope to those videos with manually curated transcripts and summaries available. Due to scarcity of research work of such kind, we have found two sources of data in the format of video transcript. The first one is lecture video transcripts scraped from the Open Yale Courses website (`https://oyc.yale.edu/courses`); the corpus contains more than 1,000 video transcripts and their corresponding summaries from lectures of 41 courses, spanning over 30 minutes in duration. We only take videos that are long enough in duration to provide a document with rich information. This dataset was split into training and testing set by an 80/20 ratio.

Additionally, to train an extractive model on top of the baseline abstractive summarization model, we are using VT-SSum dataset which contains segmentation of sentences from lecture video transcripts and corresponding 0/1 labels as to whether or not a sentence within the transcript is included in the extractive summary of that video.

3.2 Experimental Setup

Experiments were carried out on Google Colaboratory and a machine with AMD Ryzen 7 3700x, 16 GB of RAM, and Nvidia 2080 super. Python 3.8.11, Pytorch 1.9.1, Transformers 4.12.5 were used for developing the models and approaches.

3.2.1 Extractive Model Training

We used the pretrained "sentence-transformers/paraphrase-MiniLM-L3-v2 model" from HuggingFace transformers library for implementation of Sentence-BERT model. To train this extractive model, we used 80% of the VT-SSum dataset containing 3000 documents with 213066 sentences in total for training versus 500 documents with 37143 sentences for testing. The structure of the model is to first get mean pooling embeddings for each input sentence (u) and document (v), then concatenate the sentence-document pairs (Concat(u,v, u*v)), and finally feed them into dense and dropout layers for classification. In VT-SSum dataset approximately one out of ten sentences is included in the summaries. Hence

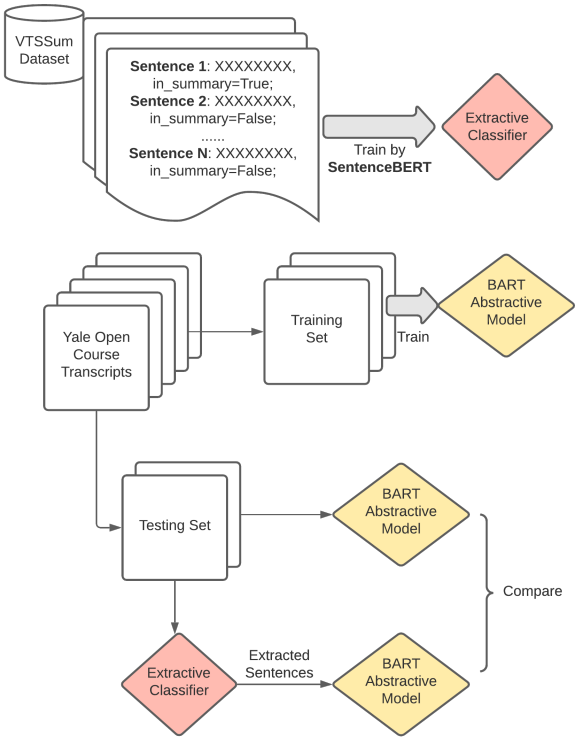

Figure 1: Workflow of the project.

we also did balancing of datasets by randomly removing sentences not appearing in the summaries such that they are approximately twice as many as those sentences included in summaries.

We used Adam optimizer with a learning rate of $1e^{-05}$. The hyper-parameters experimented in this project were as follows: dropout rate: 0.3, embedding dimension: (1152, 768), classifier dimension: (768, 1), batch size: 4, epochs: 20.

3.2.2 Abstractive Model Training

We used the $BartForConditionalGeneration$ class to define our BART model from pretrained parameters in HuggingFace's "facebook/bart-base" model. $BartForConditionalGeneration$ adds a Language Model head to our Bart model, which allows us to generate text based on the training of BART model. Note that due to the limitation of maximum input length in BART, we used the default truncation method taking only the first 1024 tokens of transcripts.

When fed into model training, we removed those videos with transcript or description less than 500 characters and 100 characters long, respectively. Given the distributions of text lengths, we set the maximum length of generated summaries to 250 tokens. We took 20% of transcripts for validation as well.

We used Adam optimizer with a learning rate of $1.5e^{-05}$. The batch sizes for training and validating were both 2. For training a total of 12 epochs were run.

3.3 Evaluation Metrics

3.3.1 ROUGE Scores

ROUGE scores were used in the automatic evaluation of our summaries because they are commonly used to compare the content overlapping of texts and have been shown to correlate with human evaluation of quality of summary.

There are two primary types of ROUGE scores that we are working with, namely ROUGE-Ns and ROUGE-Ls. ROUGE-N measures n-gram overlaps between generated and reference sentences. On the other hand, ROUGE-L measures the length of longest common sequence (LCS). The advantage of looking for LCSs is that consecutive matches are not required when in-sequence matches that showcase sentence-level order are given more weight. ROUGE-L and ROUGE-LSum are different implementations using different sentence splitting strategies and are identical if the sequences do not contain newline characters.

We used the ROUGE_score Python package (`https://pypi.org/project/ROUGE-score/`) for computation of ROUGE-N and ROUGE-L scores. For ROUGE-N, we took scores for ROUGE-1, ROUGE-2, and ROUGE-3, representing overlapping of unigrams, bigrams, and trigrams, respectively.

3.3.2 BERTScore

To gain insight into how the models perform beyond simply comparing content overlapping, we added BERTScore for evaluating semantic similarity. Analogously to common metrics, BERTScore computes a similarity score for each token in the candidate sentence with each token in the reference sentence. However, instead of exact matches, BERTScore computes token similarity using contextual embeddings derived from BERT model (Zhang et al., 2020). The implementation of computing precision, recall, and F1 for BERTScore was taken from the HuggingFace datasets package (`https://github.com/huggingface/datasets`).

4 Results

4.1 Training of Extractive Model

When we first trained this model, very serious over-fitting occurred. Accuracy on the training set reached 88%. However, on the testing set it dropped to only 67%. To handle this problem, we then added an L2 regularization to the model by specifying a weight decay parameter to the optimizer. We tried six weight decay values ranging from 0.005 to 0.16. The gap of accuracy scores on two sets narrowed down by 3% but the results are far from ideal. If time permits, there are still some ways to deal with the over-fitting problem. The first way is to reduce the complexity of the model: either choose a simpler pre-trained model with fewer parameters or reduce the dimensions of the pre-classifier and the classifier. The second way is to include more sources of training data and larger datasets.

4.2 Combining Extractive and Abstractive Models

Given the ratio of sentences included in summaries in the VT-SSum dataset, the output texts of our extractive model are mostly under 700 tokens long, which is ideal size to be fed into abstractive models for summary generation. We computed the average of ROUGE scores and BERTScores on every pair of generated summaries and ground-truth summaries in the test set of Yale dataset.

	Baseline BART	SBERT + BART	K-Means + BART
ROUGE1	0.3384	0.3319	**0.3420**
ROUGE2	**0.0747**	0.0687	0.0711
ROUGE3	0.0211	0.0207	**0.0228**
ROUGEL	0.2034	**0.2054**	0.2038
ROUGE-LSum	0.2034	**0.2054**	0.2038

Table 1: ROUGE scores for different models.

As we can see from Table 1, combining extractive and abstractive summarization indeed improved ROUGE scores, and our extractive SBERT model trained from lecture transcripts achieved better performance when we are comparing the average ROUGE-L scores.

For semantic-similarity evaluation, K-means clustering of sentence embeddings gives better results, although BERTScores do not vary much between different approaches we used.

	Baseline BART	SBERT + BART	K-Means + BART
Precision	0.8506	0.8516	**0.8528**
Recall	0.8545	0.8539	**0.8558**
F1	0.8524	0.8527	**0.8542**

Table 2: BERTScores for different models.

5 Conclusion and Future Work

Summarizing long lecture videos in textual form can be just as effective as summarizing them in audio form. Thus, to increase effectiveness of learning and save time for students we implemented this model. We tried combining unlike approaches and methods and took the advantage of producing improved summaries using abstractive and extractive methods. However, overfitting was an obstacle to achieving high accuracy of training the extractive model which was conducted on a rather large dataset. We are aware that there are ways to deal with this problem that can be implemented in future work. The first approach is to reduce the complexity of the model: either choose a simpler pre-trained model with fewer parameters or reduce the dimensions of the pre-classifier and the classifier. The second is to include more training data. Additionally, we plan to try to replace the transformer model with Transformer-XL to get rid of the limitation on text lengths in the abstractive model.

Human evaluation which looks at the readability and grammar of each summary (but does not evaluate content) and without showing the original transcript to the evaluator was excluded due to lack of time; it would have been interesting to explore as it could give further insight into the readability of the generated summaries. However, it will also be part of the future evaluation where we can develop by including linguistics metrics that evaluate grammar, unnecessary repetition, structure, and coherence of the generated summaries.

References

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages

4098–4109, Brussels, Belgium, October-November. Association for Computational Linguistics.

Yang Liu. 2019. Fine-tune bert for extractive summarization.

Tengchao Lv, Lei Cui, Momcilo Vasilijevic, and Furu Wei. 2021. Vt-ssum: A benchmark dataset for video transcript segmentation and summarization.

Derek Miller. 2019. Leveraging bert for extractive text summarization on lectures.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online, November. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun.

C.M. Taskiran, Z. Pizlo, A. Amir, D. Ponceleon, and E.J. Delp. 2006. Automated video program summarization using speech transcripts. *IEEE Transactions on Multimedia*, 8(4):775–791.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.