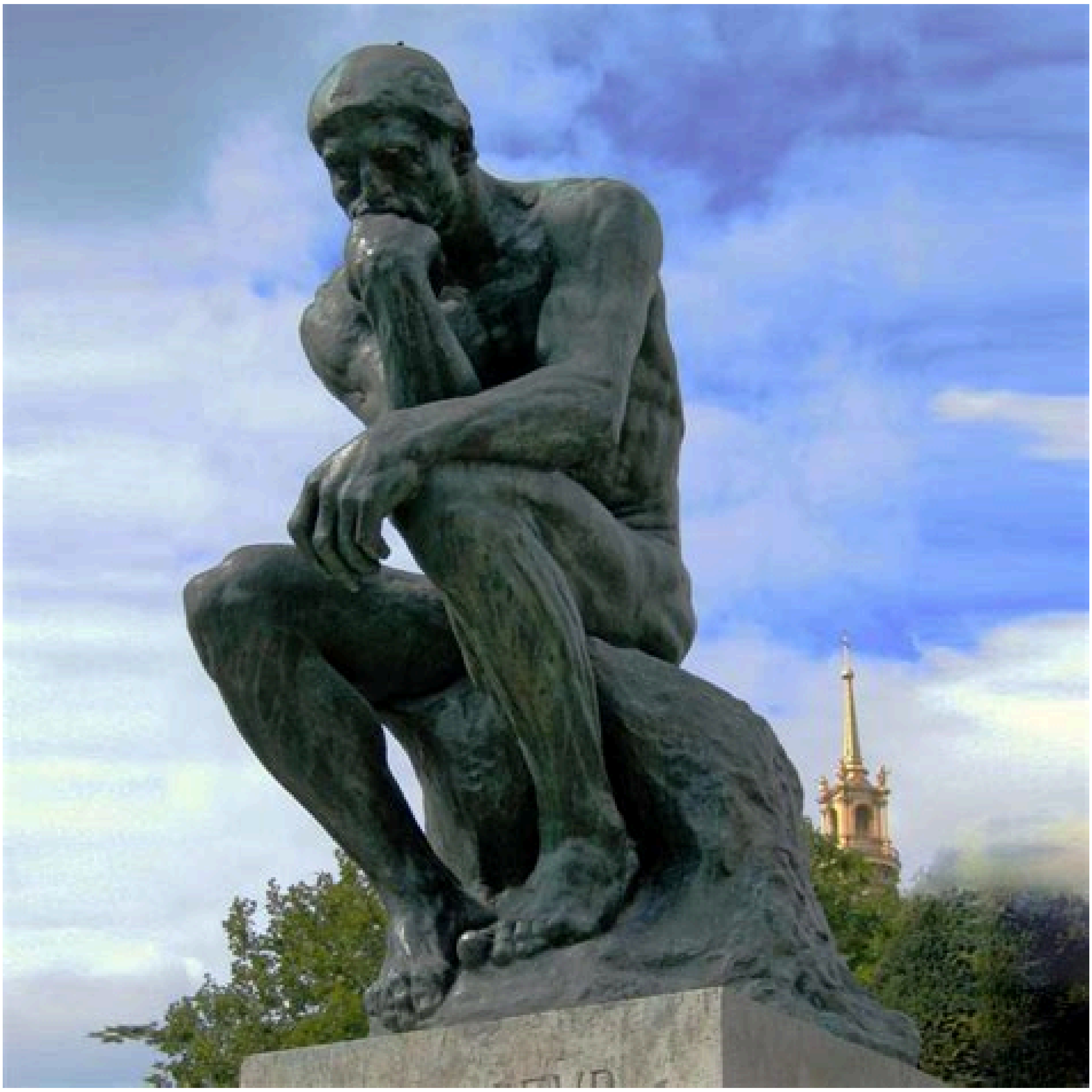
Large Language Models for Multi-Video Transcript Summarization

[a]Thorkild Kappel · [b]Rahul Tangellapalli Sai Hanuma· [c]Zhuolin Li

[a,b,c]Aarhus University

[a]202207326@post.au.dk · [b]202502250@post.au.dk · [c]202502085@post.au.dk



1 Introduction and Motivation

Information extraction and question answering are two of the key subjects when it comes to Natural Language Processing (NLP) [1], [2], [3]. Ever since the new era of Artificial Intelligence, sparked by the invention of the attention mechanism [4] as well as processing power to scale models to billions of parameters, these subjects have changed in terms of methodology [5]. Summarization models are typically trained using text to summarize datasets like TL;DR, which provide a training paradigm for models to learn how to compress key information from lengthy texts [6], [7]. However, this fails to address the scenario where the summary might benefit from multiple sources of information being integrated into the summary together [8]. Using multiple sources instead could potentially mitigate the risk of misinformation if sources disagree [9]. Further, having multiple perspectives and opinions might reveal additional sides to view a problem from that a single source cannot show.

Video platforms such as YouTube represent an important medium for information seeking, especially for explanatory queries [10]. YouTube transcripts are closer to spoken languages and typically less structured, which often include extensive explanations, examples, and subjective opinions [11], and are also typically more verbose [12]. In practice, users assess credibility and gather information from multiple video-based sources when engaging with content on platforms like YouTube [13]. Therefore, in this project our goal is not to summarize a single transcript in isolation, but to make a product that aggregates transcripts from multiple relevant videos based on user queries and generates a comprehensive summary that answers the query while highlighting commonalities and differences between videos when necessary.

Existing summarization datasets do not directly support tasks that rely on user queries or require explicit comparison [14], [15]. To minimize this gap, we constructed a custom dataset. Target summaries are generated through prompts to large language models, enabling subsequent fine-tuning of smaller models. Through this process, the models learn to integrate information across multiple transcripts and produce concise summaries. In our experiments, we compare a compact open-source base model with two fine-tuned variants and evaluate their outputs against the reference summaries using semantic similarity metrics.

2 Methodology

2.1 Datasets and Preprocessing

For this project, two datasets have been used. The first dataset used is the "Reddit TL;DR" dataset [16], which has the purpose of being used for transfer learning, because of the large amount of data it contains. A second custom dataset with less data has also been created, as the gold standard we are trying to achieve.

The first dataset contains 116,722 posts scraped from Reddit that specifically contained a TL;DR (Too Long Didn't Read) section. This is a section in the post for readers who do not wish to read the full post because it was too long. Once imported, we first split the data into samples that contain for each post: the title, text in the post, and the TL;DR. Then downsampling was done to make each subreddit have at max 2000 posts in our dataset. Afterwards, to put it into a similar format as the second dataset for the purpose of better transfer learning, grouping of the posts based on similarity was done using a greedy grouping algorithm.

To group posts together, first a TF-IDF vectorization (1) of the post texts was done. Then for each post, their 50 most similar posts in terms of cosine similarity (4) were calculated. After this, starting from one end of the data, formatted in no particular order, samples were grouped together with the three posts that were the most similar, and that had not already been grouped in another group. Furthermore, if samples are to be grouped, they must be above a cosine similarity of at least 0.3 and below 0.95 to ensure enough similarity and to ensure duplicate posts are not grouped together respectively. For a group to be formed, it must also contain at least two samples, meaning that if all 50 closest samples are already in a group, we note the sample as ungrouped for now in the algorithm, but it might be grouped at a later repetition. Groups that were ungrouped by the end were then simply thrown away.

For concatenating a group into a single sample, we made a specific format. It is created by adding each post one by one to a single string the delimited by a line separation, as well as prepending its title and a post number between one and four. For grouping the TL;DR's, 36 specific prepends were made with the intention in mind that they should be similar to the second dataset. An example of such a prepend that we used is "Transcript {a} discussed:", where "{a}" will be replaced with the transcript number, and the prepend is chosen at random for each TL;DR. Then for TL;DR's in the same group, a prepend was first attached, and the TL;DR's were then concatenated into the same text, delimited by a line space.

Secondly, another dataset was created by first generating 1109 prompts using Claude Sonnet 4.5 [17] to some knowledge one might want to gain from watching a video. Examples of these include "how to bake a pie", "why is carbon monoxide deadly", and "what is the sunk cost fallacy". Then using YouTube's API [18] together with a YouTube transcript API [19]. These prompts were used to get the top-four videos from YouTube and then extract their transcripts. Next these were put into a database containing the prompt and video transcript pairs. Now, a single sample is constructed by grouping together these four videos with the same YouTube prompt into one single text in the same format as done with the TL;DR, except that they also have the YouTube prompt prepended. This means samples will contain the YouTube prompt, and then for each video, the video number followed by the title will be prepended on the transcript. Using these samples, outputs were generated by using GPT 5.1 chat [20]. The instructions for the outputs were to create an output containing a separate summary for each transcript that should cross-reference how they differ or agree with each other, and should focus on answering the prompt using this information. The full prompt can be found in the GitHub repository of the project (Appendix). This will be our gold standard example of how the videos should be produced. Because of resource constraints due to large input sizes, inputs of more than 10.000 tokens were filtered leading to 1004 samples. This was then split into a 90/10% respectively train/test split leading to 903 train samples and 101 test samples.

2.2 Model Fine-tuning Procedure

For the purposes of comparison, two different models were fine-tuned. One model was fine-tuned only on the second dataset (model 1), and another, both datasets (model 2). Both were based on Llama 3.2, 3B, 8-bit quantized [21]. All fine-tuning was done in the same way using the same hyper-parameters. For the purposes of staying within memory and time limits, LoRA fine-tuning was utilized with around 24M trainable 16-bit parameters. LoRA adapters were injected into all Attention and Forward Neural Network layers of the model (5) (6). The LoRA r parameter was set to 16, meaning the LoRA downscaling parameter matrix (A) will have dimensions [3072 x 16] and the upscaling matrix (B) will have size [16 x 3072], where 3072 is the model's original embedding dimensional size. The alpha scaling factor is set to 32, effectively scaling LoRA adapter weights by $\frac{32}{16} = 2.0$. The learning rate was set to 2e-4. The amount of epochs was set to one for model 2 when trained on the TL;DR dataset, and for both models, the amount of epochs was set to three for the custom "gold standard" dataset, as this is the most important one.

2.3 Model Testing

For testing of the solution, we compared three models in terms of their output on the test set. The first two models are explained above in section 2.2 (model 1 and model 2). The third model we test is simply the standard 8-bit quantized LLama 3.2, 3B, used as a control model (model 0). To make it fair, we gave the standard model the exact same instruction prepend that was given to GPT 5.1. We then for each model gave it the input transcript from the test set, which neither of the models had seen before, such that they generate the candidate results. Then we calculated the F1 BERTScore (9), by comparing the candidate results and the GPT 5.1 outputs as the reference. This gave us 101 F1 BERTScores for each model. This overall process can be seen in Figure Figure 1. The embedding model used in the BERTScore calculations was BART Large [22].

BERTScore is appropriate in our case, where there is no specifically correct answer. In comparison, more "naive" word statistic methods like Bleu- or Rouge-1-scores will neglect the similarity of two different words that have the same semantic meaning.

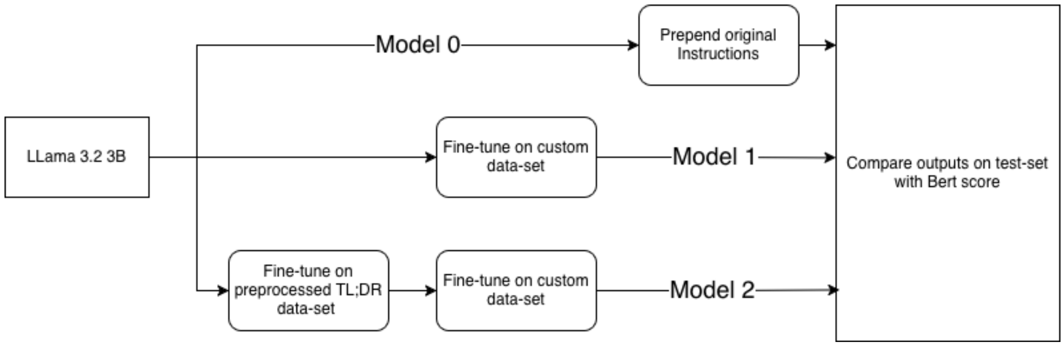


Figure 1: The testing workflow of the project

To compare the BERTScores of each, we conducted a Tukey's test [23] between the BERTScores of the three groups.

3 Results

The results of Tukey's statistical test between the BERTScores of the models are shown in Table Table 1 below:

Table 1: Results of the Tukey's statistical test

Comparison	statistic	p-value	Lower CI	Upper CI
(0 - 1)	$-.053$	$< .001$	$-.062$	$-.043$
(0 - 2)	$-.039$	$< .001$	$-.049$	$-.030$
(1 - 2)	.13	$p = .04$	.004	.023

Further, the BERTScore distributions of the models can be seen plotted in the boxplots in the Figure Figure 2 below:

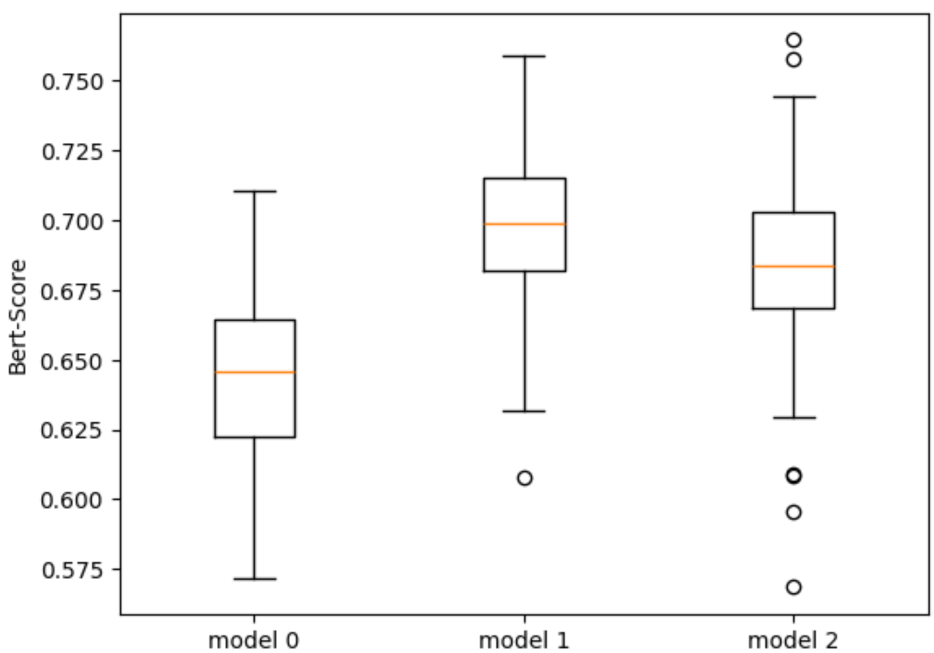


Figure 2: The testing workflow of the project

The results of the analysis show that, according to the BERTScore, both of the fine-tuned models (models 1 and 2) performed significantly better than the non-fine-tuned base model, with both results showing $p < .001$. However, looking at the Tukey statistic, we see that model 1, which was fine-tuned only on the custom dataset, performs slightly better than model 2, with $p = .04$ and effect .13. Inspecting the boxplots in Figure Figure 2 visually, taking outliers out of account, we can also confirm the findings that model 1 performs better across the full distribution.

4 Discussion

4.1 Summary and Discussion of the Results

From the results, it is clear that the model fine-tuned only on our custom dataset performs the best. This goes against the idea that using the TL;DR as an intermediate transfer learning dataset is useful. The idea was that since the custom dataset contains a relatively small amount of data, we could create a similar dataset from the TL;DR dataset. The TL;DR dataset has more data, so we could use this to “get the model started”. However, this seems to have only introduced more noise for the model, making it perform worse according to our results. We still see that it performs better than the reference model, but this is expected since it is trained specifically on samples drawn from the same dataset that we also tested the models on (although not the same samples). This drop in performance of model 2 relative to model 1 could be because the two datasets are simply too dissimilar for it to make sense. We expect that when trained on the reformatted TL;DR dataset that the model might have lost some of its more general intelligence that the model originally had from pretraining. This also seems to be in line with other results where overly fine-tuning language models actually made them perform worse as they lose their pretrained features [24]. In terms of the BERTScore values in the results, BERTScore does not generalize across tasks, and especially not across different embedding models, so we only have the models in this project to compare with for now.

However, the result that model 1 performed the best is an important result, since for the actual implementation of the full product pipeline, we now know that it will be better to use model 1. Additionally, when inspecting some sample outputs, it was also noticeable that the base model did not always create the expected format, whereas model 1 and model 2 were better at producing the correct output format in accordance with the instructions.

Our project can thus be said to be a success in accordance with our goal of creating a product that fetches multiple YouTube videos and summarizes and compares them with each other. Our project provides a lightweight and computationally efficient way of doing this, in comparison to running larger language models with larger sizes. Our model has around three billion parameters, which is a small number in comparison to GPT 5.1´s amount of parameters. Although not publically disclosed, more sources estimate that it contains more than one trillion parameters [25], [26] although, if it is based on architectures like Mixture of Experts [27], it could mean that less parameters are used for model inference. Thus, in comparison our model would save energy, time, and computational resources, and open up the opportunity of running the model locally.

4.2 Methodological Limitations

Setting our reference to be GPT 5.1 generations has its limitations. Although GPT 5.1 is a high-performing large language model as of this time, it is not perfect, and there is still room for improvement [28]. Furthermore, we cannot inspect all the summaries that it generated. Although a few sample outputs were inspected and were deemed to be well written, an expert human evaluation could have potentially improved the reference summaries.

Important to mention is also that BertScore has its limitations in terms of evaluating performance. Of course, since we are trying to generate a summary, there is no specific answer that is correct. This in the first place, makes it hard to measure the performance of the model using quantitative means. Qualitative methods, on the other hand, take time and effort to produce, especially given the 101 test examples that were made. BERTScore does not measure factual correctness, logical structure, coherence, redundancy of text or human preferences (directly). It can only measure the similarity of the contextual embeddings of each word with the most similar contextual embedding of a word in the wanted output. And the “wanted output” in our case has not been fully evaluated. Although, as written above, the few samples that were inspected worked well. In this way, human preference could theoretically lean towards any of the three models, although the prompt was designed to align with what we wanted. Thus, it has indirectly been aligned with human preference.

Since we used BART Large for the BERTScores, we also ran into a methodological problem. BART Large has a maximum context size of 1024 tokens. This means that if we generated an output of more than 1024 tokens that it would simply be truncated. While none of the GPT 5.1 outputs were longer than that. It was found that seven of the outputs from model 0 were larger than this, meaning they were automatically truncated by BART Large. However, these seven were mostly around 1100 tokens, meaning not many tokens were filtered out from these samples. This could potentially have the effect of lowering the BERTScore slightly in these samples in comparison to what they could have been, if some context in the GPT 5.1 output was missed from the filtered tokens. The same was seen with model 1 where 13 generations were truncated, with the maximum amount of tokens in a generation was 1054. Model 2, however, had 47 generations truncated, but the maximum amount of tokens in a generation was only 1075 (see appendix for distributions). This could potentially lead to an underestimation of its F1 BERTScores. However, since at max 51 tokens were filtered, this effect might not have been huge and would probably still not have made it perform as well as model 1. Additionally, the cost of doing this extra fine-tuning in terms of computational resources, energy, and time, is also a strong argument against doing it.

Lastly, sometimes YouTube transcripts do not contain a lot of talking. It could also be that the transcripts fetched are simply not relevant to answering the questions. Although the summaries that GPT 5.1 generated often took this into account, it is still not optimal. Especially if more videos corresponding to the same prompt do not contain relevant information, this could lead to a lower quality output.

4.3 Avenues for Further Studies

For further testing of our product, Comparison with larger language models on the specific task we are trying to solve could be relevant. We have argued that this performs better on the same task than similar sized standard models, but not that it performs better than general models with larger parameter numbers. This will be necessary to understand when our model is useful, and how useful it actually is.

Furthermore, we have not actually tested the full framework from start to end in one go. The next steps would be to make a full implementation, where a user sends in a YouTube query, it goes into the black box, and a summary of the top four videos is returned. This framework would have to be further optimized for efficient run times and usage, as well as being tested for this. Also, human testing would make preferential analysis much easier, but this is often costly to do.

Lastly, the project could be optimized by addressing the problem where transcripts do not contain relevant information. A mechanism could be made to decide if transcripts are bad, and if they are, other transcripts could potentially be fetched instead. Another way to address this is to make a more intelligent searching mechanism that takes a look at the immediate available information for the video, in order to decide if it is actually relevant for answering the question or not, in contrast to simply taking the top four videos ordered by YouTube.

5 Conclusion

In summary, we have tested three models having the purpose of generating a summary of multiple YouTube video transcripts, fetched from a specific search query. One model was a standard language model. Another model was trained on a custom dataset created from YouTube transcripts that was then summarized by GPT 5.1 as the target. The last model was trained using transfer learning from first being trained on a similar but more abundant dataset before being fine-tuned on the custom dataset. The test was conducted by comparing BERTScores between the models' output and a specifically instructed generation from GPT 5.1, given the same transcripts. Our analysis showed that the best performing model was the model that was only fine-tuned on the custom dataset. The model showed promising results in terms of generating a summary of the videos, taking into account factors such as referencing when video transcripts agreed and disagreed.

Bibliography

[1] R. Srihari and W. Li, “Information Extraction Supported Question Answering,” Oct. 1999, Accessed: Dec. 19, 2025. [Online]. Available: https://apps.dtic.mil/sti/html/tr/ADA460042/

[2] D. Moldovan and M. Surdeanu, “On the Role of Information Retrieval and Information Extraction in Question Answering Systems,” *Information Extraction in the Web Era*, vol. 2700. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 129–147, 2003. doi: 10.1007/978-3-540-45092-4_6.

[3] O. Kolomiyets and M.-F. Moens, “A survey on question answering technology from an information retrieval perspective,” *Information Sciences*, vol. 181, no. 24, pp. 5412–5434, Dec. 2011, doi: 10.1016/j.ins.2011.07.047.

[4] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] K. Kusunose, “Revolution of echocardiographic reporting: the new era of artificial intelligence and natural language processing,” *Journal of Echocardiography*, vol. 21, no. 3, pp. 99–104, Sept. 2023, doi: 10.1007/s12574-023-00611-1.

[6] M. Völske, M. Potthast, S. Syed, and B. Stein, “Tl; dr: Mining reddit to learn automatic summarization,” in *Proceedings of the Workshop on New Frontiers in Summarization*, 2017, pp. 59–63.

[7] S. Narayan, S. B. Cohen, and M. Lapata, “Ranking sentences for extractive summarization with reinforcement learning,” *arXiv preprint arXiv:1802.08636*, 2018.

[8] D. R. Radev, H. Jing, M. Styś, and D. Tam, “Centroid-based summarization of multiple documents,” *Information Processing & Management*, vol. 40, no. 6, pp. 919–938, 2004.

[9] E. Porter and T. J. Wood, “The global effectiveness of fact-checking: Evidence from simultaneous experiments in Argentina, Nigeria, South Africa, and the United Kingdom,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 37, p. e2104235118, 2021, doi: 10.1073/pnas.2104235118.

[10] M. Zimmermann and R. Jucks, “Investigating the Role of Communication for Information Seekers' Trust-Related Evaluations of Health Videos on the Web,” *Interactive Journal of Medical Research*, vol. 7, no. 2, p. e10282, 2018, doi: 10.2196/10282.

[11] K. Kousha, M. Thelwall, and M. Abdoli, “The role of online videos in research communication: A content analysis of Y ou T ube videos cited in academic publications,” *Journal of the American Society for information Science and Technology*, vol. 63, no. 9, pp. 1710–1727, 2012.

[12] D. Jurafsky and J. H. Martin, “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.”

[13] E. M. Hughes, R. Wang, P. Juneja, T. W. Li, T. Mitra, and A. X. Zhang, “Viblio: Introducing Credibility Signals and Citations to Video-Sharing Platforms,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, in CHI ’24. ACM, May 2024, pp. 1–20. doi: 10.1145/3613904.3642490.

[14] Y. Xu and M. Lapata, “Generating Query Focused Summaries from Web Documents,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. [Online]. Available: https://aclanthology.org/2021.acl-long.475/

[15] “Contrastive text summarization: a survey,” *Complex & Intelligent Systems*, 2023, [Online]. Available: https://link.springer.com/article/10.1007/s41060-023-00434-4

[16] V. Cachola, K. Lo, A. Cohan, and D. S. Weld, "TL;DR: Extreme Summarization of Scientific Documents," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[17] Anthropic, "Claude 4.5 Sonnet." 2025.

[18] Google LLC, "YouTube Data API (v3)." 2024.

[19] "yt-dlp/yt-dlp." Accessed: Dec. 19, 2025. [Online]. Available: https://github.com/yt-dlp/yt-dlp

[20] OpenAI, "ChatGPT (GPT-5.1)." 2025.

[21] Meta AI, "Llama 3.2: Open Foundation and Fine-Tuned Chat Models," technical report, 2024. [Online]. Available: https://huggingface.co/meta-llama/Llama-3.2-3B

[22] M. Lewis *et al.*, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703/

[23] H. J. Keselman and J. C. Rogan, "The Tukey multiple comparison test: 1953–1976," *Psychological Bulletin*, vol. 84, no. 5, pp. 1050–1056, 1977, doi: 10.1037/0033-2909.84.5.1050.

[24] A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang, "Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution." Accessed: Dec. 19, 2025. [Online]. Available: http://arxiv.org/abs/2202.10054

[25] "How Many Parameters does GPT-5 have - CometAPI - All AI Models in One API." Accessed: Dec. 19, 2025. [Online]. Available: https://www.cometapi.com/en/how-many-parameters-does-gpt-5-have/

[26] F. U. H. Zeya, "Comparing GPT-5.1, Gemini, and LLaMA 3: A Technical Deep Dive." Accessed: Dec. 19, 2025. [Online]. Available: https://medium.com/@faizulhaquezeya/comparing-gpt-5-1-gemini-and-llama-3-a-technical-deep-dive-0e58a5cd7c20

[27] Y. Zhou *et al.*, "Mixture-of-Experts with Expert Choice Routing," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7103–7114, Dec. 2022, Accessed: Dec. 19, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html

[28] M. D. McPhetridge, "GPT-5.1: A Structural Account of System Constraints." Accessed: Dec. 19, 2025. [Online]. Available: https://philpapers.org/rec/MCPGAS

6 Appendix

6.1 GitHub repository

https://github.com/au-nlp/project-milestone-p2-group-6

6.2 Equations

TF-IDF (Term Frequency - Inverse Document Frequency):

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{1}$$

where

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d}} f_{t',d} \tag{2}$$

$$\text{IDF}(t) = \log\left(\frac{N}{1 + \text{DF}(t)}\right) \tag{3}$$

and $f_{t,d}$ is the frequency of term t in document d, N is the total number of documents in the corpus, and $\text{DF}(t)$ is the document frequency (the number of documents containing term t at least once).

Cosine Similarity of two vectors:

$$\cos(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \; \|\boldsymbol{v}\|} \tag{4}$$

Normal linear layer

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \tag{5}$$

LoRA adapted linear layer

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \frac{\alpha}{r}\boldsymbol{B}\boldsymbol{A}\boldsymbol{x} \tag{6}$$

Where the LoRA adaptation is scaled by α/r. In LoRA training, we let the weights of the original $\boldsymbol{W}$ matrix stay as is, and only train the downscaling Matrix $\boldsymbol{A}$ and upscaling matrix $\boldsymbol{B}$ by backpropagation and gradient descent as usual. $\boldsymbol{x}$ is the input vector to be projected by the weights, e.g. embeddings.

BERTScore Precision and Recall

$$\text{Precision} = \frac{1}{|\boldsymbol{C}|} \sum_{\boldsymbol{c} \in \boldsymbol{C}} \max_{\boldsymbol{r} \in \boldsymbol{R}} \cos(\boldsymbol{c}, \boldsymbol{r}) \tag{7}$$

$$\text{Recall} = \frac{1}{|\boldsymbol{R}|} \sum_{\boldsymbol{r} \in \boldsymbol{R}} \max_{\boldsymbol{c} \in \boldsymbol{C}} \cos(\boldsymbol{r}, \boldsymbol{c}) \tag{8}$$

Where $\boldsymbol{C}$ is the matrix containing candidate contextual embeddings (The generations) and $\boldsymbol{R}$ is the matrix containing the reference contextual embeddings (The ground truth)

The F1 score is then harmonic mean of the precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

6.3 Figures

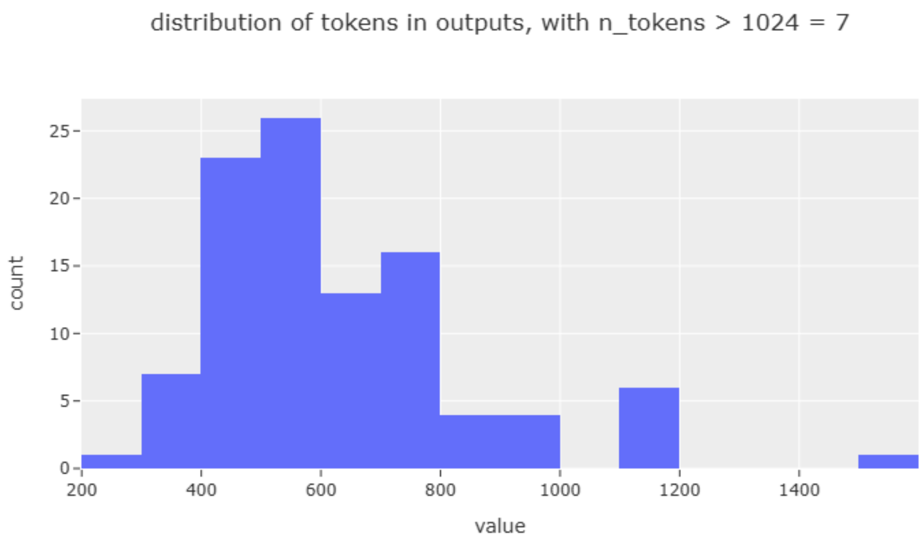


Figure 3: Output Tokens Length from Base Model (LLAMA 3.2 3B)

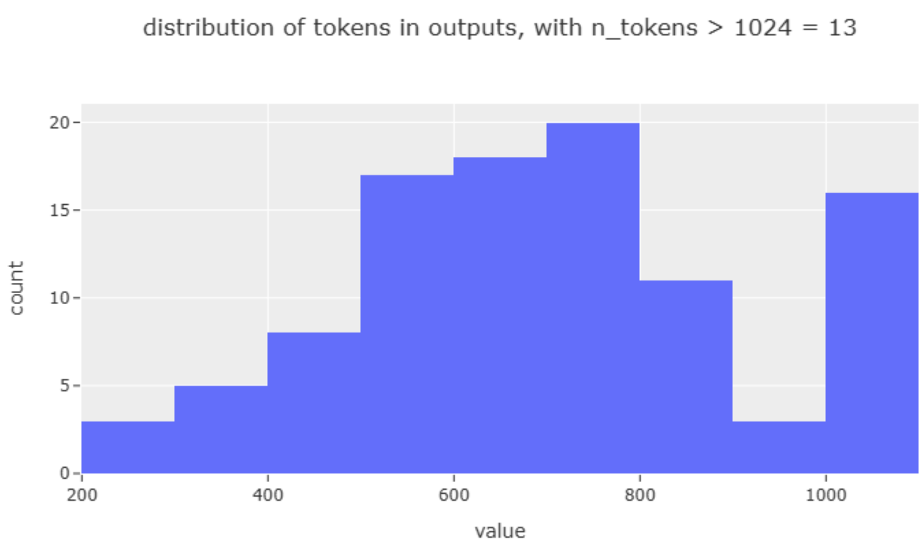


Figure 4: Output Tokens Length from Fine-Tuned Model (LLAMA 3.2 3B with Custom Dataset)

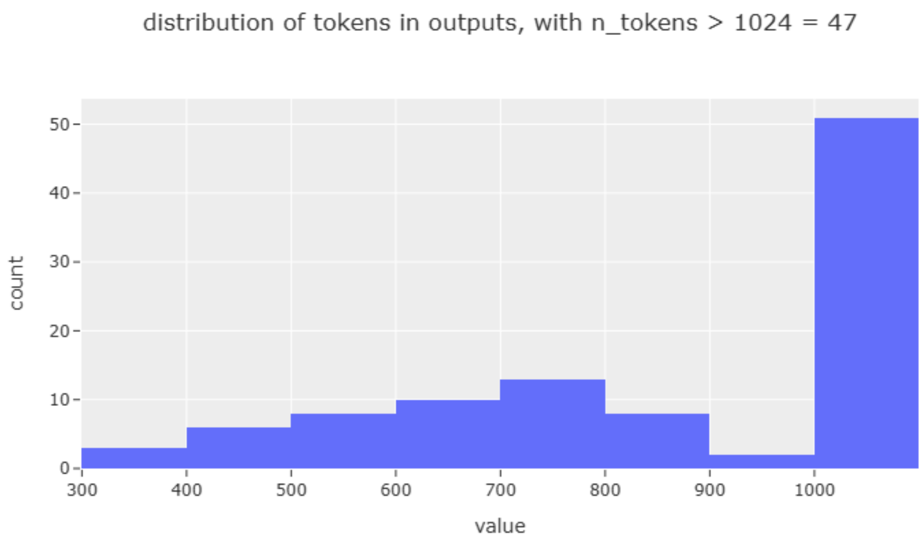


Figure 5: Output Tokens Length from Fine-Tuned Model (LLAMA 3.2 3B with TL;DR and Custom Dataset)