Text Summarization with T5

Nate Omdalen
Section: CSCI 6502-001B, ID: 110026359
nathan.omdalen@colorado.edu

ABSTRACT

In this paper, I present my project for CSCI 6502, Big Data Analytics. A machine text summarization system takes as input a piece of text and produces as output a summary of the text. Such systems have many applications, most notably in search engines. In my project, I use the deep neural network architecture T5 to investigate text summarization. I train the network on the WikiHow dataset. I explore and compare three fine-tuning techniques. In the first, all the parameters of the model are free to be tuned during the entire training process. In the second, gradual unfreezing, blocks of the encoder and decoder of the network are unfrozen one at a time throughout training. In the third, adapter tuning, special fully-connected modules are inserted into the network, and only these layers are free to be tuned. I also investigate the extent to which T5 still generalizes to other data after being fine-tuned by testing it on other datasets.

KEYWORDS

text summarization, neural networks

ACM Reference Format:
Nate Omdalen. 2022. Text Summarization with T5.

1 INTRODUCTION

A text summarization system takes a piece of text as input and outputs a summary of the text. There are many applications for text summarization. Most notably, these systems have become important for search engines such as Google. When someone enters a query into a search engine, it is now common for the engine to display a short summary of the most relevant webpage. Another possible application is social media analysis. Perhaps, someone has a large dataset consisting of social media posts, and they would like to have a system that summarizes what is being talked about in the dataset. Text summarization may also be useful in academic research. These days, there are far too many research articles written in any given domain, making it difficult for a researcher to keep informed about current developments. Text summarization systems can alleviate this concern by providing summaries of one or more articles that the researcher can use to better curate the literature. Given these and many other applications, the development of effective and efficient text summarization systems is important.

There are two different types of text summarization systems: extractive and abstractive. An extractive summary finds the most relevant sentences of a text and returns them as the summary of the text. An everyday, human example of extractive summarization is highlighting or underlining the passages of a text that best capture its meaning. An abstractive summary generates new text that captures the meaning of the text. Some human examples of this might be writing a plot synopsis of a story or writing a review of an academic paper. Abstractive summarization is often considered to be more difficult than extractive summarization. This is because abstractive summarization necessitates the production of original text which requires a greater understanding of the language than simply identifying passages in a text.

In this paper, I propose the use of the T5 deep neural network architecture for abstractive text summarization. T5 stands for Text-to-Text Transfer Transformer. As the name suggests, this architecture is a transformer-type model. It stands apart from other deep language models in that the procedure used to pre-train T5 is specially geared towards downstream natural language tasks such as text summarization. The dataset that I use is the WikiHow dataset. This dataset, scraped from the popular how-to website, consists of short tutorials on various topics paired with brief summaries of the tutorials. Using the WikiHow dataset and the T5 network I compare three kinds of fine-tuning techniques. The first tunes all the parameters in the network throughout all of the training, while the second starts by tuning only the higher layers and slowly unfreezes the remaining layers and the third only tunes certain adapter layers inserted into the network. I compare the performance of these methods on a held-out testing set using the text summarization metric ROUGE. I also test which method best maintains its generality by evaluating each method on a different dataset, the CNN news article dataset.

The rest of this paper proceeds as follows. Section 2 explores the details of the T5 Network. Section 3 introduces the WikiHow dataset. Section 4 details the ROUGE score. Section 5 discusses the three fine-tuning techniques that I utilize to train the T5 network. Section 6 analyzes the fine-tuning behavior of each method on the WikiHow training set. Section 7 details the results of each fine-tuning on the WikiHow test set. Section 8 looks at some example summaries generated by the models on the WikiHow dataset. Section 9 explores the generalizability of each fine-tuning technique by testing them on the CNN news article dataset. Section 10 provides some example summaries produced by the models using the CNN dataset, and Section 11 provides some concluding remarks.

2 THE T5 NETWORK

In this project, I utilize the small T5 deep neural network architecture, first developed in [5]. The architecture follows a typical encoder-decoder transformer language model. Figure 1 illustrates the general architecture of a transformer. The green blocks constitute the encoder, and the red blocks constitute the decoder. In what follows next, I provide a brief description of each component of Figure 1.

The bottom, or beginning, of the network starts at the bottom left corner. There are two words, "thinking" and "machines," that are being pushed through the network. The first component of almost any language model is a word embedding. A word embedding takes

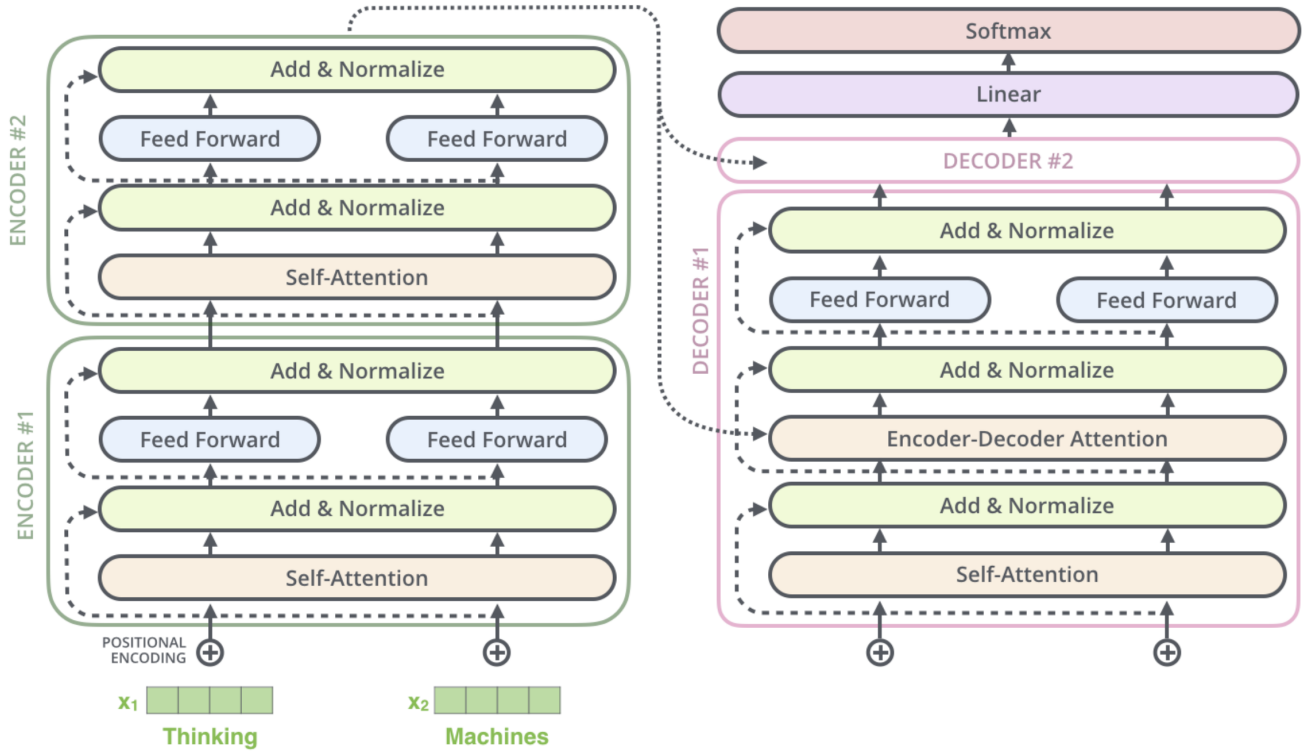

Figure 1: General Transformer Architecture - Source: http://jalammar.github.io/illustrated-transformer/

each individual word of a piece of text and converts it to a vector that encapsulates its meaning. The rectangles above each word in the figure represent this vector. The word embedding can be learned from scratch along with the rest of the network. This is especially effective when a large dataset is used. However, it is sometimes the case that the weights of the word embedding are trained separately using methods like the skip-gram model.

After going through the word embedding, each word vector is combined with a positional encoding. A positional encoding simply consists of a vector that enumerates the position of a word in the body of the text. The reason why this is needed is that, unlike recurrent neural networks and Long-Short Term Memory networks, the transformer architecture does not process text sequentially, i.e., it does not preserve the order. Therefore, the positional encodings are needed to send the network a signal regarding the position of each word in the text.

After the positional encoding, the text enters the encoder. The encoder consists of a sequence of blocks that have the same general architecture. In the small T5 architecture, there are 6 such blocks. Each encoder block consists of a self-attention layer followed by layer normalization, a feed forward layer, and another round of layer normalization. Self-attention is a mechanism by which the context-dependent meaning of a word is encoded. This is best demonstrated by an example. Consider the sentence: "I gave my cat a treat, and she enjoyed it." What does "she" and "it" refer to? It is clear that "she" refers to the cat and "it" refers to the treat. This meaning is captured by the transformer using self-attention. I will not relate here the details of the mechanics of self-attention, but I will note that the computation follows a typical query-key-value method.

Following self-attention, layer normalization is applied. Layer normalization is a common method for reducing internal covariate shift, a phenomenon in which the distribution of the output of a layer changes dramatically in an undesirable way during training. Layer normalization remedies this issue by normalizing the input using the mean and standard deviation of the input to a layer.

After layer normalization, the text goes through a standard feed forward neural network. This is followed by yet another round of layer normalization. Note that throughout the architecture there are residual connections. These are represented in Figure 1 by dotted lines inside the rectangular blocks.

The decoder blocks have a very similar structure as the encoder blocks, except that they have a special encoder-decoder attention layer. This works in a way that is similar to self-attention, except the keys and values come from an encoder block. In other words, the decoder blocks are paying attention to the output of the encoder blocks. The small T5 architecture has 6 decoder blocks.

Finally, after the decoder blocks, there is a linear layer that takes the output of the decoder and projects it to the dimension of the vocabulary. This is fed into the softmax layer which outputs a probability distribution over the vocabulary.

T5 is very similar to the popular BERT language model. However, there are a few notable differences. First, the large T5 model

has about twice as many parameters as BERT. Transformer language models consist of a stack of blocks, where each block is like a miniature neural network (i.e., it contains an attention mechanism on a feed-forward neural network). Most models consist of one such stack, but T5 has two stacks, which is why it has twice as many parameters. Second, T5 is pre-trained on the novel Colossal Clean Crawled Corpus (C4). Pre-training a neural network language model is the process of teaching a network generalized knowledge about natural language. The process consists of taking a large text corpus (such as all of Wikipedia or even the entire Internet), feeding it through the network, and making the network perform a task such as next token or sentence prediction, or in the case of models like T5 and BERT, masked token prediction, where the network must guess the tokens that have been "masked" or deleted in the corpus. Pre-training on data scraped from the internet presents many problems, as a lot of text on the internet is not in a natural language or is corrupted in some fashion. The C4 dataset is a cleaned-up version of datasets scraped from the internet, and pre-training on this higher quality dataset gives T5 a significant advantage. Finally, T5 has been optimized for transfer learning since it also pre-trains on data for downstream tasks such as text summarizatioin. Transfer learning is the process of taking a pre-trained model such as T5, modifying its final layers, and fine tuning the network for tasks such as machine translation, text classification, and text summarization. All of these improvements to the typical transformer architecture make T5 an excellent choice for the backbone of a text summarization system.

The Python library Hugging Face has many different language models including T5. These models come in the form of PyTorch nn.Modules, which allows for customization. This is particularly important for this project, as some of the fine-tuning techniques require modifications to be made to the architecture. Pre-trained weights can be downloaded and fine-tuned. For this project, I use the small Hugging Face T5 model with pre-trained weights and the accompanying tokenizer.

3 WIKIHOW DATASET

The WikiHow dataset was introduced by [3] as an abstractive text summarization dataset. WikiHow is a website where people from all walks of life can write "how-to" articles on topics as far ranging as "How to Survive an Encounter with an Ostrich" to "How to Change a Car Battery." A typical article contains several paragraphs each of which is prefaced by a short summary. [3] used scrapy from Python to collect the data. In particular, they concatenated the paragraphs of each article to constitute the "x-data," and they concatenated the summaries of each paragraph to constitute the target summary. There are 230,843 such pairs. The average article is about 580 words, and the average summary is 62 words. The vocabulary size of the entire corpus is 556,461 words. The following is an example from the training set:

Input Text: *Caged animals such as hamsters, guinea pigs, rodent, reptiles, and amphibians can be taken to a friend's or sitter's home. Create a document that outlines the feeding and water needs, cleaning schedule, and temperature control. Pack all of the things that mimic your pet's environment at your home such as bedding, heated surfaces, and decorations. If the cage is not mobile, someone will need to come check on your pet daily. A rabbit, ferret, or guinea pig is live bait in the wild. Relocating your pet to a home with canines or young children can be stressful and dangerous for your pet. If your pet is used to being in a quiet home with only adults, find a place that mimics that. The relocation environment is extremely important. Your pet can become disoriented and possibly sick from sudden lifestyle changes. Birds and cats enjoy being in consistent, familiar environments. Birds may become restless and pluck their own feathers in unfamiliar environments. Similarly, cats can become distressed in different environments. It's best to find a sitter to stay in your home or stop by your home every day. If you have a bird, make sure that the sitter is comfortable and knowledgeable about birds. Cats should not be left alone at someone else's home. They are likely to wander, escape, and try to return home. A kennel should be the last resort for this type of pet. Make sure the kennel you choose caters to your pet. If your pet will be at a kennel that also houses dogs and cats, it is best if it will be housed in a separate area. Always check out the kennel before you allow your pet to stay there. Ask your veterinarian about boarding options for this type of pet. Some veterinarians provide boarding services as well.*

Target Summary: *Relocate your caged pet. Choose an environment that is similar to your home. Get an in-home sitter for birds and cats. Take your pet to a kennel.*

The authors discuss a number of advantages that make this dataset superior to other text summarization datasets, including the popular CNN/Daily Mail dataset. First, most other datasets consist of news articles, which are particularly easy to summarize. This is because they are often shorter, and the first few sentences often already provide a summary for the entire article. Second, the WikiHow dataset has a higher level of abstractness. Abstractness is defined as the number of n-grams that are in the reference summary but are not in the article. An n-gram is defined as a sequence of n words. Articles with a higher level of abstractness are harder to summarize, as simply copying portions of the article does not suffice as a summary. The WikiHow dataset also has a higher compression ratio. The compression ratio is defined as the ratio of the average length of sentences to the average length of summaries. Articles with higher compression ratios are more difficult to summarize, as the semantics of the longer sentences of the article may be difficult to compress into the shorter sentences of the summary. The WikiHow dataset has a compression ratio of 2.38, while the CNN/Daily Mail dataset has a compression ratio of only 1.44. All of these properties make the WikiHow dataset more challenging for a text summarization system. The difficulty of the dataset is a major motivation for using it to fine-tune T5.

The authors of [3] used the WikiHow dataset to train a number of deep neural networks as well as other machine learning systems. In particular they looked at a seq-to-seq network with attention. They found that it achieved a ROUGE-1 score of 22.04 on the WikiHow dataset, compared to the CNN/Daily Mail datset, which achieved a ROUGE-1 score of 31.33 (I discuss the ROUGE score below). These results provide verification that the WikiHow dataset is more challenging to summarize than the traditional news-based datasets. The authors of [5] and [3] did not consider T5 or any other transformer language model. Therefore, my project presents new insights into the capabilities of T5. If T5 is able to adequately capture the challenging WikiHow dataset, then this will be an indication of the

network's superiority. I next turn to the ROUGE score, which I utilize to evaluate T5.

4 ROUGE SCORE

The primary metric used in the text summarization field is the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score and was introduced in [4]. There are a number of different ways of defining the ROUGE score, each of which has its advantages and disadvantages. One way of defining it is

$$ROUGE_{Recall} = \frac{\#\ overlapping\ ngrams}{\#\ of\ words\ in\ reference\ summary} \tag{1}$$

This definition adequately captures the recall of the summarization. However, a summary produced by the system may perform well on this metric but still be a poor summarization. While the summary may contain many of the words in the reference summary, it may also contain a large number of unnecessary or meaningless words. Therefore, we must also consider

$$ROUGE_{Precision} = \frac{\#\ overlapping\ ngrams}{\#\ of\ words\ in\ predicted\ summary} \tag{2}$$

This defintion captures the precision of the summarization. However, it also has a downside. It could occur that many of the words in the predicted summary are in the target summary, yet the predicted summary still may leave out many necessary words from the target summary. Therefore it is important to look at both of these metrics. One way to convey the importance of the recall and precision in a single metric is to compute the F1 score. In this context, the F1 score is defined as

$$ROUGE_{F1} = 2 \times \frac{ROUGE_{Recall} \times ROUGE_{Precision}}{ROUGE_{Recall} + ROUGE_{Precision}} \tag{3}$$

In this project, I consider the recall, precision, and F1 definitions of the ROUGE score with $n = 1$. For the remainder of this report, *ROUGE* will refer to the $ROUGE_{F1}$ score with $n = 1$ unless otherwise noted.

It should be noted that there is broad agreement that the ROUGE metric is an insufficient way to evaluate a text summarization system. The reason for this is that the ROUGE metric deems a summary to be good when there is a lot of overlap of the n-grams in the target and predicted summary. However, it can be argued that, for example, two summaries of a text may both adequately capture the meaning of a text, yet the two summaries have few words in common. With this difficulty in mind, there emerges another difficulty, namely that evaluating the goodness of a summary can be rather subjective. However, as of yet, there is no method that has been proposed to overcome these obstacles. Therefore, despite these flaws, the ROUGE metric is the best quantitative metric for evaluating a text summarization system. But, these difficulties suggest that a quantitative evaluation should be supplemented with a qualitative evaluation, so it is important to compare different text summarization systems on concrete examples that they produce.

5 FINE-TUNING TECHNIQUES

In this report, I compare three different fine-tuning techniques. Each technique modifies the way in which the parameters of a network are updated during fine-tuning. In the first and simplest technique, all of the parameters of the network are updated during the entire fine-tuning process. However, there is reason to believe that this approach is not optimal. One of the difficulties faced by this method is the potential for catastrophic forgetting. Catastrophic forgetting is the phenomenon when the performance of a trained neural network is diminished rather than improved when training is continued on a new dataset.

Given the threat of catastrophic forgetting, it is often advisable not to fine-tune all the parameters of a network, at least not during the early stages of fine-tuning. An alternative strategy that works for transformer models, called "adapter tuning," proposed in [1] is to only train adapter layers that are inserted into the pre-existing neural network architecture. These adapter modules consist of two dense layers, the first of which upsamples the input and the second of which downsamples to the dimensions of the input. These modules are inserted after every transformer module in the network. During fine-tuning, the original parameters of the network are frozen, and only the adapter modules are trained. The key hyperparameter in this method is the adapter size. This refers to the number of dimensions to which the input to the adapter is upsampled. It is recommended that for transformers such as T5, which have an input dimension of 512 into the adapters, that the adapter dimension be set to 2048. It is well known that the higher layers of a neural network closer to the output require more fine-tuning than the layers closer to the input. The reason for this is that the high-layers catch more fine-grained and task-specific features of the data, while the lower layers encode more general features. The fine-grained features need to be adapted to the new data while the courser features do not. This raises the question of how many of the higher layers should be fine-tuned. Adapter tuning allows this question to be answered in an adaptive fashion, as the adapter modules learn which layers require fine-tuning and which layers do not.

The third fine-tuning technique that I will be considering is called "gradual unfreezing." This method, presented in [2], is motivated by similar insights as behind the adaptive tuning method. In particular, since the lower layers encode more general knowledge than the higher layers, the higher layers require more fine-tuning than the lower layers. Therefore, in gradual unfreezing, the parameters of a network are unfrozen one layer at a time during fine-tuning, starting from the top of the network and working down. In encoder-decoder models like T5, it is suggested that one should unfreeze the layers of both the encoder and decoder in parallel. The essential hyperparameters are how often layers are unfrozen during fine-tuning and how many are unfrozen at a time. Following the conventions of the literature, I will be unfreezing just one layer of the encoder and one layer of the decoder at a time. The frequency with which layers are unfrozen is determined by computing the total number of iterations needed for fine-tuning and dividing this total by the number of layers in the encoder and decoder.

6 TRAINING BEHAVIOR

The training was done for 2 epochs with a batch size of 8. Therefore, the total number of iterations was 39,314. Figures 2 through 5 below display the behavior of each fine-tuning method during training. Figure 2 shows the training loss for each fine-tuning technique.

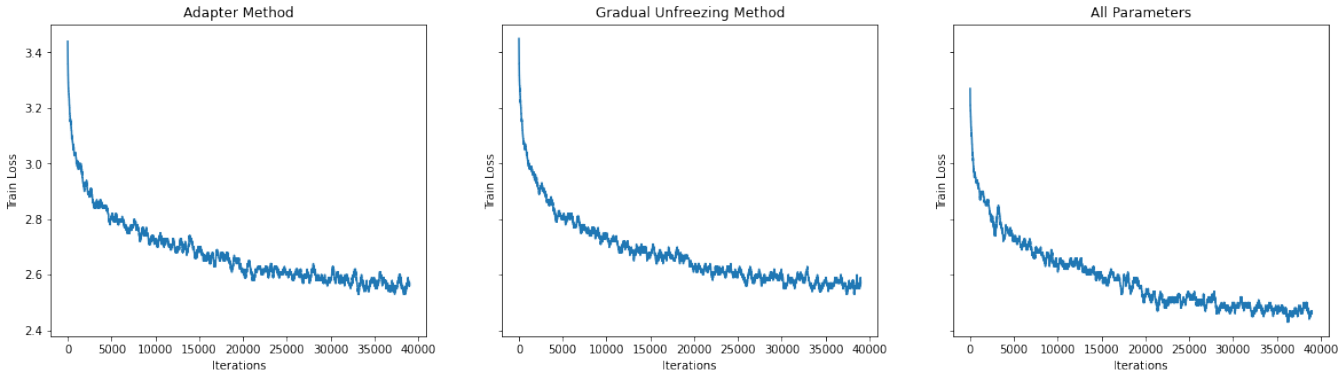

Figure 2: Train loss for each fine-tuning technique

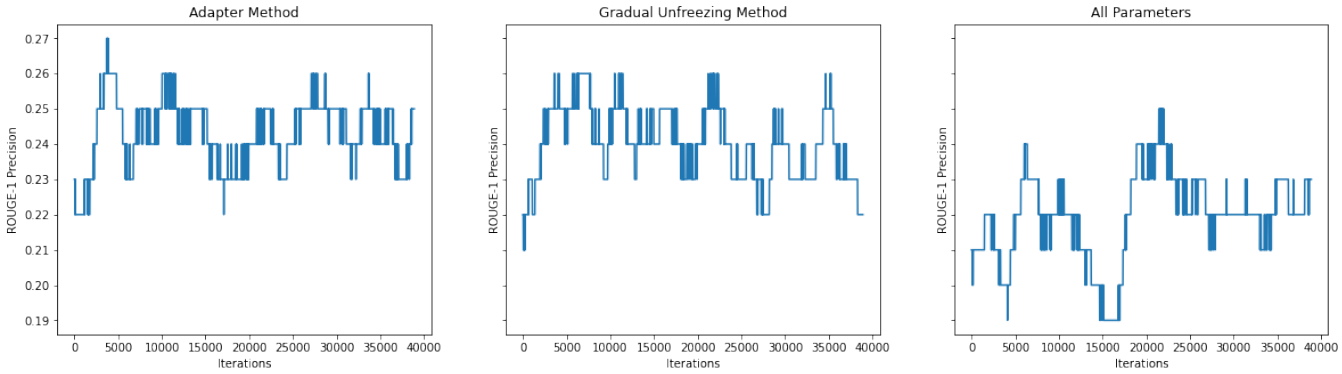

Figure 3: Train ROUGE-1 precision for each fine-tuning technique

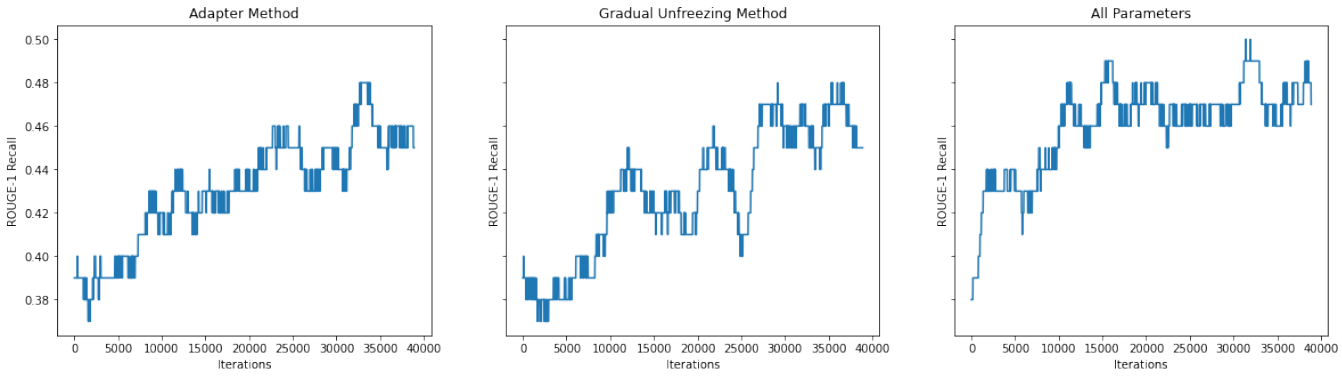

Figure 4: Train ROUGE-1 recall for each fine-tuning technique

The train loss was computed after every iteration. Displayed in Figure 2 is the moving average of the train loss with a window size of 300. There are two important observations to make about this figure. First, it is interesting to note that the training loss of the adapter and gradual unfreezing methods have a very similar behavior. Second, the "all parameters" method, the method which trained all the parameters simultaneously, had a training loss that decayed more rapidly and which was somewhat lower at the end of training compared to the other two methods.

Figure 3 displays for each fine-tuning method the behavior of the $ROUGE_{precision}$ score during training. This metric, along with the next two to be discussed, was computed on a batch of training examples every 50 iterations. The moving average of window size 300 is what is displayed in the graphs. There are a few important things to note about this figure. Overall, all three methods demonstrated

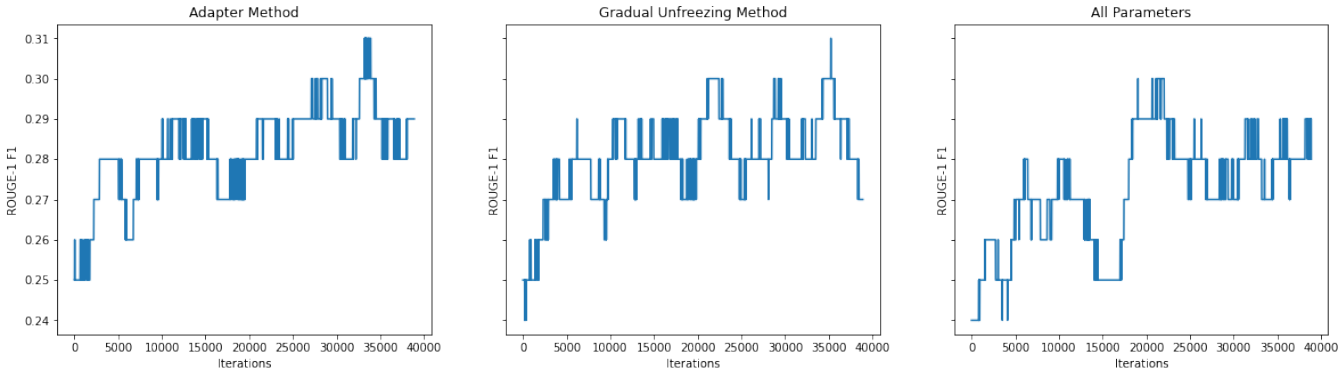

Figure 5: Train F1 for each fine-tuning technique

very little improvement in this metric. First, again it can be seen that the adapter and gradual unfreezing methods behaved similarly. They also on average maintained a score a few points higher than the all parameters method. Second, the behavior of this score for the all parameters method is considerably more volatile compared to the other two methods. In particular, sometime between the 10,000th and 15,000 iterations, the $ROUGE_{precision}$ score suddenly collapses. This phenomenon is similar but not quite as serious as the catastrophic forgetting described above. Yet it demonstrates a potential danger of fine-tuning all the parameters of a deep neural network, which is that it is possible that the performance of the network will suddenly degrade as if it has forgotten all that it had previously learned about the data. Fortunately, it appears that the network recovered from this loss but still ultimately achieved a lower $ROUGE_{precision}$ score on average compared to the other two methods.

Figure 4 displays for each fine-tuning method the moving average of the $ROUGE_{recall}$ score during training. All three methods showed significant improvements in this metric throughout training, and the metric is about 20 or more points higher than the precision for each model. By the end of training, the all parameters method achieved the highest score on this metric. It is important to note the sudden drop that occurred between iteration 20,000 and 25,000 for the gradual unfreezing method. This occurred shortly after unfreezing a layer in both the encoder and decoder. It appears that the kind of forgetting that occurred for the all parameters method for the precision may have occurred for the recall under the gradual unfreezing method. Adding more parameters to be optimized caused a temporary forgetting.

Figure 5 displays the moving average of the $ROUGE_{F1}$ score of each method during training. Since the F1 score is a kind of average between the recall and precision, it is not surprising to see that the performance of the methods on this metric is somewhere between the performance of the methods on the precision and recall. One important observation to note is that the forgetting that occurred for the all parameters method seems to have had a greater impact on the F1 performance than the forgetting that occurred for the gradual unfreezing method.

7 TEST RESULTS

The $ROUGE_{F1}$ score on the WikiHow test dataset for each method is displayed in the table in Figure 6. Note that the "baseline" refers to a network that was not fine-tuned. It can be seen that all three methods outperformed the baseline. Gradual unfreezing performed the best with a score of 30.3. The adapter method was a close second with a score of 29.1, and in third is the all parameters method, which achieved a score of 27.6. Therefore, it is clear from these results that fine-tuning T5 for the WikiHow dataset has a significant impact on performance. In addition, the more nuanced fine-tuning techniques, the adapter and gradual unfreezing methods, perform better than the all parameters method.

8 EXAMPLES FROM WIKIHOW DATASET

Figures 7 and 8 contain generated summaries from each model for examples 1 and 2, respectively. The original article that was used to generate the summaries in example 1 is about how to maintain laminated floors. The first observation to make is that the fine-tuned models all use the same imperative tense that is found in the target summary, while the baseline mode uses a mix of both. Also, the summaries generated by the fine-tuned models are more general than the summary generated by the baseline model. The summaries from the fine-tuned models describe a number of ways to maintain laminated floors, while the summary of the baseline model only mentions one. However, the summaries generated by the fine-tuned models are not perfect. All three say "Replace chairs with rubber wheels." According to the target summary, a more accurate phrase is "Replace plastic casters with rubber wheels."

The original article used in example 2 found in Figure 8 is about how to set Bing as a default search engine. It is interesting to note that the summary from the baseline model makes very little sense. It starts off with an incomplete sentence, and the rest does not provide a good summary of the article at all. The adapter model also did not provide a good summary, as it only states "Close the browser." Both the gradual unfreezing and all parameters models provide decent summaries. They both address most of the items in the target summary. However, perhaps the summary of the all parameters model is too terse.

ROUGE-1 F1 – WikiHow Test Dataset	
Fine-Tuning Technique	**Score**
Baseline	23.6
All Parameters	27.6
Adapter	29.1
Gradual Unfreezing	**30.3**

Figure 6: Test results on the WikiHow test dataset for each fine-tuning method

Fine-Tuning Technique	Summary
Baseline	Do not drag or push furniture across the floor when you move it around. Lifting your furniture will prevent scratches and scuff-marks that can be caused by dragging your furniture.
Adapter	Put mats at your doorways. Replace chairs with rubber wheels. Lift furniture if possible.
Gradual	Place rugs under your furniture. Use mats. Replace chairs with rubber wheels. Lift the furniture.
All Parameters	Place rugs or carpets in heavy foot traffic. Place mats at your doorways. Replace chairs with rubber wheels. Lift the furniture.

Figure 7: WikiHow Example 1 - Target Summary: Attach protector pads to the bottom of furniture. Put carpets or rugs on your laminate flooring. Place entry welcome mats at doorways. Replace plastic casters with rubber wheels. Lift furniture instead of dragging it. Slide heavy furniture across the floor. Keep the humidity levels in your home between 35 and 65 percent. Keep the nails of your pets trimmed. Avoid wet mopping. Clean spills from the laminate flooring as soon as they occur. Don't use a vacuum with rotary brush heads. Make repairs quickly.

Fine-Tuning Technique	Summary
Baseline	the list. Click "Set as default" on the lower right corner of the window. Bing will appear with the "Default" text under its status. Click the "Close" button on the lower right corner of the window to exit.
Adapter	Close the browser.
Gradual	Search for Internet Explorer. Go to the Tools menu. Click "Add-ons" from the top right corner of the header toolbar. Select a search provider. Set Bing as default.
All Parameters	Go to Tools. Manage add-ons. Set as default. Close.

Figure 8: WikiHow Example 2 - Target Summary: Launch Internet Explorer. Open the Manage Add-ons menu. Click on "Search Providers" under the "Add-on Types" column. Set Bing to default. Exit the menu.

9 TESTING ON CNN DATASET

When evaluating the fine-tuning of a language model like T5, it is important to test the extent to which the model still generalizes after the fine-tuning. Therefore, using the ROUGE score, I have tested each model on the CNN news article test dataset, a dataset that the models did not see during fine-tuning. This dataset consists of CNN news articles together with brief summaries of the articles. It is part of the CNN/Daily Mail dataset discussed above. The average ROUGE scores achieved by the models on this dataset are in Figure 9. The baseline model far outperformed the fine-tuned models, achieving a score of 36.4. The best fine-tuned model is the gradual unfreezing model, which achieved a score of 23.4. Therefore, it is apparent that the fine-tuning eroded the ability of T5 to generalize

ROUGE-1 F1 – CNN Test Dataset	
Fine-Tuning Technique	**Score**
Baseline	**36.4**
All Parameters	14.8
Adapter	23.7
Gradual Unfreezing	23.4

Figure 9: Test results on the WikiHow test dataset for each fine-tuning method

to datasets outside of the WikiHow dataset. However, the all parameters model scored much lower than the adapter and gradual unfreezing models, indicating that the more nuanced fine-tuning methods maintain more generalizability after fine-tuning.

10 EXAMPLES FROM CNN DATASET

Figures 9 and 10 contain generated summaries from each model for examples 1 and 2 of the CNN dataset, respectively. The article used to generate the summaries in Figure 9 is about an event where a marriage ceremony was held for pets. It is clear that the all parameters method produced the worst summary. It doesn't even mention that any wedding occurred. The gradual unfreezing method produced a slightly better summary, as at least it mentions something about a wedding. The summary produced by the adapter model is fairly good. It states clearly the central topic of the article. It may even be said that it is better than the baseline summary that, while more verbose, does not mention anything about a wedding. However, the summary from the baseline model includes more details than the other summaries.

The article used to generate the summaries in example 2 is concerned with recent scientific evidence regarding the evolutionary history of humans. The all parameters model again produced the worst summary. It consists of one sentence repeated twice. The adapter method produced a summary with a bit more information. The summary from the gradual unfreezing method is the most informative of the three summaries from the fine-tuned models. However, only the baseline model produced a summary with any significant details that overlap with the target summary. In particular, it mentions the speculation about the ancestors of humans having tentacles, which is they key point of the article.

It is clear from these examples that there is one fundamental flaw of the summaries produced by the fine-tuned models. We saw in the WikiHow examples how the baseline model did not capture the imperative case of the target summaries. Here, we see that the fine-tuned models carry over that imperative style and fail to adopt the "matter-of-fact" style of the target summary. The fine-tuning has made these models too accustomed to the WikiHow dataset. The imperative style of the target summaries for the WikiHow dataset have inhibited the fine-tuned models from capturing different kinds of styles of summarizing.

11 CONCLUSION

In this project, I have compared three techniques, the all parameters method, the gradual unfreezing method, and the adapter method, that are commonly used to fine-tune deep neural networks. In particular, using the WikiHow dataset, I used these techniques to fine-tune T5 for abstractive text summarization. The test results indicate that the adapter and gradual unfreezing fine-tuning techniques strike a balance between a model being able to generalize to other datasets and being able to perform well on a dataset-specific task, such as summarizing WikiHow articles. These two techniques achieved the highest ROUGE score on the WikiHow test dataset, and they did not perform nearly as badly as the all parameters method on the CNN dataset.

Overall, this project has advanced our understanding of the T5 text summarization model. The WikiHow dataset is harder than the datasets usually used for text summarization. Fine-tuning T5 on this dataset has truly put it to the test. The fact that T5 was able to produce informative summaries is an indication that it is a quality text summarization model. This project's exploration of the trade-off between fine-tuning and generalization has shed new light on the possibilities and limitations of T5.

Big data analytics is all about working with large models and massive datasets. Due to time and resource limitations, this project could only work with a smaller T5 model on a limited dataset. However, the results of this project have brought some insight into the scalability of T5 and text summarization systems in general. In particular, the results of the project reinforce the principle that more data is usually better. If text summarization systems are to be truly general, they must be trained on diverse datasets. Fine-tuning on just WikiHow articles, as was done in this project, will produce models that are fairly good at summarizing other WikiHow articles. But, as we saw, this may diminish a model's ability to summarize other kinds of text. Perhaps if a T5 model were to be fine-tuned with a larger dataset containing data from many different domains, it would generalize better than the models studied in this project.

One may suspect that utilizing a fine-tuning technique such as the adapter method or gradual unfreezing may result in shorter training times, since fewer parameters are updated. However, this is not the case. The major computational bottleneck of the gradient descent algorithms for deep neural networks is the computation of the gradients. Since each of the these fine-tuning techniques requires the parameters of various layers throughout the entire network to be unfrozen, all the gradients must be computed even though only a smaller subset of the parameters are updated. Therefore, these fine-tuning techniques do not promise to aid any scaling up as the time to train will always be about the same as if all the parameters were updated during training. The time to fine-tune the models used in this report was around 5 hours, so for larger models and datasets, one should expect training times to exceed this.

A major conclusion of this project is this - more research is needed in the field of text summarization. I have just proposed

Fine-Tuning Technique	Summary
Baseline	ceremony was organized by a new social media app designed for pets. the collective ceremony, which was billed the first of its kind to be held in China, was organized by a new social media app.
Adapter	a mass doggy wedding saw more than 40 pooches tie the knot in a park.
Gradual	Organize a collective ceremony. Attend the wedding.
All Parameters	Organize the collective ceremony.

Figure 10: CNN Example 1 - Target Summary: The pets wear tulle wedding dresses and tuxedos with bow ties. Couples are given marriage certificates after 'exchanging vows.' Wedding is organized to promote a social media app designed for pets. No expense spared as pooches arrived in BMWs and a stretch Hummer.

Fine-Tuning Technique	Summary
Baseline	from the Catalan Institution for Research and Advanced Studies (ICREA). We evolved into bilateral creatures, and what we evolved from, has been a cause of some debate. one theory suggests that our ancestors had appendages - or tentacles - that were used for movement and food collection.
Adapter	Observe our distant ancestors. Study how we evolved into bilateral creatures.
Gradual	Observe our distant ancestors. Recognize that humans and other organisms were bilaterally symmetric. Consider the warming climate. Understand how we evolved from human ancestors.
All Parameters	Learn about the evolution of human ancestors. Learn about the evolution of human ancestors.

Figure 11: CNN Example 2 - Target Summary: Russian scientist says distant ancestor of humans had tentacles. They lived more than 540 million years ago and used them for food. It's likely they also had a complex nervous system like we do today. Challenges another theory that says our ancestors were more worm-like.

one possibility of future work when discussing the need to study more diverse datasets. In addition, more work needs to be done investigating the efficacy of the fine-tuning techniques discussed in this project when applied to other types of language models used for text summarization, such as BERT. Perhaps these techniques will yield different results when applied to other models. Also, in light of the conclusions of this project, it is clear that what is needed is new fine-tuning techniques that better balance the trade-off between generalization and performance on a dataset-specific task. Such techniques could unlock the hidden potential of text summarization systems and would have far reaching implications for their myriad applications.

12 APPENDIX: HONOR CODE

On my honor, as a University of Colorado Boulder student I have neither given nor received unauthorized assistance. – Nate Omdalen

This project was completed individually by Nate Omdalen

REFERENCES

[1] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. *ArXiv* (June 2019). https://arxiv.org/abs/1902.00751

[2] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *ArXiv* (May 2018). https://arxiv.org/abs/1801.06146

[3] Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A Large Scale Text Summarization Dataset. *ArXiv* (Oct. 2018). https://arxiv.org/abs/1810.09305

[4] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv* (Oct. 2019). https://arxiv.org/abs/1910.10683