Replication of YouTube's User-Generated Content Summarization and Topic Extraction Model

Tommy Shang Wu, Jake McFarland, Abiel Jeong-Joon Kim

Abstract

In 1956, George A. Miller proposed his notion of "The Magical Number" regarding an exploration of human memory and its limitations. His paper, titled "The Magical Number Seven, Plus or Minus Two", posits that the human (short term) memory is only able to retain roughly 5 to 9 "chunks" of information at a time. Fast forward to the 21st century, large scale digital platforms offer information at unprecedented volumes, leading to cognitive and information processing problems such as information overload. Our paper explores the notion of information overload with respect to the YouTube platform, and more specifically, deep learning methods and NLP techniques that solve information overload with respect to summarizing and compressing general viewership of a YouTube video comment section. By accessing the comments on any YouTube video, an AI comment section summary can be used to determine what users are talking about in a video. i.e. By taking in large swaths of natural language text, our model attempts to output abstractive summaries of a number of these topics. The model is intended for use on native YouTube comments and is built to replicate the YouTube Topic AI model.

1 Introduction

As the digital age progresses forwards, the amount of information that is available to society is increasing exponentially. Due to population growth, increased ease of access to the internet, and the continued cultural shift towards electronic commerce and electronic communication, social media platforms are being brought to the forefront of communication. While some of these communications are directed towards another, a large amount of communication is intended for the general populace to peruse. Examples of this include tweets, Facebook posts and comments on YouTube videos, as opposed to directly messaging someone.

Due to the incomprehensible amount of content that is generated on a day-to-day basis, it is impossible for one to watch every video or read through every post on social media. As evidence, YouTube alone receives 500+ new hours of video uploads every minute (YouTube Official Blog). Using summarizer models, which take in large swaths of data and output more easily digestible sentences or phrases, what was previously thought of as unapproachable is now significantly easier to comprehend.

Taking inspiration from YouTube's recent feature currently in beta testing, YouTube Topics, we have attempted a replication of the comment summarization system currently being worked on. By taking the contents of the comment section of any YouTube video and cleansing the data to remove usernames, the transformer-based model we have created first performs k-means clustering in order to determine the number of topics that it should be summarizing, before summarizing each individual cluster in an easy-to-understand sentence or two.

In this manner, by taking the closest n comments of each cluster of YouTube comments and appending them together, our model is intended to summarize the topic behind each of the clusters and output a one-line summary of the discussion happening.

We tested several methods for both the clustering of k-means and the summarizer model. For the k-means clustering, we determined the number of clusters in three ways; the elbow method, the silhouette method, and by matching to the number of topics on a YouTube video in question. With regard to our model, we changed our base model between Google's T5 base, Google's T5 small, and Facebook's BART base.

2 Prior Related Work

Despite the relative novelty of the YouTube Topics AI, it becomes increasingly clear that summarization is a large part of the increase in demand for ar-

tificial intelligence. The first paper published with regard to automatic summarization was in 1957 by Hans Peter Luhn (Luhn, 1957). In his paper, *A Statistical Approach to Mechanized Encoding and Searching of Literary Information*, he discussed how automatic summarization might be achieved through statistical analysis of text documents. By recording the context and concordance of each concept, he theorized that a paper could be summarized by separating these concepts into notional families. Each of the notional families present in a paper would show what the paper would be about.

Luhn's paper shows an early form of the concept of contextual embeddings. Every word would be grouped under a larger family, and the size of the family relative to the document would describe the document. Current similar research describes word embeddings as a d-dimensional space, where words that have similar meanings are placed close to one another. Typically, these word embeddings have a dimensionality of 50-300, whereas Luhn's paper suggested approximately 1000 families for words to fall under.

As the internet grew past the 50s, it became clear that longer content was losing it's place in some communities. The culture on several online platforms shifted towards shorter messages, moving from lengthy blog posts to the short form content of the current era. When lengthier content was posted on the popular social media sit Reddit, users would often sum up their posts in a few sentences at the bottom of the page, with the acronym TLDR (too long didn't read). When users did not give their own summary of long explanations or stories, other users created bots to do so for them. The TLDR bot, u/autotldr, summarizes external links from other users when it was called (u/AutoTLDR).

Further research on the idea of concentrating data into distinct sets remained a popular topic of discussion. In the paper *Review on determining number of Cluster in K-Means Clustering* (Kodinariya and Makwana, 2013), many ways to determine the number of distinct sets that should be presented in a dataset are presented, two of which were used in determining this model's clustering method.

This idea of summarizing papers stagnated in the 50s, but became popular again in recent times with the rise of artificial intelligence. With the increasing amount of research being done into artificial intelligence, summarizers improved greatly. Products that use summarization as a service are becoming increasingly more common, present both within some social media platforms and as a purchase from a third party service. Scholarcy, a summarizing AI intended for researchers and students, is used to create digestible versions of otherwise lengthy reports and textbooks (Scholarcy). By decreasing the size of these documents, it is possible to reduce the amount of time needed to study previous examples and new material. This save of time can then be used to further research on things beyond the scope of the document being read.

Other related works have built on top of the summarization of text and moved into summarizing multiple aspects of internet traffic at the same time. For example, Recall, another summarizing tool, claims to create a 'knowledge graph' that takes higher level concepts from summarized content and saves them to a database, then makes connections on content you have watched or read. It then helps you recall these topics using active recall strategies (Recall Documentation). Recall can use information from news articles, PDFs, YouTube videos, podcasts, and various other online platforms. As such, Recall is similar to the YouTube Topics AI in that it must bridge connections between multiple pieces of content instead of one cohesive story.

3 Approach

The high-level approach and system pipeline remained consistent since the project proposal with heightened details with respect to low-level implementation decisions. To reiterate, the ultimate project objective is to engineer an NLP system that takes in a swath of user-generated comments and outputs a summary of key overlapping topics as a means of communicating general viewership thought or sentiment.

Our proposed system pipeline comprises two primary systems which abstractly factor the project directive into two more tractable tasks. Namely, the identification of key overlapping topics via K-means clustering followed by the summarization of each key topic via a deep-learning NLP model. The following sections elaborate on the implementation-specific decisions that we made to bring this system to life.

3.1 Topic Extraction Subsystem

3.1.1 Sentence-Embedding Projection Method

The core idea is to implement take the set of user comments, project them into the embedding space,

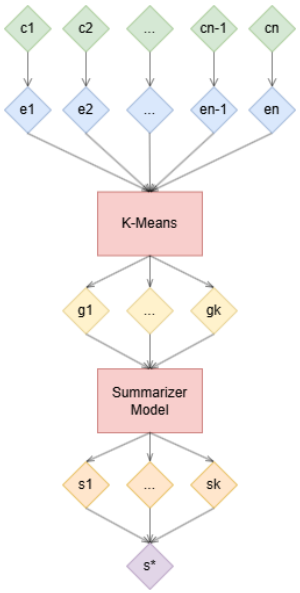

Figure 1: High Level System End to End Pipeline

and then run K-means clustering across the embedding vectors to identify key overlapping topics to which can then be iteratively fed into the summarization model.

We implemented the mapping of sentence embeddings directly via a popular pretrained model from Hugging Face found here: (**?**). As per their description, the transformer-based model, all-mpnet-base-v2, is specifically designed for sentences and paragraphs revolving around tasks like clustering or semantic search which matches the needs of our topic extraction subsystem perfectly. Note that the sentence embeddings have 768 dimensions as opposed to GloVe or Word2Vec's typical 300.

3.1.2 K-Means Clustering: Finding Optimal K

Once each comment has been mapped to a 768-dimensional space, we implemented K-means over the sentence-embedding space as a means to find key overlapping topics with respect to general viewer sentiment or thought. However, in order to determine the number of clusters, K, that our system should identify, we implemented 3 main approaches: Predefined K, search K via elbow method, search K via silhouette score.

The first approach, using a predefined K, is the simplest approach and least computationally costly. We simply manually set the number of clusters K that our system should interpret before running SKLearn's KMeans algorithm over the set of sentence embeddings.

$$K_{opt} = C$$

The second approach searches for an optimal K value via the elbow method. It works by defining a range of K values to try, and picking the optimal K which minimizes the sum of squared distance from a centroid to its member sentence embeddings, called WCSS (Within Cluster Sum of Squares). This approach discovers a more suitable K than the predefined-K approach at the cost of greater computational cost.

$$\text{WCSS} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

The third approach searches for an optimal K value via leveraging silhouette scores. It works by defining a range of K values to try, and picking the optimal K which maximizes the average silhouette score, which is a global measure of how cohesive clusters are and how well they are separated. The average silhouette score requires computing the silhouette score for all embedding vectors which is calculated by considering how cohesive it is in the context of its associated cluster (average distance to every other vector in the same cluster) and how well separated it is from unassociated clusters (average distance to every other vector in complementary clusters). i.e. Maximizing the average silhouette score sees that every point is highly cohesive (close to every related member point from the same cluster) and well separated (far from every other unrelated point from different clusters).

$$S_{avg} = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

3.2 Generative Summarization Model

3.2.1 Brief Survey of Baseline Models

Once the K-Means subsystem has clustered relevant embeddings $\{e_1, e_2, ..., e_n\}$ together to produce clusters $\{g_1, g_2, ..., g_k\}$, we must now construct a generative model that takes in these k clusters as input and outputs k summaries for each comment cluster. The ideal summary is a succinct one-line sentence or phrase that succinctly summarizes the general topic of a cluster g_i for $i = 1...k$.

In order to achieve this, we will select 3 baseline models and fine-tune them with our chosen HuggingFace dataset. In section 5, we will elaborate more on the quantitative results and com-

parisons between the models. Our proposition is that summarization requires a baseline model that can abstract semantically meaningful ideas from large swaths of text. Therefore we select candidate models predicated on their ability to perform abstractions.

Table 1: BART Architecture

Description	BART (base)
Model architecture	Bidirectional Encoder-Decoder
Pretraining strategy	Token masking, token deletion, text-infilling, sentence permutation
Total parameters	∼140M
Encoder layers	6
Decoder layers	6
Hidden size	768
Feedforward (FFN) size	3072
Attention heads	12
Pretraining corpus	BOOKCORPUS, WIKIPEDIA, CC-NEWS, STORIES

The first model that we chose is the BART model which is a bidirectional encoder, originally pretrained to reconstruct corrupted or missing text (Lewis et al., 2020). Intuitively, we chose BART for its ability to robustly handle variations and unpredictability in input and generate semantically meaningful content. The key idea is that BART was pretrained on noisy input, forcing the model to learn to abstract meaning from verbosity which gives us a solid launchpad to work from in the fine-tuning stage.

The second and third models that we chose are T5-base and T5-small. The intuition for using these T5 variants follows a similar train of thought for our reasoning of BART insofar that both T5-base and T5-small were originally pretrained to reconstruct text from span-corrupted, noisy inputs, which aligns with our proposition that the model may perform abstractions of noisy syntax to learn and generate semantic meaning (Raffel et al., 2020).

3.2.2 Model Training Environment

Each of the 3 baseline models were fine-tuned on our chosen HuggingFace dataset with the same hyper parametric configuration as outlined in figure 2. The training-validation split was 80% to 20%

Table 2: T5-Base Architecture

Description	T5-base
Model architecture	Encoder-Decoder
Pretraining strategy	Span corruption
Total parameters	∼220M
Encoder layers	12
Decoder layers	12
Hidden size	768
Feedforward (FFN) size	3072
Attention heads	12
Pretraining corpus	COLOSSAL CLEAN CRAWLED CORPUS

Table 3: T5-Small Architecture

Description	T5-small
Model architecture	Encoder-Decoder
Pretraining strategy	Span corruption
Total parameters	∼60M
Encoder layers	6
Decoder layers	6
Hidden size	512
Feedforward (FFN) size	2048
Attention heads	6
Pretraining corpus	COLOSSAL CLEAN CRAWLED CORPUS

for all fine-tunings training cycles. Note that the $predict\_with\_generate$ flag in the HuggingFace library configures the models to generate output sequences during evaluation which enables us to compute ROUGE scores.

Table 4: Training Hyperparameters

Hyperparameter	Value
Number of Epochs	6
Learning Rate	`2e-5`
Train Batch Size (per device)	`16`
Eval Batch Size (per device)	`16`
Weight Decay	`0.01`
Predict With Generate	`True`

A single total training cycle for the T5-Small took ~30 minutes whereas the T5-Base and BART models took ~1 hour and ~35 minutes respectively

while remotely connected to CSIL's Intel Device a780 and NVIDIA Corporation Device 2684 GPUs. During training, roughly 6GB of VRAM out of 24GB were used per training cycle across all 3 models.

4 Data

In the making of this model, it was necessary for there two be two different types of data presented to the model, one for training and one for testing. Although the models designed are built off of existing models, the fine-tuning of these models was still a requirement. To begin this fine-tuning, a dataset with summaries attached to each piece of data was required. The summary could then be provided as feedback for the models during training. Once the models had been trained, the model required the YouTube comments input as another source of data. The model could then be tested in the scenario it was designed for.

For the fine-tuning dataset, our dataset of choice was *Context Based Chat Summary Plus*, by Prithiv Sakthi and hosted on HuggingFace (Prithiv Sakthi (prithivMLmods)). The dataset is a collection of newspaper articles, organized by their headlines and bodies. Using this dataset, the model was fine-tuned using the headline as a summary of the information enclosed in the bodies.

For the real-world data, it was imperative for the model to be tested on a wide variety of comment types. In this manner, videos with intentionally spread demographics were selected. Due to the nature of the YouTube Topics AI still being in development, this feature was not available on every video, so only comments on videos that were selected for this feature were able to be tested.

YouTube's Topics AI, while not explicitly stated, appears to avoid videos with polarizing views, depictions of violence, and sexual content. The first choice for a model trained on the newspaper dataset would be the comment section of a video that would be similar in nature; a comment section on a news report uploaded to YouTube would be optimal. However, YouTube's Topics AI appears to avoid these comment sections to a great degree. Additionally, many news channels turn off comments on their videos to avoid brigading and counter-narratives.

There were two videos that ended up being selected for testing purposes. The first video selected was an informational video by Stand-Up Maths titled *The 10,000 Domino Computer* (Stand-up Maths). This dataset represented a slightly more scholarly video on YouTube, with the hope that it would align similarly to the testing data. The second dataset selected was a segment from a comedy game show hosted by Dropout, with the video titled *Sam Tortures Brennan With a Bird Trivia Contest He Cannot Win* (Dropout). This dataset was selected to be the contrast to the previous dataset, where the summarizer would be run on words it had likely never heard before and inside jokes about the Dropout show.

5 Experiments

5.1 Implementation

5.1.1 Topic Extraction Implementation

To obtain the comments from any YouTube video, the package yt-dlp was used. In order to determine the optimal k for k-means clustering using the elbow method, the KneeLocator method from Python's kneed module was used (Satopaa et al., 2011). Additionally, the k-means embeddings calculations were performed using Python's scikit-learn module, as well as the calculation for the silhouette score (Pedregosa et al., 2011).

5.1.2 Generative Model Implementation

In order to achieve proper summarization, the *Summarization* guide on HuggingFace was used (Hugging Face, 2024). This guide was followed for the T5-small model, and was modified to use a T5-base model as well as a BART model, including the use of the Sentence Transformers module (Reimers and Gurevych, 2019).

5.2 Results and Evaluation

The primary evaluation metric we use for comparison purposes are 4 variants of the ROUGE score (Recall-Oriented Understudy for Gisting Evaluation). ROUGE scores compare structural similarities between our model's generated summarizations and the dataset's reference summaries via computing n-gram overlaps using precision, recall, and F1 scores.

More concretely we used: ROUGE-1 which measures the overlap of unigrams and ROUGE-2 which measures the overlap of bigrams. We also used ROUGE-L and ROUGE-LSUM which compute measures of n-gram overlap of the longest common subsequences between texts. The key difference

is that ROUGE-LSUM considers an entire document as a single LCS whereas ROUGE-L computes the average n-gram overlap of the LCS for each pairwise sentence. i.e. Split the document into sentences based on newlines, then compare sentences pairwise.

5.2.1 Topic Extraction: K-Means Evaluation and Methods Comparisons

In order to determine the most suitable approach to determining or approximating the optimal number of clusters K in our K-means clustering subsystem, we first compared the reported optimal k values between the K-means approaches over 984 samples from our dataset. That is, compute the reported optimal k values from leveraging the elbow method vs computing average silhouette scores and analyze their behaviors for varying ranges of candidate K values.

In order to reason about what approach we should take in order to find the optimal number of clusters, K, in our embedding space, we decided to investigate and answer the following 3 questions: Do the approaches yield a similar K value?

If both approaches converge to similar K values, then we could leverage this as evidence on deciding upon the method to find the optimal K dynamically during inference, say by leveraging a hybrid approach of the two methods.

5.2.2 Elbow Method Analysis

The elbow method converges to an optimal K of 10 as we test a greater number of candidate clusters over the 984 samples. Conversely, the elbow method decreases its optimal K count as the number of tested candidate clusters decrease (we lose the optimal K of 10 at around $K_{max} = 50$).

This finding suggests that, across the 984 sample space, the elbow method requires a certain upper bound of candidate Ks in order to identify a clear elbow of the WCSS vs K-Clusters plot. Otherwise, the distance curve becomes too flat and the inflection point becomes less apparent. It therefore follows that the elbow method is a stable and promising approach that yields the optimal K (likely around or equal to 10) but only when provided a sufficient K_{max}.

5.2.3 Silhouette Method Analysis

Typically, a well-clustered dataset should see the average silhouette score peak at an optimal K with its scores degrading in value for varying candidate Ks diverging from the optimal. Interestingly, in the context of our 984 sample, the silhouette score approach favors $K = 4$ clusters when testing smaller ranges of candidate Ks, $K_{max} < 50$ but diverges for large ranges of K, $K_{max} > 50$. In fact, the data tends to yield optimal K = $K_{max} - 1$ for large K_{max}.

This finding suggests that as we scale the upper bound threshold of our K range, the K-Means model overfits to the 984 sample with clusters reducing to singleton clusters where each embedding vector becomes its own centroid. In the limit, one can realize this by imagining the uniform distribution across all sentence embeddings in 768 dimensions. This aligns with the theoretical literature underpinning the behavior of the average silhouette score where a rough upper bound threshold is suggested at about:

$$K_{\text{max}} \approx \sqrt{\frac{N}{2}}$$

Where N represents the sample size.Consequentially, the silhouette score approach reveals that a clustering structure indeed emerges with an optimal K around 4 when testing comparatively lower upper bounds thresholds, but loses stability and over fits as we scale K_{max}.

5.2.4 K-Means Final Verdict

A standalone implementation of the elbow method is promising, but requires a sufficient K upper bound threshold in order to identify a clear inflection point and hence optimal K. That is, comparatively larger compute costs are required to recognize a clear convergent optimal K, otherwise the WCSS vs K-clusters curve becomes too flat - likely a function of the underlying dataset's complexity. *Thus, the standalone elbow method is eliminated.*

The standalone silhouette approach empirically suggests the requirement of a relatively smaller upper bound threshold using our aforementioned rule of thumb formula which is promising. A hybrid approach that combines the silhouette score and elbow methods are also promising i.e. consider taking the arithmetic average between both approaches.

Therefore, by process of elimination, we decided that the final K-means approach to be either a standalone implementation of the average silhouette score method or a hybrid combination of both the average silhouette score approach and elbow method. For the final project, we plan on defining

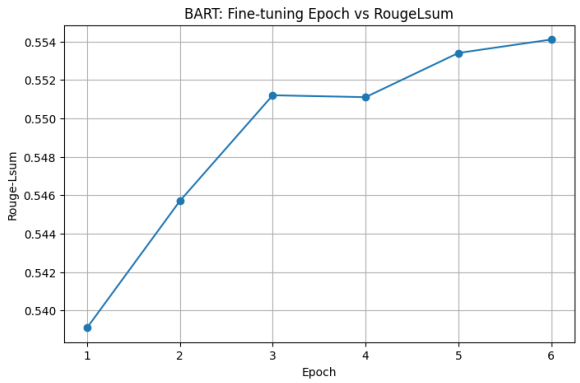

Figure 2: BART Plot of EPOCHS vs ROUGE-Lsum

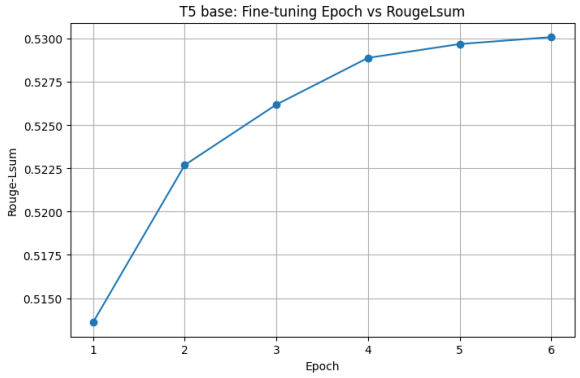

Figure 3: T5-Base Plot of EPOCHS vs ROUGE-Lsum

certain thresholds for sample size N that invokes either or as a piecewise function.

5.2.5 Generative Summarization Models: Evaluation and Comparisons

The key finding with respect to the ROUGE score evaluation comparisons is that the BART model outperforms both T5 variants across all ROUGE metrics and validation loss, with the T5-Base marginally outperforming the T5-Small. All models had similar generation lengths, with the BART model averaging between 16.1 to 16.3 tokens, the T5-Base producing around 15.8 to 16.0 tokens, and the T5-Small averaging the lowest summary text generation length at 15.7 to 15.9.

When measuring our models' fine-tuning metrics, every model was found to decrease in training and validation loss as the epochs progressed. With regards to the Rouge1 score, the T5-small model approached a value of 0.54, the BART model approached a value of 0.60, and the T5-base model approached a value of 0.57.

Despite these promising scores, all of the models underperformed when concentrated on real-world data. With output significantly longer than the intended length, and incomprehensible summaries, there was not a meaningful output produced. Some outputs were one of the comments on YouTube with the middle of the sentence taken out, while others were one comment stitched on top of another.

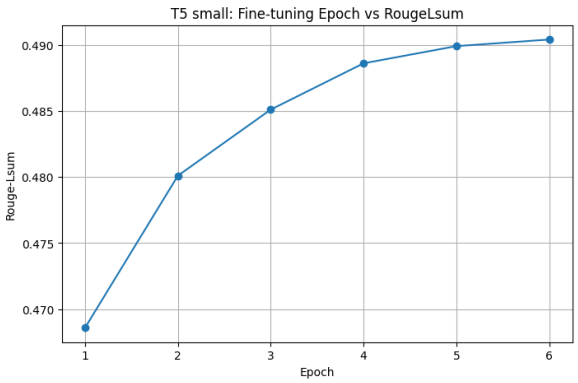

Figure 4: T5-Small Plot of EPOCHS vs ROUGE-Lsum

6 Analysis

Some changes were made during testing to the YouTube comments being input to the models. It was determined that many comments that were replies to other comments had no meaning, and were emojis only or otherwise not useful to a summarizer model. After noticing this, the data fed into the models had replies removed, which showed a marginal improvement in the summarization, with significantly less usernames present in the output. However, there were still a significant number of shortcomings in the summarized results.

Upon review, there are a number of reasons that accumulate to these failures. With regards to the selected training dataset, it does not fit well to the real-world implementation of our model. Our results indicate issues in the datasets, both with the selected training dataset and with the YouTube comments. The text of a newspaper article is significantly different from that of YouTube comments in three major ways.

Due to the nature of a newspaper article being a report on an event, and YouTube comments being a reaction to the media presented, the perceptive used is very different. In newspapers, it is common to refer to third person events, without using first person language. In YouTube comments, first person perspective is significantly more prevalent, with many users commenting on how they enjoyed a certain part of the video or their own impressions on it. Another discrepancy between the two is the

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	1.7995	1.5395	0.5111	0.2758	0.4686	0.4686	15.7735
2	1.6830	1.4672	0.5235	0.2861	0.4803	0.4801	15.8905
3	1.6300	1.4323	0.5286	0.2910	0.4853	0.4851	15.8390
4	1.5836	1.4133	0.5322	0.2945	0.4886	0.4886	15.8953
5	1.5743	1.4038	0.5336	0.2961	0.4900	0.4899	15.9254
6	1.5622	1.4004	0.5341	0.2965	0.4906	0.4904	15.9326

Table 5: Fine-tuning metrics for T5-small.

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	1.3562	1.1568	0.5823	0.3463	0.5393	0.5391	16.2936
2	1.1912	1.1097	0.5896	0.3540	0.5459	0.5457	16.0473
3	1.0620	1.0927	0.5951	0.3592	0.5514	0.5512	16.1174
4	0.9854	1.0851	0.5954	0.3605	0.5512	0.5511	16.1923
5	0.9064	1.0827	0.5975	0.3622	0.5536	0.5534	16.0260
6	0.8766	1.0848	0.5986	0.3635	0.5542	0.5541	16.1133

Table 6: Fine-tuning metrics for BART

formality of the language used. As internet culture has evolved, so has informal messages and slang. Words used in YouTube comments may never be seen in previously printed newspaper articles.

Finally, the presence of references to other users in YouTube comments occurs in a manner dissimilar to that of newspaper articles. In newspaper articles, named entities are very often accompanied by a description of their occupation or relation to the article. However, due to the anonymous nature of the internet, many users refer to one another without referencing their relation to the media at hand, based solely on their reaction to it. Additionally, references such as those along the lines of 'but can it run Doom' that were commonly seen in the *The 10,000 Domino Computer* video's comments expect the viewer to understand the relation of the named entity to the video without explicitly stating it. Our summarizer model was not trained on data of this nature, and fails to understand the meaning behind these comments.

7 Conclusion

Throughout the process of creating these models, the timeline proposed was followed without deviation. The multiple k-means methods used gave a satisfactory way to separate the input into multiple clusters. The fine-tuning of each model was done in a successful manner, with the models producing Rouge1 scores of over 0.50, suggesting a significant amount of success. As the epochs of each model increased, a decrease in the training loss and validation loss was recorded, as well as a raise in Rouge1 scores.

Where each model failed was in the dataset being used to generate this Rouge1 score. Due to the poorly selected dataset, the similarity between the training and the test datasets were minimal at best and resulted in outputs that had no sensible similarity to the task they were being evaluated on. The result of these findings suggests that with a more properly oriented dataset, the output will be significantly closer to that of the YouTube Topics AI.

References

Dropout. Sam tortures brennan with a bird trivia contest he cannot win. Accessed: April 10, 2025.

Hugging Face. 2024. Summarization. Accessed: 2025-03-20.

Trupti M. Kodinariya and Prashant R. Makwana. 2013. Review on determining number of cluster in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6):90–95. Accessed: April 10, 2025.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information.

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeLsum	Gen Len
1	1.2999	1.1388	0.5586	0.3200	0.5136	0.5136	15.8302
2	1.1905	1.0944	0.5682	0.3283	0.5226	0.5227	15.9561
3	1.1525	1.0733	0.5777	0.3316	0.5261	0.5262	15.9509
4	1.0914	1.0595	0.5746	0.3339	0.5288	0.5289	15.9509
5	1.0800	1.0571	0.5759	0.3355	0.5298	0.5297	16.0033
6	1.0634	1.0566	0.5761	0.3356	0.5300	0.5301	15.9739

Table 7: Fine-tuning metrics for T5-base

IBM Journal of Research and Development, 1(4):309–317.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Prithiv Sakthi (prithivMLmods). Context-based-chat-summary-plus. Accessed: 2025-04-10.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21.

Recall Documentation. Docs. Accessed: 2025-04-10.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171.

Scholarcy. About-us. Accessed: 2025-03-20.

Stand-up Maths. The 10,000 domino computer. Accessed: April 10, 2025.

u/AutoTLDR. Faq: Autotldr bot. Accessed: 2025-04-10.

YouTube Official Blog. Youtube for press. Accessed: 2025-04-10.