Abstractive Summarization of Meeting Transcripts

Rajiv Nair, Hailey Wu
rajiv.nair@berkeley.edu, wuqq09@berkeley.edu

1. Abstract

With the development in speech recognition, summarization of spoken English becomes the next hot topic. This paper focuses on using the recent research on Text-to-Text Transfer Transformer (T5) and the BERTSUM model to summarize long transcripts and conversational-based texts. As a result, we find T5 performs better with more succinct texts, after word filtering and with the BERTSUM extractive summaries. We archieve ROUGE-2 F-score of 9.815 on AMI corpus and 4.649 for the ICSI corpus.

2. Introduction

Virtual meetings have become the norm in 2020. With the advancement of speech recognition, live transcript is now a common tool. Building upon it, we would like to explore abstractive summarization of meeting transcripts, which ideally can facilitate the exchange of ideas and information without reviewing lengthy texts.

There are several methods developed for summarizing English meetings (Qin et al.,2017; Shang et al.,2018; Li et al.,2019; Zhu et al.,2020). Qin et al.(2017) looks at using a joint modeling based on Markov chain Monte Carlo (MCMC) chain of both critical discussion points and discourse relations. Shang et al.(2018) focuses on graph-based framework as an unsupervised learning. Li et al.(2019) uses multi-model hierarchical attention mechanisms by integrating inputs from videos, transcripts and audios. Zhu et al.(2020) leverages the encoder-decoder transformer architecture (Vaswani et al., 2017) to build the Hierarchical Meeting Summarization Network (HMNet), with hierarchical structure to capture both token-level and turn-level understanding for multi-speaker situations.

Most of the papers (Shang et al.,2018; Li et al.,2019; Zhu et al.,2020) develop an end-to-end meeting summarization framework, which is also our major focus: a pipeline reading in original scripts and summarizing succinctly with human-readable English.

As Zhu et al.(2020) mentions, compared to document text, the two challenges for meeting summarization are the conversational relationship and lengthy transcript. Each meeting candidate can have a unique role. Potentially, a meeting can be driven by only one person. Meanwhile, the pipeline needs the whole meeting content at once if possible. Because at any point in time, there can be a key topic.

To solve these concerns, we look at the recent end-to-end framework, Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020), achieving state-of-art in abstractive summary of CNN/Daily Mail (Hermann et al.,2015). However, T5 has a limitation of input based on the memory capacity. With long transcripts, we utilize word filtering and BERTSUM (Liu et al., 2019) as pre-steps of T5. The extractive summaries generated by BERTSUM can provide condensed contents. We use ROUGE metrics for automated evaluation to reference summaries, which also help compare to the benchmarks.

It turns out that T5 performs better with more succinct text input. With word filtering and/or BERTSUM extractive summary as pre-steps, ROUGE-1, ROUGE-2, and ROUGE-L F-scores increase gradually in the dev dataset and/or test dataset. In the end, the pipeline reaches ROUGE-2 F-score of 9.815 on the AMI test set and 4.649 for the ICSI test set.

3. Background

3.1 BERTSUM

In addition to the standard SEP token between each sentence, the BERTSUM model changes the input encodings by adding CLS tokens. In the paper, Liu et al., (2019) experiments with various summarization layers, which seems to have no significant difference based on the ROUGE-L metric. Thus, we use the simpler classifier as the summarization layer.

With compilation issues, the published BERTSUM code could not be used as-is. Therefore, we use the paper as a reference and build up the input tokenization and model from scratch.

3.2 Text-to-Text Transfer Transformer (T5)

T5, proposed by Raffel et al. (2020), is a text-to-text-format framework that can use the same model, loss function and hyper-parameters for distinct NLP tasks. Its architecture is based on the transfer learning and the encoder-decoder transformer. The team develops and pre-trains T5 on 750 GB Colossal Clean Crawled Corpus (C4). For fine-tuning, T5 adds task-specific prefixes to the input, such as "summarize: " for the summarization task. Built upon the originally proposed transformer (Vaswani et al.,2017), T5 applies rescaling layer normalization for each sub-component input and a residual skip connection to add the initial input to block output. T5-11B (11 billions parameters) reaches the state-of-the-art performance on the CNN/Daily Mail (Hermann et al.,2015) benchmark with a ROUGE-2 F-score of 21.5.

In this project, we use the Hugging Face T5 library in PyTorch, which requires minimum code development but thorough model understanding, since the model is pretty new.

3.3 ICSI Corpus and AMI Corpus

The ICSI Meeting corpus (Janin et al., 2003) is a collection of 75 meetings during 2000-2002. The speech files range in length from 17 to 103 minutes. There are approximately 795,000 words and 13,000 unique words in the transcripts. The meetings included are "natural" meetings that are not set up for experiments.

The AMI Meeting corpus (McCowan et al., 2005) consists of 171 meetings for around 100 hours. These meetings are not “natural” with a role-playing set up.

4. Methodology and Analysis

4.1 Dataset

Both AMI and ICSI are in the NXT format, which is generated from the application used by the transcribers to develop extractive and abstractive summaries. All relationships are captured in a bespoke XML format across multiple files. Since the transcript is generated by the ASR system, there is a word error rate of 36% for AMI and 37% for ICSI (Shang et al., 2018). Though there are many open-source codes for parsing these two datasets, it still takes a while for us to understand and parse it the way that BERTSUM and T5 can accept.

To handle conversational-based texts and long transcripts, we split each meeting into per-speaker documents, which gives on average 2040 words per document and 465 transcripts for ICSI, and on average 1472 words per document and 556 transcripts for AMI.

4.1.1. Training, Development, Test Split

We follow the Shang et al. (2018) method to split the training/development/test dataset. It has 20 and 6 meetings respectively for AMI and ICSI test set, and 47 and 25 for development set.

Since we parse the data per meeting speaker, some documents might have an empty abstractive summary. For example, speaker A is only agreeing but not contributing to the topic; thus speaker A's words are not related to the human generated summary. These texts won't help the model and thus are dropped. Therefore, strictly following Shang et al. (2018) splitting messes up with the 70-20-10 rule for the dataset. However, we continue on with this method, so that we can compare with others.

4.1.2 Heuristic

ICSI Document Counts (.txt)			
	DEV	TEST	TRAIN
Original Text	148	39	278
A	744	165	1231
B	411	93	697
C	338	79	577
D	159	40	287

AMI Document Counts (.txt)			
	DEV	TEST	TRAIN
Original Text	188	80	288
A	666	281	908
B	382	164	535
C	320	130	452
D	190	83	299

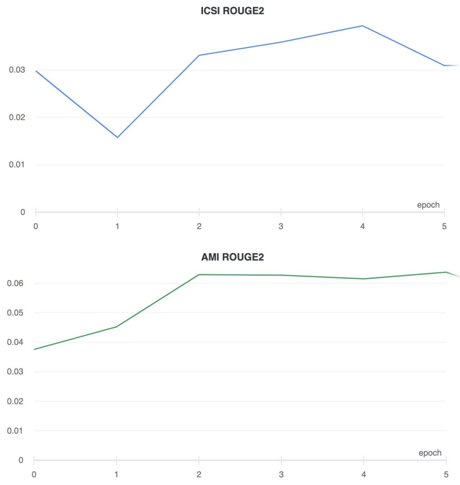
A:512 tokens; B: 1024 tokens; C: B + filtering words; D: C as extractive summary

In addition to per-speaker split, we also test a few additional data cleansing to have more condensed content per meeting speaker.

Original Text per Speaker in 512 Tokens: We chunk the data into 512 tokens for each input.
Original Text per Speaker in 1024 Tokens: We chunk the data into 1024 tokens for each input, since T5's input length is not a hard limit.
Filtering Words per Speaker in 1024 Tokens: We clean out a list of filter words such as "Ok","Umm", etc, and also remove consecutive repeated words.
Filtering Words of Extractive Summary per Speaker in 1024 Tokens: Extractive summary is a subset of original sentences, which can represent the meeting contents more succinctly. This can be achieved in the pipeline by using BERTSUM as a pre-step.

4.2 Pipeline Baseline Model

For the first try, we use the original text per speaker in 512 tokens as input to the T5-small model. The T5-small model has 6 layers each in the encoder and decoder. Sub-layers and embeddings have a dimensionality of 512. The dimension of the forward neural network is 2048. It has 8 attention heads. Overall it has around 60 million parameters. We use a dropout of 0.1. Different from the T5 original paper, which uses AdaFactor, we start by using Adam optimizer. We choose a batch size of 1 and 5 epochs. We pick a learning rate of 0.0005 initially. For beam search, we use 12 beams for AMI and 9 for ICSI. The test is conducted on Colab Pro with 25 GB RAM on GPU.



ICSI and AMI ROUGE2 F-score per Epoch

Both AMI and ICSI training losses continue to drop. For AMI, though not linear, ROUGE2 F-score at the meeting level seems to gradually increase, indicating more epochs might be useful. However, the ROUGE2 F-score for ICSI has a slight drop, indicating that the model can be stepping too fast or fails to understand on the meeting level with per-512-token texts.

Additionally, we learn that batch size of 1 is already consuming on average 10G RAM for 512-token texts. To test longer texts, we will not increase the batch size with the compute power we have.

4.3 T5 Tuning Methods

Based on the baseline model, there are few things we experiment to enhance the framework.

Apply heuristic to the dataset: Thought T5 can take any text input, it is still important to send the data that can capture the most for the meeting content. We use different heuristics to the original text (section 4.1.2). The usage of extractive summary requires the BERTSUM model, where we highlight its separate tuning and result in section 4.4 and 5.2.
Pre-trained model size: We experiment both T5-small and T5-base, considering the data size. T5-base has around 220 million parameters with a similar layout as BERT-base. T5-small has around 60 million parameters.
Epochs: We run up to 50 epochs to see if repeating the dataset will help the model to grab more information.
Learning rate: We test 0.001, 0.0005, 0.0001 for the learning rate to avoid too big or too small steps.

4.4 BERTSUM Tuning Methods

For the initial BERTSUM implementation, we configure the model as-is in the paper with a sigmoid classification in the summarization layer. We set all layers to be trainable. Since we develop the model from scratch, we first test on a small dataset with basic predictions. The BinaryCrossEntropy loss function is used.

4.4.1 CNN/DailyMail Dataset

Since there is no extractive summarization benchmark available for our corpus, we verify our BERTSUM model on the CNN/DailyMail dataset if it can reach a similar benchmark.

We used the standard splits of Hermann et al. (2015) for training, validation, and testing, which are 90,266 /1,220/1,093 CNN documents, and 196,961/12,148/10,397 DailyMail documents.

To prevent early corruption of the BERT layers in the training, we use a linear schedule with a warmup of 20 percent of the total training steps, as number of epochs times number of batches. For backpropagation, we use the Adam optimizer and set the learning rate to 0.00005. All BERT layers are trained. During inference, we use trigram blocking (Paulus et al., 2018) to reduce redundant sentences in the output.

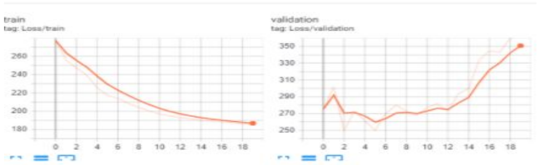
Model	ROUGE-1	ROUGE-2	ROUGE-L
A	43.23	20.22	39.60
B	39.16	16.92	30.03

A: BERTSUM+Classifier (paper authors);
B: BERTSUM+Classifier (us)

A single epoch runs approximately 5.5 hours. We complete only three epochs in total with a batch size of 8, where the original paper trains on 50 epochs with a batch size of 32. Therefore, our scores are expected to be lower.

4.4.2 BERTSUM Baseline

After validating with CNN/DailyMail dataset, we run the initial BERSUM baseline.We notice, due to the small size of our dataset, the model overfits during the training. While the training loss keeps reducing, we find that the validation loss starts going up after a few iterations, where we investigate further.



Loss curves (BERTSUM Baseline)

5. Results and Insights

5.1 Texts with Heuristics to Abstractive

Increasing from 512-token (A) to 1024-token (B) has distinct impacts on the two datasets. While ICSI drops on all ROUGE F-scores, AMI increases on all ROUGE F-scores. The word filtering (C) also has similar results. While ICSI performs worse, AMI corpus has the best ROUGE F-scores in the test set with this heuristic, reaching 36.480 ROUGE1 F-score, 30.570 ROUGEL F-score, and 9.815 ROUGE2 F-score.

ICSI						
	Dev			Test		
Model	rouge1	rougL	rouge2	rouge1	rougL	rouge2
A	26.290	20.220	4.459	24.692	20.595	3.822
B	26.710	12.170	4.215	25.320	21.868	3.643
C	25.860	19.840	3.534	23.746	19.704	2.489
D	25.130	20.510	4.169	**26.822**	**22.271**	**4.649**

AMI						
	Dev			Test		
Model	rouge1	rougeL	rouge2	rouge1	rougeL	rouge2
A	21.620	20.220	7.171	24.074	26.107	6.733
B	23.010	28.640	8.033	25.264	29.325	8.025
C	37.310	31.390	11.810	**36.480**	**30.570**	**9.815**
D	35.360	30.790	10.400	35.688	30.000	9.147

Model A:512 tokens; Model B: 1024 tokens; Model C: B + filtering words; Model D: C + T5-base;

We think the reason for the two datasets reacting so differently is by their nature. AMI corpus contains role-play meetings. Therefore, after words filtering (C), it has a clearer structure of discussion based on design. Meanwhile, ICSI has actual meetings, which tend to flow less structurally. In which case, adding more texts (B) or filtering words (C) may not necessarily help with the context. This is also why AMI has on average much higher ROUGE F-scores than ICSI, which is also seen by other related papers (Shang et al.,2018; Zhu et al.,2020).

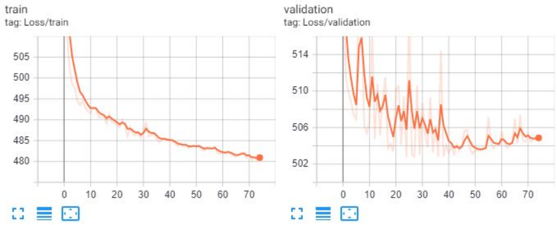
Though not written in the table, increasing the number of epochs helps with both datasets. The table uses the best dev epoch results, which ranges from epoch 9 to 34. Further than epoch 35 has a minimal impact on the ROUGE results, but we have seen Zhu et al.(2020) using 50 epochs for the development.

We also test two different model sizes - T5-base (D) and T5-small. Surprisingly, T5-base (D) has the best result in the test set for ICSI. Considering such a small dataset, we thought a model with 220 millions parameters might not be helpful. It will be a good next step to investigate even larger model sizes.

5.2 Extractive Summary to Abstractive

5.2.1 BERTSUM Results

To fix the overfitting problem, we test freezing all BERT layers and stacked three linear layers of dimensions 300, 300 and 100 using Relu activation at each layer with a final sigmoid (model two). This produces better loss curves and very close ROUGE F-scores to the BERTSUM baseline. Thus, we choose this model moving forward.



Loss curves (Model Two)

Different from the baseline, we remove the trigram blocking technique (Paulus et al., 2018), since it is more applicable for the CNN/DailyMail datasets where repeatable sentences are frequent. However, we completely get rid of sentences of three words or less from the output as these sentences typically produce noisy results.

	AMI			ICSI		
Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
1	0.61	0.38	0.59	0.69	0.39	0.57
2	0.6	0.38	0.59	0.65	0.38	0.57

Model 1 :BERTSUM+Classifier (all layers trained); Model 2: BERTSUM + 3 linear layers of dimensions 300, 300 and 100 with Relu activation

5.2.2 BERTSUM and T5 Results

ICSI						
	Dev			Test		
Model	rouge1	rougL	rouge2	rouge1	rougL	rouge2
E	27.660	19.950	4.287	22.837	19.705	2.881
F	**30.510**	**23.380**	**4.874**	23.666	20.575	2.523
G	28.040	19.810	4.569	24.506	19.795	1.906

AMI						
	Dev			Test		
Model	rouge1	rougeL	rouge2	rouge1	rougeL	rouge2
E	**39.480**	31.540	12.260	28.803	24.076	6.907
F	38.250	**31.730**	**12.710**	25.959	20.641	5.811
G	36.930	31.250	11.830	31.107	24.338	7.398

Model E: 1024 tokens + word filtering + BERTSUM + 0.0005 Learning Rate (LR);
Model F: E but 0.0001 LR; Model G: E but 0.001 LR

Changing from original texts to extractive summaries as a pre-step outperforms other heuristics on both ICSI and AMI dev datasets. Especially with a small learning rate, ROUGE 2 F-score reaches 12.710 on the AMI dev set and 4.874 in the ICSI dev dataset. This aligns with our expectation that with more succinct meeting content, T5 can generate better per-meeting level abstractive summary. However, it doesn't work well with unseen data. In fact, its ROUGE can be lower than our baseline (A). We think the major reason is the pipeline integration of BERTSUM and T5. For a strong test result, it would require a good extractive summary from BERTSUM and a strong abstractive summary on the unseen dataset from T5. How to make these two models work better together can be a future step to explore. Currently, we just pass through the results from BERTSUM into T5 for abstractive summarization.

6. Conclusion

In this project we experiment with T5 along with the BERTSUM model to perform an abstractive summary of the meeting transcripts from ICSI and AMI datasets. We achieve ROUGE2 F-scores of 9.815 on the AMI test set and 4.649 for ICSI test set. Though the result doesn't reach comparable scores from other papers, we do find T5 performs better with more succinct text inputs. With the challenge of long transcripts, using BERTSUM as a pre-step or conducting word filtering can help with the summarization. For future study, it will be interesting to discover a better way to incorporate BERTSUM results into the T5 inputs, such that the pipeline is more suitable for unseen data.

7. References

Devlin, Jacob, et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” *ArXiv.org*, 24 May 2019, arxiv.org/abs/1810.04805.

Hermann, Karl Moritz, et al. “Teaching Machines to Read and Comprehend.” *ArXiv.org*, 19 Nov. 2015, arxiv.org/abs/1506.03340.

Janin, Baron, Edwards, Ellis, Gelbart, Morgan, Peskin, Pfau, Shriberg, Stolcke, et al. 2003. The icsi meeting corpus. *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03)*,, 1:I–I.

Li, Manling, et al. “Keep Meeting Summaries on Topic: Abstractive Multi-Modal Meeting Summarization.” *ACL Anthology*, 2019, www.aclweb.org/anthology/P19-1210/.

Liu, Yang. “Fine-Tune BERT for Extractive Summarization.” *ArXiv.org*, 5 Sept. 2019, arxiv.org/abs/1903.10318.

McCowan, Carletta, Kraaij, Ashby, Bourban, Flynn, Guillemot, Hain, Kadlec, Karaiskos, et al. 2005. The ami meeting corpus. *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*,, 88:100.

Qin, Kechen, et al. “Joint Modeling of Content and Discourse Relations in Dialogues.” *ArXiv.org*, 14 May 2017, arxiv.org/abs/1705.05039.

Raffel, Colin, et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” *ArXiv.org*, 28 July 2020, arxiv.org/abs/1910.10683.

Romain Paulus, et al.”A deep reinforced model for abstractive summarization” *ArXiv.org*, 13 Nov. 2017, https://arxiv.org/abs/1705.04304

See, Abigail, et al. “Get To The Point: Summarization with Pointer-Generator Networks.” *ArXiv.org*, 25 Apr. 2017, arxiv.org/abs/1704.04368.

Shang, Guokan, et al. “Unsupervised Abstractive Meeting Summarization with Multi-Sentence Compression and Budgeted Submodular Maximization.” *ArXiv.org*, 14 Nov. 2018, arxiv.org/abs/1805.05271.

Vaswani, Ashish, et al. “Attention Is All You Need.” *ArXiv.org*, 6 Dec. 2017, arxiv.org/abs/1706.03762.