Building NLP Models From NLP Lecture Transcripts

Themistokils Haris, Themistoklis Nikas[1]

Abstract

This project report describes a series of approaches taken to explore the vast landscape of text summarization and generation tasks in NLP. We examine a series of techniques, both from neural network architectures (Transformers, LSTMs, RNNs) and more traditional machine learning algorithms (extractive summarization) for summarizing a piece of text. We apply these techniques to an interesting dataset: the lecture transcripts from the Fall 2023 edition of the BU CS 505 (NLP) class, in hopes of accurately extracting meaningful and digestible information about the various NLP concepts outlined. After extensive experimentation with a variety of data processing techniques, we achieve note-worthy summarization results with relatively high ROUGE metric scores. We overcome major challenges relating to the quality and immense quantity of the transcript data, using a layered summarization algorithm that we produced independently, without observing it in the relevant literature. Finally, mostly for curiosity's sake, we fine-tune state-of-the-art text generation models like GPT-2 on the lecture transcript dataset and obtain an interesting AI analog of our beloved Professor Snyder.

Keywords: NLP, Deep Learning, Summarization, Concept Generalization, Data Processing

Code: tkhar/lecture_summary: Scraping public lectures and summarizing them via neural models. (github.com)

Introduction

Text summarization is a very important NLP task with applications in many fields of work, from education to law and logistics. It is, however, a difficult task because it is very context dependent - extracting meaningful information from text might require drawing elaborate connections between seemingly unrelated parts.

For this reason, even though the problem has been studied since the 1960s, most techniques would fall short of human-level summarization. Early techniques focused on **extractive summarization**, where the goal is to choose a subset of the sentences that contain the most relevant information. In conjunction with techniques like POS tagging and named-entity recognition, pipelines were built that were able to reasonably condense a text to its major points of information.

[1] See last page for individual team member contributions.

With the advent of neural networks in the last couple of decades however, things have drastically changed as the field has moved on to the more general task of **abstractive summarization**, where preserving the language of the original text is no longer a requirement. Neural networks with varying degrees of context-keeping capabilities can output meaningful, creative and extensive summaries. The disadvantage of these techniques is that they tend to repeat themselves and have a hard time dealing with Out-Of-Vocabulary tokens and words (OOV). Our results will also showcase this weakness.

The most successful techniques are arguably the **hybrid summarization** techniques, which aim to build pipelines that combine neural architectures with more functional intermediate representations that can extract not only semantics but also meaning.

For a comprehensive survey on the advantages and disadvantages of different existing techniques please see [1] and [2].

To evaluate the accuracy of summarization models, two main metrics have been used: the BLEU and the ROUGE metrics. The ROUGE metric, or Recall-Oriented Understudy for Gisting Evaluation, is a metric designed specifically for the evaluation of summaries. There are many ROUGE metrics, but we focus on ROUGE-N, which measures the n-gram overlap between the summary and the reference summaries. N can be 1, 2, or more, capturing overlap at word level, bigram level, etc. Higher ROUGE-N scores indicate more overlap with the reference summaries. This evaluation happens in training and validation time because we possess reference summaries. During test time, quality has to be checked empirically, where we judge based on factors such as readability, compression and expressivity.

Contents

This report is structured as follows:

1. In the next section we will provide an overview of the summarization models we employed and ways they can be compared to one another.
2. After that, we will discuss our datasets and data processing techniques and pipelines.
3. Then we will present our results.
4. As an aside, we also present a discussion on fine-tuning generative models on our dataset.
5. Finally, we conclude with future directions to this project and lessons learned.

Summarization Models

State-of-the-art abstractive summarizers are transformer-based architectures, like BERT, T5 and BART. We use the small versions to ensure we can handle the computational resources required. We summarize them in the table below:

Model	Number of Parameters	Layers	Hidden Layer Size	Comments
BERT-Small	110 million	12	512	Encoder-only

				architecture, focuses on masked language modeling and sentence representation.
T5-Small	220 million	24	768	Encoder-decoder architecture, excels at sequence-to-sequence tasks like translation, summarization, and question answering. Trained on Colossal Clean Crawled Corpus (C4).
BART-Small	137 million	12	768	Encoder-decoder architecture with additional features like denoising and infilling, suited for text generation and summarization. Fine-tuned on CNN Daily Mail.

We also experimented with LSTM architectures: Our architecture of choice is a decoder-encoder Seq2Sec LSTM-based architecture, with a series of attention layers. In a sense, it is a precursor to the transformer model. See the figure below for a rough sketch of this architecture:

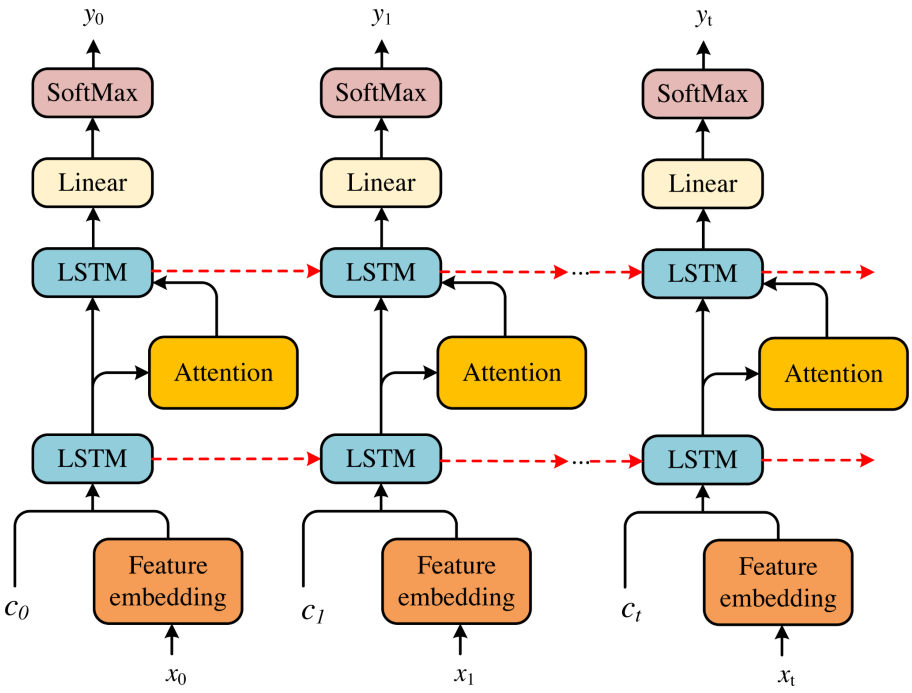


Figure 1: Encoder-Decoder LSTM Seq2Seq architecture with Attention

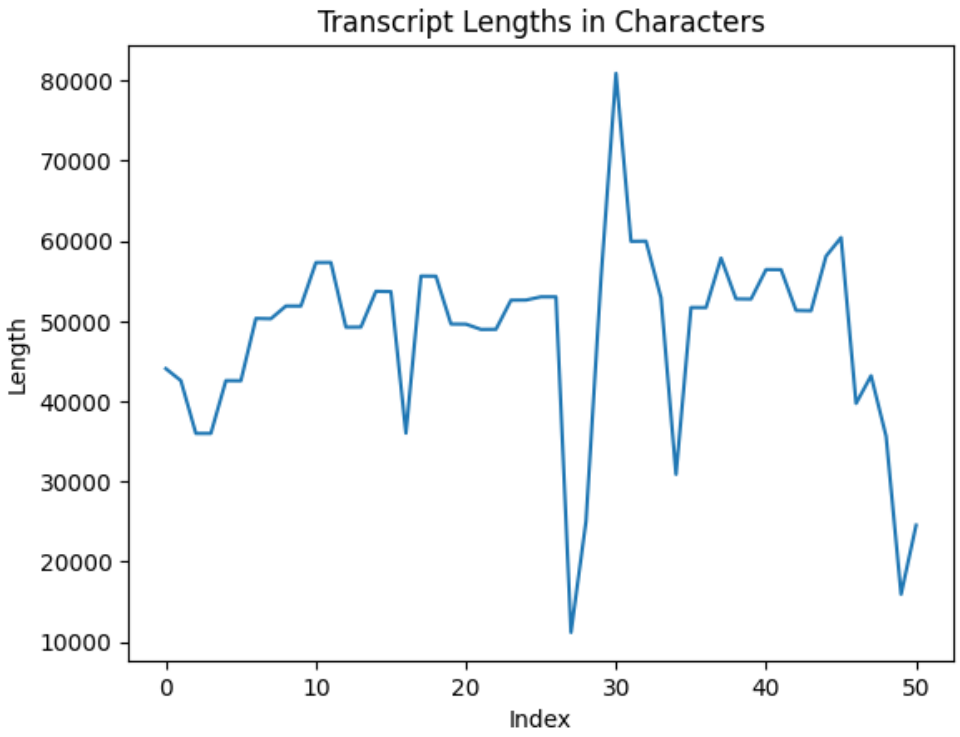
TODO: Write about the LSTM architecture and how it was built.

Our Dataset and Data Processing Pipeline

The novelty of our project lies in the dataset we chose and the data-processing pipeline we established. This section will describe these in detail.

Our dataset was scraped from Professor Snyder's YouTube Channel: (127) Wayne Snyder - YouTube. The lectures for our class have been publicly available there, and we can use Youtube's API to enumerate them and send a specific GET request to obtain transcripts for them.

Enumerating the lectures is easily done by sending a content GET request via a valid API key. The result is a JSON string which can be broken up into GUID identifiers for each video. Then, we take these identifiers and use Youtube's undocumented API to get the automatically generated transcripts, which come to us also in a JSON format. The JSON contains two pieces of metadata: transcript sentences, along with their timestamps. We can then extract the transcript sentences and piece them together to form the transcript of the entire lecture. The results are rather long pieces of text which we then have to process. As a reference, the lengths of the transcripts are shown below:



Our main goal is to summarize each of those transcripts. However, as evidenced by the graph above, the length of each transcript is too big to fit in a model. Even if we were to break up the transcripts into 1000 pieces it would barely be big enough to fit in a single model evaluation. To make matters worse, calling a model 1000 times would result in a single summary taking hours to produce, which is prohibitive for us.

To solve these problems, we put each transcript through a data-processing pipeline designed to remove extraneous information and information that has not been properly generated by the transcript mechanism.

- ☐ Remove Out-Of-Vocabulary words using the `TextBlob` library.
- ☐ Remove stopwords by using the NLTK list.
- ☐ Stemming and Lemmatization
- ☐ Remove repeated words by using a regular expression: `(\b\w+\b)(?:\s+\1)+`
- ☐ Split the transcript into thematically similar parts by using the `SpaCy` NLP library.
- ☐ Concatenate batches of these parts heuristically in order to result in chucks of around 500 tokens each.

After we have generated a list of batches for each lecture, we put it through a **summarization pipeline**. Given a summarization model, the pipeline proceeds in a layered fashion as follows:

- ☐ Summarize each fragment for a transcript.
- ☐ Batch the summaries into larger segments.
- ☐ Summarize the segments again.
- ☐ Repeat for a number of layers until a summary of the desired size is produced.

The results of this process are outlined below.

We conclude this section with an important note about the quality of the data we were working with. Apart from the prohibitively large size of the dataset, we were working with quite poor transcript results. Youtube's algorithm can definitely generate semi-accurate transcripts, but it fills the data with slightly mispronounced words and misleading tokens, which we cannot use and therefore had to get rid of. Furthermore, the transcript timestamps were overlapping, meaning that we could not use them to separate sentences - no matter how much we tried. We thus had to resort to using SpaCy for thematic separation, but its results are naturally not ideal. Overall, our methods would probably have been a lot more successful and efficient on a properly formatted and created dataset, but we were limited by that factor.

Please see our code to learn more specific information about the ideas outlined above.

Results

Efficiency of Summarization Techniques

The table below shows the efficiency of various models being used on the text summarization task. Apart from pre-trained transformers and our LSTM architecture, we also examined a hybrid approach as detailed in [3], although we did not end up implementing it.

Briefly, the approach in [3] combined an LSTM-based Deep Architecture for abstractive summarization with the idea of **concept generalization**. Generalization is a functional method that allows for inference of semantic and thematic properties of text. In the paper, a taxonomy of concepts is used to generalize sentences and then process the generalizations via deep neural networks. After that, an *inverse generalization* is done to get the final result. The figure below from the paper shows an example of concept taxonomy:

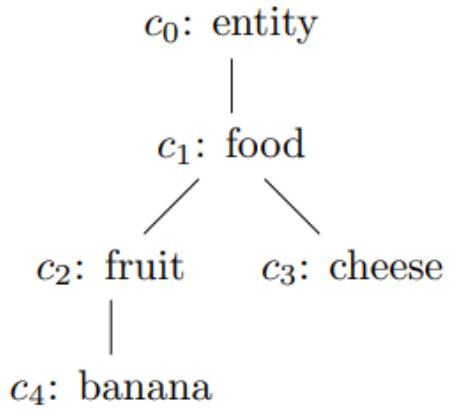


Figure 1: A taxonomy of concepts.

The table below shows the results obtained from each method we considered. We look at ROUGE-2 which is effectively the bigram overlap of summaries.

ROUGE-2 (A, B) = # of 2-grams that appear in A and B / # of 2-grams that appear in B

The results below are based on test with the following prompt to a summarization model (note we have to include the word “summarize” in the prompt)”

```
“Summarize: The Inflation Reduction Act lowers prescription drug costs,
health care costs, and energy costs. It's the most aggressive action on
tackling the climate crisis in American history, which will lift up
American workers and create good-paying, union jobs across the country.
It'll lower the deficit and ask the ultra-wealthy and corporations to pay
their fair share. And no one making under $400,000 per year will pay a
penny more in taxes.”
```

Our baseline approach is a standard deep learning state-of-the-art summarizer (like BERT) [3]

Model	ROUGE-2 score for summarization
Baseline (BERT)	0.657
T5-Small	0.618
BART (CNN finetuned)	0.81

Lecture Summarization Examples

ROUGE and other related metrics can only evaluate the quality of summaries given references. Our transcript summaries have no references to compare to, so the best judge is human understanding. In this section, we present some examples of the summaries we got using our interface:

Example 1**:** Lecture on Transformers [*slightly edited]*
“I asked people in the department what materials they had in Transformers so they gave me100 slides. If you're trying to characterize what words are important, really really common words are not important because they're in all documents. You start training if you don't rerun the model when you create the model it initializes it with random parameters then we can refine it by using document frequencies making words using the idea of how often they occur in different documents.”

Example 2: Lecture on RNNs

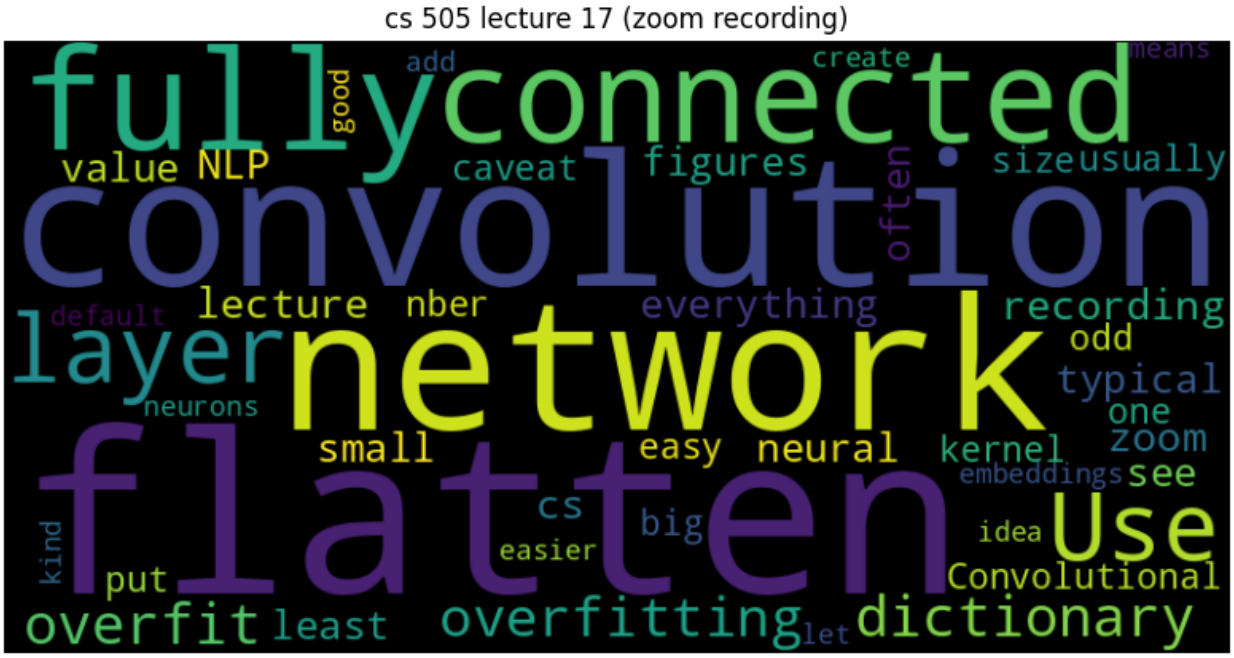
“We're going to [..] get into the really cool stuff for current networks …. The activation here [is] from the previous step [...] it's going to get around like this: this is the same neuron at different moments in time in the sequence there for the Loop controlling process . A lot of different configurations [...] an example of this give it a vector and have it generate a state [...] a nice example is image captioning. It can be done with advantages [...] In networks we can normalize between layers so you have a layer and the information goes through the layer. Now we used to do twitter and then now you can get a lot of data it's really hard to say "see I've never done”.

It is clear that the summaries created are not coherent, end-to-end summaries. Apart from the data quality, that’s also because we are lacking much of the information, like Professor Snyder’s slides, and his gestures towards them. So the model does not nearly possess all the information needed to make a good summary.

Word Cloud Generation

However, it is worth noting that even with those setbacks we obtain summaries that grasp basic ideas of each lecture, allowing us to at least know what is being discussed and some aspects about it. To show this more accurately, we created word clouds for each summary and cross-referenced them with the syllabus of the class. We directed our attention towards Zoom lectures, where we anticipated a clear, or at the very least, a more accurate transcript.

The initial word cloud pertains to Lecture 17, focusing on neural network tuning, advanced features, and strategies to mitigate overfitting. By examining the word cloud, we observe keywords related to neural network architecture and specifically note the presence of the term **'overfitting'**!

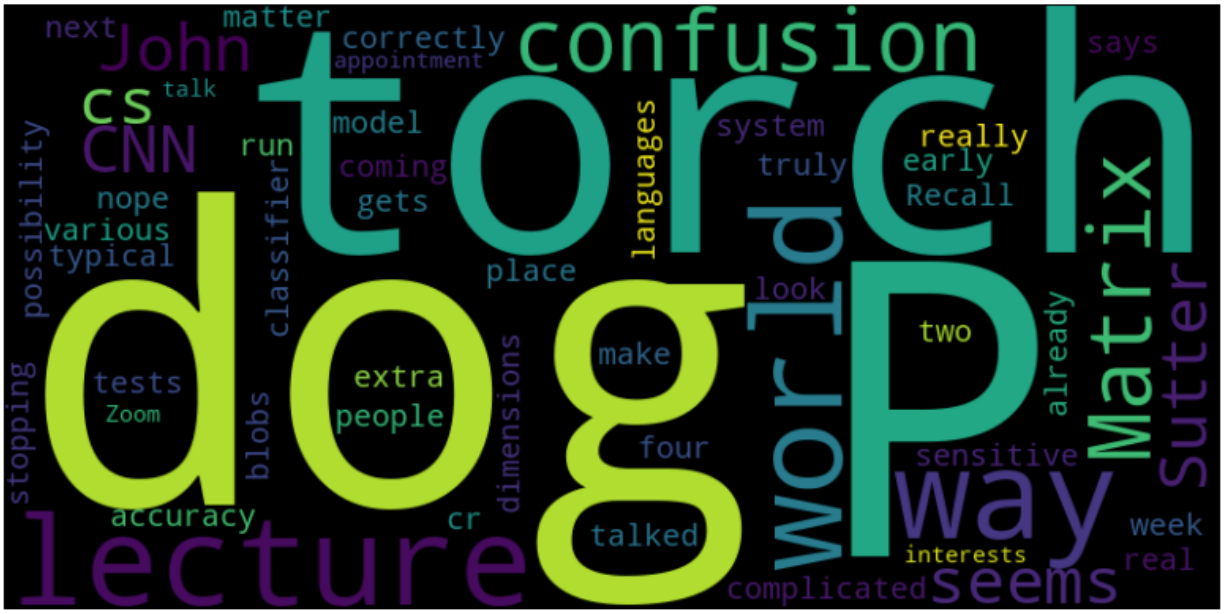


Moving on to the word cloud for Lecture 11, which delves into the evaluation of classifiers and generalization, as sourced from the course webpage. Notably, the lecture primarily featured a binary classification task involving the prediction of whether an image contains a dog or not. The word cloud accurately reflects this main example, though it lacks substantive content.

Furthermore, the word cloud highlights a key term related to a well-known classification problem and its corresponding dataset, 'Blobs.' Despite the challenges posed by the transcript, the summary also includes essential terms like confusion matrix, accuracy, classifier, Recall, tests, and early stopping. Collectively, these terms shed light on the lecturer's core focus: evaluating classifiers. Finally, one could potentially gather that the primary datasets and

examples used in the lecture were related to images of dogs and the Blobs dataset.

cs 505 lecture 11



Lastly, we present the word cloud for the guest lecture by Jena Jordahl, a Generative AI and Model Tuning Specialist currently at Google. Three prominent words stand out in the center of the word cloud. While two of them align with the speaker's background, the inclusion of 'wild' might seem unexpected. A closer look at the lecture slides reveals the title as 'NLP in the Wild,' explaining its prevalence. However, the appearance of 'MIT' may seem intriguing since the speaker is a BU graduate. This can be attributed to the lecture's emphasis on examples from MIT's curriculum, providing context to these terms within the word cloud.

cs505 guest lecture 21 slide view

Sanity testing: Similarity Metrics between Transcripts

Most YouTube lectures were accompanied by two versions: one capturing the full classroom view and another focusing on the slides. Due to the camera's positioning in the middle of the classroom, the full view lacked visibility of the slides. This dual video approach resulted in two transcripts for the same lecture, ideally reflecting identical content. We will now examine the summaries of these transcript sets using the aforementioned summarization models.

The experimental design unfolds as follows: initially, we utilize the baseline model to generate summaries for each transcript. Subsequently, we evaluate the similarity between summaries within the same lecture and compute the average similarity across pairs of transcripts. Maintaining the same approach, we then apply the more advanced summarization model and calculate the corresponding similarity scores. Ultimately, we compare the two sets of similarities to determine the superior model.

To assess the similarity between these summaries, we employ cosine similarity, a widely used and established technique in natural language processing for measuring similarity in text data. Cosine similarity measures the cosine of the angle between two vectors, representing the textual content of the summaries in our context. The resulting similarity score ranges from -1 to 1, where a higher value indicates a closer alignment in content and a lower value suggests greater dissimilarity. A score of 1 signifies identical content, 0 indicates no similarity, and -1 denotes complete dissimilarity.

Model	Lowest sim	Highest sim	AVG sim
T5 small	0.48	0.99	0.80
DistilBart	0.63	1.00	0.84
BartLargeCNN	0.25	1.00	0.79

We can observe that the average cosine similarity between the two summaries of the same lecture is above 0.8, indicating a high level of agreement in content. This suggests a consistent depiction of the lecture content in both the baseline and the more advanced DistilBart and Large Bart models. It is important to note that there is no significant difference in the average cosine similarity between the two classes of models, despite the advanced sophistication of DistilBart and Large Bart respectively. It is worth mentioning that although Large Bart is the most advanced model, it also reported the lowest similarity.

We measured the lowest cosine similarity between the transcripts of lecture 23, and even after carefully looking at both videos, we still do not know what caused this difference. In terms of the highest similarity, numerous pairs of lectures achieved the maximum, signifying identical content. Notably, this was observed particularly in Zoom lectures, characterized by enhanced sound quality. Consequently, the transcripts obtained from the YouTube API were of higher quality, aligning with the expected outcome.

Generative Model Fine-Tuning

After finding inspiration in Homework 6 and obtaining transcripts from lectures on Youtube, we became intrigued by the possibility of fine-tuning a model to generate text that captures the distinctive style of Professor Snyder's teaching.

We encountered numerous challenges and variations compared to Homework 6, with two major hurdles. First, the nature and size of the data posed significant obstacles. Second, we faced constraints due to the extremely limited resources available for running our experiments. Despite utilizing SCC for all tasks, the insufficient GPU memory proved to be a notable limitation.

To overcome these obstacles, for this experiment we did not utilize the GPU, so the data can fit in the memory. The reduced number of training epochs allowed us for such a modification. Additionally, we employed the distilled GPT2, which was developed using knowledge distillation and was designed to be a faster, lighter version of GPT-2. Finally, in terms of preprocessing, we removed stop words as well as any word that does not appear in the English vocabulary.

Evaluating fine-tuned models presents a challenging task. To facilitate comparison, we opted to showcase and contrast the outputs of the original distilled GPT-2 and the fine-tuned model. For prompting the generative models, we selected a phrase that we believe captures Professor Snyder's favored lecture introduction: '**Ok folks**'.

Below, we present the outputs (additional details available in the corresponding notebook).

```
print(snyderpt('Ok folks'))
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'Ok folks I know going along with some fun stuff like the last time around I actually got to talk about going through life pretty much every semester pretty much everything going going forth really c
ool stuff like trying to dig into these little things trying to learn new stuff'}]
```

+ Code + Markdown

```
print(gpt2('Ok folks'))
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'Ok folks. If there were a single individual I would always be happy to let them know what I think that is, or a specific opinion. So I hope this guide helps me have a chance to ask a little bit abou
t how to create the idea'}]
```

While the output of the fine-tuned model may not resemble a typical lecture, it does manage to capture some of the excitement that Professor Snyder brings when teaching about state-of-the-art technologies, as we experienced in this class. Conversely, the vanilla GPT-2 model produced a more generic output. It's important to note that both models exhibit a lack of coherence.

```
print(snyderpt('neural networks'))
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'neural networks that can also offer data analysis information about natural phenomena.\n\n\n\nThis paper discusses the importance of deep learning among many topics related to deep learning.\nThis p
aper also discusses how deep data storage works and how deep learning works.'}]
```

+ Code + Markdown

```
print(gpt2('neural networks'))
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'neural networks that can provide a platform to support both the client and the server. If the server is a server that’s been hacked recently, you must be a registered user for a user account.\n\n\n
\n\nIf you were'}]
```

In the second generative example, the fine-tuned model associated the prompt 'neural networks' with academic papers, revealing a more refined contextual understanding, likely

acquired during the fine-tuning process on the lectures. In contrast, the vanilla model failed to establish this connection.

Future Directions and Lessons Learned

To conclude our report, we discuss the challenges we faced, lessons learned and future directions.

Among our greatest challenges was the handling of our dataset in a way that would allow us to extract maximum information from it. Among the things we wanted to try but didn't have the time for was using intermediate learning tasks such as sentence separation and POS tagging as part of our pre-processing. SpaCy definitely does some of that under the hood, but we wanted to be explicit with it and see if it improves our performance at all.

The most important lesson we learned through this project is the importance of having good, well processed datasets when handling NLP tasks. Even the best, most sophisticated algorithms will fail miserably on a poor dataset, and data visualization and processing skills are extremely important in these cases. It may be true that "the more data the better", but really it should be "the better data the better".

This project is far from being complete. Apart from a wide variety of summarization techniques we haven't tried yet, there are approaches to getting better transcripts by utilizing more sophisticated transcript pipelines. Another approach would be to directly use different transcript-making techniques on the video files themselves and see if we got better results. In any case, we believe that our original vision of automatically summarizing lectures can have a lot of positive impact on educational purposes, so it is worth pursuing further and iterating so that better and more robust results can be obtained.

Bibliography

1. Wang, Guanghua, and Weili Wu. "Surveying the Landscape of Text Summarization with Deep Learning: A Comprehensive Review." arXiv preprint arXiv:2310.09411 (2023).
2. Allahyari, Mehdi, et al. "Text summarization techniques: a brief survey." arXiv preprint arXiv:1707.02268 (2017).
3. See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017).
4. Kouris, Panagiotis, Georgios Alexandridis, and Andreas Stafylopatis. "Abstractive text summarization based on deep learning and semantic content generalization." (2022).
5. Abstractive Text Summarization Using Transformers | by Rohan Jagtap | The Startup | Medium
6. A Gentle Introduction to Text Summarization - MachineLearningMastery.com
7. Text Summarization with T5, PyTorch, and PyTorch Lightning | by Ajazahmed Turki | Medium

Team Member Contributions

Themistoklis Haris

- [ ] Experimented with Transformer Summarization Models and obtained baselines and ROUGE-2 scores.
- [ ] Scraped the transcript data using YouTube's API
- [ ] Created the data processing pipeline
- [ ] Created the layered summarization pipeline algorithm.
- [ ] Wrote the final project report

Themistoklis Nikas

- [ ] Word Cloud Generation
- [ ] Similarity Metrics between summaries
- [ ] LSTM model experimentation
- [ ] Transcript-based fine-tuning
- [ ] Code cleaning and functionality