Summarization of Video Lectures using Transcripts

-Final Project Report

Rohit Kumar A S
Department of Computer Science and Engineering
PES University
Bengaluru,India
rohitkumaras24@gmail.com

Pavan Kumar Desai
Department of Computer Science and Engineering
PES University
Bengaluru,India
pavankumardesai39@gmail.com

P Deepak Reddy
Department of Computer Science and Engineering
PES University
Bengaluru,India
deepakreddy14@gmail.com

***Abstract*—In these days of relentless online classes and lectures, regular attendance and focus is a difficult activity. Automatic summarization[1] of these lectures is a beneficial and attractive prospect. Automatic summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning. In this paper, we use extractive text summarization[2] to generate summaries of youtube lecture videos using pre-generated transcripts. We utilise the pithy notion that a lot of the lecture content is composed of repetitive sentences. Hence, dissimilarity values are generated for every sentence with respect to every other sentence in the transcript. The sentences with high dissimilarity are included in the summary. Since there are no standard metrics present to evaluate the validity and effectiveness of such a lecture summary, we employed the turing test wherein we asked participants to give their opinion whether a summary was generated by a human or the model.**

I. Introduction

With the rapid growth of technology, most students now have access to their classroom lectures through the internet. Online learning has also shown significant growth over the last decade as many organisations came forward to provide people with the opportunity to gain new skills. Lectures may sometimes contain repetitive information or unnecessary interaction between students and teachers, which some students may not be interested in. Online lecture summarizer is a good solution to this problem. Students may want to quickly revise a topic/lecture but may find watching the whole lecture to be not worth-while. Lecture summarizer will provide the student with only the important information in the lectures and thus reading the summary takes very less time compared to watching the lecture. The summaries will also provide useful insights about the topics covered in the lecture and helps students make a sound decision whether it is the right kind of lecture the student is looking for. The text summaries for lectures can be generated given that there are transcripts available or can be auto -generated. The usual text summarization techniques for lecture summarization may not give desired results as lectures will have different pattern/format compared to articles or documents. So it's necessary to develop new techniques for online lecture summarization.

Given a long text, the job of text summarization model is to generate short and concise summary of the same. There are two types of text summarization models, 1) Extractive Text Summarization, which focuses on picking up sentences from the text which tend to convey overall information of text. 2) Abstractive Text Summarization [3] , which is more like a human describing the overall information in his own words.

Though the Abstractive Summarization is highly desirable, it is very challenging to produce summaries as it requires a lot of computational resources for training. Keeping these challenges in mind, we stick to extractive text summarization, which is also proven to be useful with recent advancements. Here we propose an unsupervised and extractive text summarization technique which uses PageRank for creating short summaries.

PageRank:

PageRank algorithm[4], which is used by search engines such as Google, is used to rank the webpages in the internet based on their inbound and outbound links. The basic idea here is that the rank of a page is calculated by the sum of contributions of all pages that link to it(inbound links), which is then equally distributed among all the pages that it links with(outbound links).

The same idea can be utilized here also, i.e ranking each sentence based on its similarity with other sentences and then selecting those sentences which have high ranks for creating summaries. This way selecting top-k sentences will create a summary with k sentences.

Sent2vec:

Sent2vec [5] is an open-source python library to create sentence embeddings i.e encoding a sentence into higher-dimensional vector space. Sent2vec has two approaches, 1) Bert[6] Language model 2) Word2vec[7] approach. Here we stick to the Bert Language Model. Bert (Bidirectional Encoder Representations from Transformers) which was introduced by google recently has achieved state-of-art results in various NLP tasks. Bert creates contextual meaning between words in a sentence using attention mechanisms. In our case, to generate a language model, only an encoder is sufficient which will output the embeddings.

II.Previous Work

Recently, BERT architecture has been incorporated to text summarization models which produced convincing results. Derek Miller(2019)[8] leveraged BERT for extractive summarization of lectures. Transcripts are taken from udacity lectures and the sentences are tokenized. The tokenized sentences are passed to the BERT model for inference to output embeddings and then clustered the embeddings with K-Means. The embedded sentences that were closest to the centroid are selected as the candidate summary sentences. One of the main drawbacks is that it doesn't perform well on large lectures(100 or more sentences).

Alexandra Savelieva, Bryan Au-Yeung,Vasanth Ramani(2020)[9] developed an abstractive model that is used with an encoder-decoder architecture, with a combination of pretrained BERT encoder with a randomly initialized Transformer decoder. The encoder portion is almost kept the same with a very low learning rate, with a separate learning rate for the decoder to make it learn better. The model generated summaries had good understandability and fluency, as claimed by the authors.

Ming Zhong, Pengfei Liu(2020)[10] took up the problem of text summarization as text matching. The principle idea is that a good summary should be more semantically similar as a whole to the source document than the unqualified summaries. Instead of scoring and extracting sentences one by one to form a summary formulating the extractive summarization task as a semantic text matching problem, in which a source document and candidate summaries will be (extracted from the original text) matched in a semantic space.

Chintan Shah , Dr. Anjali Jivani(2018)[11] adopted a A Hybrid Approach for Text Summarization Using Latent Semantic Analysis and Deep Learning. TF-IDF is used for feature extraction and latent Semantic Analysis is then used to pick significant sentences. A Three layer feed network with universal approximation Artificial Neural Network is then used where the sentence is checked to add as part of the summary or not.

Prabhjot Singh, Prateek Chhikara, Jasmeet Singh(2020)[12] developed an Ensemble model for Extractive Text Summarization. Various methods were used for word and sentence scoring.Ensemble

model is created which consists of the 7 base models. Parallel ensembling technique Is used where base learners generate results in parallel. The classification is then done based on the voting on the results Calculated by the base learners.

Sansiri Tarnpradab, Fei Liu, Kien A.Hua(2018)[13] used Hierarchical Attention Networks (HAN) to generate summaries for threads on different forums. Each word is in the text replaced by a pre trained word embedding using word2vec and fed to the neural network(LSTM).A bi-directional LSTM is used where two vectors, where one which carries over the semantic information from the beginning to the current step and another encodes the information from the current step to the end of the sentence.
Global saliency is captured by projecting the word representation to transformed space.The sentence vector Si is generated as a weighted sum of word representation.A sequence of sentence vectors is taken as input to give output of a thread vector and each sentence is represented as a concatenation of corresponding sentence and thread vectors. Cross-entropy loss is the cost function used.

III.Data

Various lecture videos from MIT OpenCourseWare[14] channel with captions/transcripts available on Youtube are put into a playlist for summarization. A python package called youtube_transcript_api is used to fetch details of all videos in the playlist and get_transcript() function is then called to get the transcripts of the videos in a string format. Necessary pre-processing is then done to each transcript.

IV.Proposed Solution

The algorithm used can be broadly categorised into the following steps

I. PreProcessing :

- Segment transcript into sentences
- Join conjunctive sentences
- .Remove stopwords
- Lemmatize[15]

II. Build Dissimilarity Matrix

- Split transcript sentences into batches
- Encode sentences using Bert vectorizer
- Calculate cosine similarity[16]
- Subtract cosine similarity from ‘1’ to generate dissimilarity

III. Pagerank

- Use dissimilarity matrix to calculate pagerank of every sentence
- Use damping factor = 0.85 and largest permissible deviation= 1.0e-8

IV. Select summary sentences

- From every batch select 50% of the top ranked sentences in chronological order
- Concatenate all the batches’ summary sentences

Preprocessing

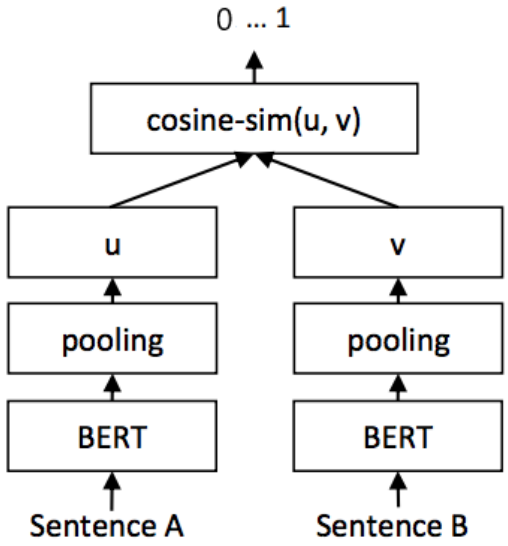
1. Split the transcripts into sentences based on newline as delimiter
2. Tokenize the sentence into words
3. Join sentences starting with conjunctions (and,but,because,so,it) with the previous sentence
4. Remove sentences with less than 4 words
5. Remove stop words
6. Expand contractions (such as I’m => I am, can’t => cannot, I’ve => I have)
7. Tag POS (Part-Of-Speech) for every word using nltk’s ‘pos_tag’ function
8. Lemmatize using WordNetLemmatizer with help of the POS tags generated

Dissimilarity-matrix

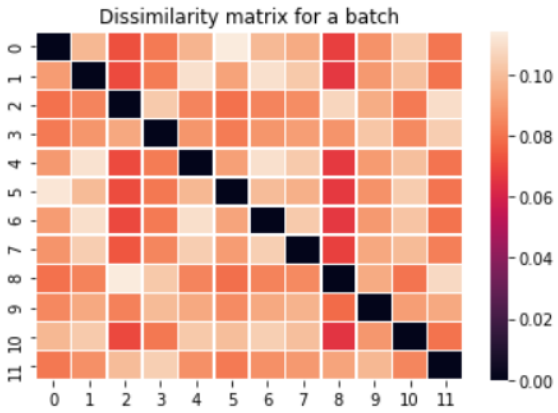
A dissimilarity matrix is a matrix of real values that denotes the dissimilarity between the row element and the column element.

Split the entire transcript into batches of 12 sentences each. For every pair of sentences, generate real-valued word embeddings using sent2vec’s BERT vectorizer. Use the vectors for the two sentences to

calculate the cosine similarity. Subtract it from the number ‘1’ to obtain the dissimilarity.



Similarly doing this for every pair of sentences in the batch, a dissimilarity matrix of dimensions 12 X 12 is generated for every batch, wherein the value in every cell denotes the dissimilarity index (1 - cosine similarity) between the sentences in that row and column.



Lighter colors indicate higher dissimilarity

Page rank

Once the dissimilarity matrix is generated for each batch, the ranks of sentences in each batch is generated using the page rank algorithm.

The initial ranks of the sentences are randomized and then normalized by dividing with the highest rank. Then a damping factor of 0.85 is used so as to not completely ignore the sentences which have extreme dissimilarity with other sentences.

Hence the matrix used for the pagerank iterations will be calculated as:

$$\widehat{M} = d * M + \frac{1-d}{no.\,of\ sentences}$$

We calculate the new rank vector as a product of the matrix vectors with the rank vector.

This is done repeatedly until none of the ranks change more than the largest permissible deviation of 1.0e-8

$$v = \widehat{M} * v$$

Select summary sentences

After Pagerank, all the sentences are associated with their respective ranks. We then concatenate the ranks of all batches and then sort the sentences according to their rank(high to low).After this we select the top ‘k’ percent of sentences(k $\in$ (0,100]). Thus, a k-percent summary of the respective lecture is generated.

IV.Experiment and Results

Here is a summary generated for lecture on physics taken MIT OpenCourseWare youtube channel:

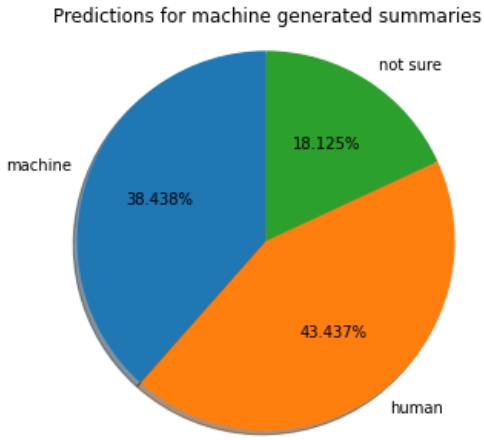
“ This is something you've undoubtedly seen before . Now, if m is a constant, then I can rewrite this as F is

equal to the time derivative of the mass times the velocity, or equivalently as the time derivative of the momentum, since mv is just equal to p. So I actually want to stress this much--I'm going to put it in a box because it's very important that I can write Newton's second law, instead of F equals ma, as F equals the time derivative of the momentum . In all of those cases, this form of Newton's second law is correct . What I want to do now though, is to take a closer look at this equation, force is equal to the time derivative of the momentum . Whenever we have a relation involving derivatives like this, we can always also rewrite it in an equivalent integral form, which can be very useful and give us a different way of looking at the same information. So let's take a look at that. So if I take this equation and integrate both sides with respect to time, then I can write that as the integral of F with respect to time is equal to the integral of the right-hand side, dp/dt with respect to time . Now, let's make this a definite integral . And suppose I had some complicated function that looked like that . The impulse is just the area under this curve . Going over the same time interval."

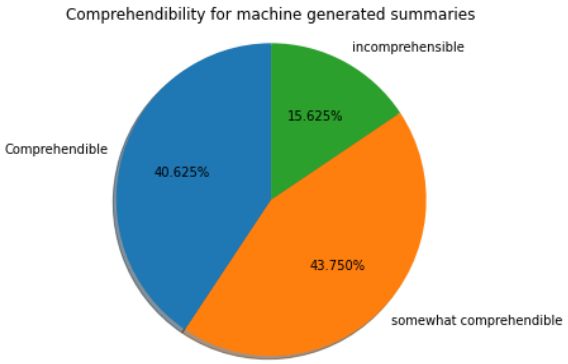
Since it was infeasible to generate golden summaries for every lecture and perform word-by-word/ ROGUE comparison to analyse the efficacy of the generated summaries, we went with a turing-test approach wherein 10 machine generated summaries and 5 human generated summaries were intermixed in random order and participants were asked to opine as to the origin of the summary (human or machine). The following confusion matrix illustrates the obtained results :

	Predicted as Machine	Predicted as Human
Machine	123	139
Human	68	65

As can be seen from the matrix, a precision score of 53.05 % was obtained, meaning our model managed to fool humans roughly once in every two instances.



We also asked users to select one among the three [‘completely comprehendible’, ‘somewhat comprehendible’, ‘incomprehensible’] for every summary. For the machine generated summaries, 40% instances fell into the ‘completely comprehendible’ category while 16% fell into the incomprehensible category.



V.Conclusion

In this paper we have implemented a pagerank based solution for extractive text summarization by making use of dissimilarity matrix. Results observed

were moderately successful but in-line with expectations.

One noteworthy observation was the drastic increase in quality of the summary observed when sentences beginning with conjunctions were considered a part of the previous sentence.

The main challenges faced was the time constraint: time required to generate summaries goes on increasing as the number of sentences in lecture transcripts increases.

VI. Future Work

Further work can be focused on generating a standard dataset of online lecture transcripts and manually annotated summaries. This dataset can be used for an abstractive summarization method to generate better and more compact summaries. Extractive summarization performs poorly when most sentences contain subject pronouns and hence good summarization can be achieved when the model is extended to understand what the subject pronouns is referring to, and therefore subject pronouns can be replaced with what they are referring to in the final summary.

VII. References

[1]https://en.wikipedia.org/wiki/Automatic_summarization

[2] https://medium.com/sciforce/towards-automatic-text-summarization-extractive-methods-e8439cd54715

[3] Zheng, C., Zhang, K., Wang, H.J. and Fan, L., 2020. Topic-Aware Abstractive Text Summarization. arXiv preprint arXiv:2010.10323.

[4] Wikipedia, “Pagerank”, https://en.wikipedia.org/wiki/PageRank.

[5] Matteo Pagliardini, Prakhar Gupta, Martin Jaggi, 2017, “Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features”.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”.

[7] https://en.wikipedia.org/wiki/Word2vec

[8] Derek Miller. 2019. “Leveraging BERT for Extractive Text Summarization on Lectures.” Atlanta, Georgia.

[9] Savelieva, A., Au-Yeung, B. and Ramani, V., 2020. “Abstractive Summarization of Spoken and Written Instructions with BERT.”

[10] Ming Zhong et al. 2020. “Extractive Summarization as Text Matching.”

[11] Chintan Shah, Anjali Jivani, 2018, “A Hybrid Approach of

Text summarization Using Latent Semantic Analysis and Deep Learning.”

[12] Prabhjot Singh, Prateek Chhikara, Jasmeet Singh, 2020. “An Ensemble Approach for Extractive Text Summarization.”

[13] Tarnpradab, S., Liu, F. and Hua, K.A., 2018. “Toward extractive summarization of online forum discussions via hierarchical attention networks.”

[14] https://www.youtube.com/c/mitocw/videos

[15] Wikipedia, “Lemmatization”,https://en.wikipedia.org/wiki/Lemmatisation.

[16] Wikipedia,“Cosine Similarity”, https://en.wikipedia.org/wiki/Cosine_similarity.