Summarization On AMI Meeting Corpus

Bharat Runwal

May 2021

1 Introduction

In this task we are givent to generate summaries of AMI meeting corpus dataset.Summarization can be done in two ways one is the Abstractive Summarization and the other is Extractive Summarization , in the Abstractive summarization the model generates the human-like summary which are not exactly taken from the source whereas in the Extractive Summarization , the model takes relevant/important sentences from the source and then concatenates them to give the summary.This Work done here has a **Major Limitation** for the limited Computing resources because of the Dependecny of the models on the Length of sequences due to the full Attention mechanism.I will also describe other methods which remedy this limiation for longer sequences in the end of the report.

2 Dataset

I have used the Dataset given in the Gdrive in the assignment, it consists of 3 splits ,for train , validation and test set repsectively. Using pandas i created the csv files for each of the 3 splits which contains the Two columns one is the source meeting paragraph and it's respective Summary.
Some insights into the Dataset :

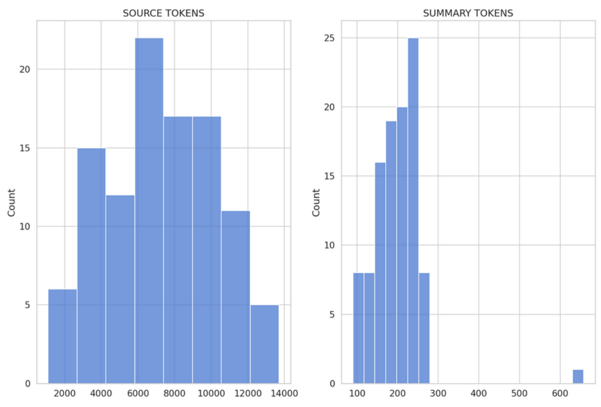

Figure 1: Number of Tokens of Source meeting para and Summary

- We can see that the length of the source paragraph is quite bigger than what generally Language models take with limited GPU resources.
- We can see that the length of summary are skewed that is mostly smaller than 300.

If we don't have a large GPU resources then the Language models have constraints on the input tokens size, so there is always this problem of GPU resources as these are costly, so because of limited GPU resources , here in this assignment i have constrained myself to input **512,1024** tokens and summary length of **128** mostly.
There are lot of results that has been established which proves that taking a portion of the input can produce very good results on certain tasks,the reason they believe is because of the information content in those relevant portion i.e. most of the useful information is present in that part only so using a portion of the wohole input may give good results and this has been proved also.But, howerer one can not say this for general without trying for their own dataset.
The length of each splits :

- Train set : 105
- Valid set : 17
- Test set : 20

3 Tokenizer

One of the important step for building the Language model is the tokenizer, now we can choose a pre-trained tokenizer or we can train our own tokenizer (We can use here ByteLeveTokenizer) on our dataset.Both have certain benefits but most of the time using pre-trained tokenizers gives good results as the tokenizers were traiend on a lot of various/different types of data and as our dataset is small so we can say that we can go for pretrained tokenizer.
Now According to our choice of pre-trained model we will use we can grab the respective pre-trained tokenizer from Huggingface library.
I have used the following tokenizers available on the Huggingface Library:

- T5Tokenizer
- T5TokenizerFast
- PegasusTokenizer
- BartTokenizer

Note: I have tried using the T5Tokenizer Fast here , this tokenizer is based on the rust based tokenizers which offers high performance tokenizer to speed up the tokenizing process.

4 Models

This section provide details about the model i have used in order to fine tune on The AMI meeting dataset. Due to the constraints of GPU memory, i was not able to finetune on current state of the art Summarization models like Big-bird which was released recently (More discussion at the end).
The reason i have used the models pre-trained on the Huge News dataset is because it has been seen the results of the model pre-trained on News dataset can generalize to the other domains well.
I have used the following pre-trained models from the huggingface library :

- T5-base ,T5-small
- Pegasus-xsum
- bart-large-xsum

Note: I was able to fine tune the T5 model on the google colab , and the rest of the model i finetuned on my own machine which has two GPUs and used distributed training (availbe code form huggingface).

4.1 Why T5

Text-To-Text Transfer Transformer (T5) was introduced by the google on the chain of research in the field of using Transfer learning to produce SOTA results on NLP text-text task.
How is it different from Bert : So as we know BERT models can output only a span of inputs or the label of class in resepctive task settings , the T5 frames each task in a text to text format i.e input is text string and output is also text string.This is benefical because now the same model,loss function and hyperparameter setting can be used across any task like summarization , classification etc.This T5 model was pre-trained on C4 dataset which is a Large clean crawled corpus which uses different preprocessing like deduplication(removing of duplicated examples) etc on Large Common crawled datasets and it was bigger , diverse than the other Datasets before that.
Note: All the Results are shown in the later portion of the report.

4.2 Why Pegasus:

This is one of the most popular and SOTA model for Abstractive summarization introduced by Google.Earlier model we discussed T5 was pre-trained using self supervised or unsupervised learning and this is true for other models also like RoBERTa,GPT2 etc.The Pegasus paper focusses on using a specific pre-trained self supervised objective which they called Gap sentence generation for transformer encoder-decoder model to improve the fine tuning performance.
This model is also pre-trained on a large corupus like T5 but also fine tuned on the specific dataset , i have used the Pegauss-xsum and finetuned it on our dataset.

4.3 Why BART

This model actually combines two things together , its a denoising autoencoder for pre-training seq-seq language models.We can see BART model as generalizing of Bert(Bidirectional Encoder) and GPT 2 (left to right decoder similar to decoder only transformers).So BART model combines both of them.This is the reason i chose this model to fine tune on our dataset.

5 Fine Tuning

I have used the Huggingface library as it's API is very easy to use and most of the code is derived from their example notebooks. I have fine-tuned on the small training set with different models and Various hyperparameter choices.I have fine tuned the T5 model on the google colab and the rest models in the other machine(HPC) because of the size of the other models as the colab has limited amount of GPU memory of nearly 12 GB. I have also used pytorch-lightning which is very easy to use because of its API and have a lot of addtional parameters to play with.

5.1 Hyperparameter Selection:

I have taken a naive way of selction of hyperparameter i.e. using GridSearch,because of the constraint of the computation resources Searching on large-space of parameters was not possible , os i tried different values of hyperparameters and saw the validation results and accordingly choose the best hyperparametrs of the model.
List of hyperparameters:

- Learning rate ,Weight Decay(L2 norm) ,Adam parameters
- Dropout
- Number of Epochs
- Batch size (here i didn't have much freedom because of computation issues)

I have also used Early stopping wherever necessary.For the Beam serach i have used 4 and 6 mostly(number of beams)
Note: There exists other Hyperparameter Optimization methods like with pytorch lightning we can use ray-tune, which provides more better Algorithms like Baysian optimization , HyperOpt which are famously used in hyper parameter tuning.

5.2 Results

All the metrics are reported on Test set and Precision is reported here for different metrics.

5.2.1 T5 :

With using T5- base model from huggingface and fine tuning it on our dataset , the best result was for the following set of hyperparameters:
lr = 5e-4 , epochs = 5 , batch-size = 4 , Grad_clip at 0.5(Norm clipping):

- Rogue-1: 0.46273
- Rogue2:0.1525
- RogueL:0.27905
- RogueLsum:0.27716

5.2.2 Pegasus

With using Pegasus-large pretrianed model and fine tuning it on our dataset, the best result was for the following set of hyperparameters:
weight-decay = 0.01,epochs = 2000 ,lr decay with 1e-6,

- Rogue-1: 0.406699
- Rogue2:0.12984
- RogueL:0.2372

- RogueLsum:0.23654

Note: Note here that the Rogue-1 score for the pegasus model is lower than the T5 , the reason i think is the choice of hyperparameters , i was not able to do hyperparameter serach for pegasus because of the limited resources.

5.2.3 BART

With using Bart-Large-Xsum pretrianed model and fine tuning it on our dataset, the best result was for the following set of hyperparameters:
Epchs = 3, batch-size =4, lr-rate = 5e-5 :

- Rogue-1: 0.5746
- Rogue2:0.2307
- RogueL:.40181
- RogueLsum:0.40119

Note: Here also i didn't performed a hyperparameter search because of the limited resources and time ,but we can see that it clearly gave the best results on our dataset and i believe that by carefully tuning the hyperparameters we can produce even better results.

5.3 Other Results

Here i list the results for different sets of hyperparameter and other optimization strategy, ALl the metrics are reported on Test set.

5.3.1 T5

Changing Learning rate : Epchs = 3, batch-size =4, lr-rate = 3e-4 :

- Rogue-1: 0.4470
- Rogue2:0.1360
- RogueL:0.2827
- RogueLsum:0.2850

Epchs = 3, batch-size =4, lr-rate = 5e-4 :

- Rogue-1: 0.457095
- Rogue2:0.1338
- RogueL:0.2596
- RogueLsum:0.26032

Using SWA optimization strategy: Here i investigated by using different optimization strategy then SGD , SWA stochastic weight averaging also has the power of ensemble models like fast Geometric Ensembling methods, it has been found that the flatter minimas are the ones which leads to better generalization.
Epchs = 5, batch-size =4, lr-rate = 5e-4 :

- Rogue-1:0.4274
- Rogue2:0.1167
- RogueL:0.25074
- RogueLsum:0.2509

Gradient Clipping (Norm) Here i used gradient clipping i.e. If the Norm of the weights are higher than the threshold value clip the gradients to the threshold value, this prevents the problem of exploding gradients and considered a good parameter to use in practice .

- Rogue-1:0.46273
- Rogue2:0.15256
- RogueL:0.2790
- RogueLsum:0.2771

5.3.2 Pegasus

During pre-training of pegauss i have saved the model at different checkpoints between 2000 epochs.

Epchs = 500 ,same parameters defined earlier

- Rogue-1:0.40211
- Rogue2:0.1213
- RogueL:0.2300
- RogueLsum:0.2313

Epchs = 1500 ,same parameters defined earlier

- Rogue-1:0.4066
- Rogue2:0.1298
- RogueL:0.23720
- RogueLsum:0.2365

Epchs = 2000 ,same parameters defined earlier

- Rogue-1:0.40439
- Rogue2:0.12458
- RogueL:0.2339
- RogueLsum:0.2352

I didn't perform any hyperparamet search here.

5.3.3 BART

Epchs = 3 ,same parameters defined earlier

- Rogue-1:0.57466
- Rogue2:0.2307
- RogueL:0.40181
- RogueLsum:0.4011

Epchs = 5 ,same parameters defined earlier

- Rogue-1:0.5278
- Rogue2:0.1892
- RogueL:0.3640
- RogueLsum:0.3651

6 Further Work

Now , as we saw that the GPU resources are one of the key thing in getting SOTA performances especially in case of longer sequences. So i here describe some of the models that are available for us to use in out task for longer sequences which would definitely gives the better performances than the methods described earlier.

6.1 BigBird model:

The model was introduced by the Google. The main important difference in this and the other model like BERT is the Quadratic Dependency on sequence length is reduced to linear in Big Bird using Sparse Attention mechanism.With the same hardware as the earlier models , Big bird can handle 8x larger sequence lengths. Here they also showed in thir paper that the performance of Abstractive summarization by using longer contextual encoder is improved , so i think this would be very promising model for our case of fine tuning on Meeting Dataset.The model is also available on Huggingface Library.

6.2 HMNet model By Mircosoft

This is a model for Abstractive Summarization , This model utilizes the meeting dataset structure , as they come with natural multiTurn structure with reasonable number of truns like 289 turns per meeting on avg in AMI dataset.And the number of tokesn in a turn is much lower than that of the whole meeting , so Here they adopted a two-level transformer structure to encode the meeting transcrip , one is word-level transformer Which process the tokens of one turn in the meeting and second is Turn- level tranformer which process the information from all turns in a meeting where one turn is given by the word-level transformer.They achieved a Rogue-1 score of 53.02 and Rogue-2 18.57 with statistical margin of 0.05 on AMI Dataset.

6.3 HEPOS model

Recently I read a paper ”Efficient Attentions For Long Document Summarization” , They also address the saeme problem of Computational and memory complexity of large Trasnformers for long document summarization.In this paper they propsed a efficient ”encoder-decoder attention with head-wise positional strides” which effictively pinpoints the salient information from the source.Their two observations are very important, they observerd that the Attention heads are redundant and any inidividal head rarely attends to the several tokens in the row which is intuitively correct also. So they used seperate encoder-decoder heads on the same layer to cover different subsets of source tokens at fixed intervel rather than a continuous.So for a stride of ”s” for h-th head it attends ot ” n /s ” tokens per head where n is tokens ,so it gives Memory Complexity of O(m*n/s) where m is the Output length , Which is Quite better than the Qudratic dependency earlier.
We can use this approach also their codebase is available but i didn’t had time to go through it completely because of exams , but we can definitly adapt to larger sequence lengths using this method for our summarization task.

7 References

- https://github.com/huggingface/transformers/tree/master/examples/pytorch/summarization
- https://www.microsoft.com/en-us/research/publication/end-to-end-abstractive-summarization-for-meetings/
- Huang, Luyang et al. “Efficient Attentions for Long Document Summarization.” ArXiv abs/2104.02112 (2021): n. pag.
- https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html
- Zaheer, M. et al. “Big Bird: Transformers for Longer Sequences.” ArXiv abs/2007.14062 (2020): n. pag.