WCN — Weighted Contextual N-gram method for evaluation of Text Summarization

Aditya Shah

November 14, 2021

1 Introduction

Summarization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content. Since manual text summarization is time expensive and generally laborious, the automatization of summarization is gaining increasing popularity and therefore constitutes a strong motivation for academic research.

There are important applications for text summarization in various NLP related tasks such as question answering, legal texts summarization, news summarization, and headline generation. Moreover, the generation of summaries can be integrated into these systems as an intermediate stage which helps to reduce the length of the document.

In general, there are two different approaches for automatic summarization: **extraction and abstraction**

The extractive approach: Extractive summarization picks up sentences directly from the document based on a scoring function to form a coherent summary. This method work by identifying important sections of the text by cropping out and stitching together portions of the content to produce a condensed version.

Abstractive summarization Abstractive summarization methods aim at producing summary by interpreting the text to understand the semantic meaning and then generate a new shorter text — parts of which may not appear as part of the original document. The summary conveys the most critical information from the original text, requiring rephrasing sentences such as a human-written abstract usually does.

In this assignment, I use a T5 model to produce an extractive summary. The model is fine-tuned on Extreme Summarization (XSum) Dataset. Further, I briefly discuss the drawbacks of existing n gram and contextual based evaluation metrics. Additionally, I also propose ***Weighted Contextual N gram*** **(WCN)** method as an alternate evaluation metric for text summarization.

2 Objective

Formally, given a piece of text T with tokens: $t_1, t_2, t_3, ...t_n$, our goal is to produce an abstractive summary S with tokens $s_1, s_2, s_3, ...s_k$ where $k << n$. I use T5 model in order to generate abstractive summary from the given input text.

3 Data Preprocessing

For training, we need an input sequence and a corresponding target sequence. The input sequence is fed to the model using input_ids generated by the T5 tokenizer's encode function. Next for target sequence, we use the input_ids of the gold summary S.

Since T5 is text-to-text encoder-decoder model so for each task, we specify a task prefix which is prepended in the input text. For summarization, we use the prefix: *summarize:*.

Given the input text:

Fast forward about 20 years, and it's fair to say he has done just that. The business he runs, Frasers Hospitality, is one of the world's biggest providers of high-end serviced ...

We prepend the task prefix so we have:

summarize: Fast forward about 20 years, and it's fair to say he has done just that. The business he runs, Frasers Hospitality, is one of the world's biggest providers of high-end serviced ...

This input text is converted into tokens and then input_ids are generated using T5 tokenizer. Likewise for the given gold summary, we generate input_ids and use them as target sequence input_ids.

In addition, we must make sure that padding token id's of the target sequence are not taken into account by the loss function. In PyTorch, this can be done by replacing them with -100, as that is the ignore_index of the CrossEntropyLoss.

Additionally in the given dataset, there are few instances where the output summary is simply a '.' or just one or two words. So if the no. of words in the output summary are less than 3, then we simply discard those data pairs.

4 Model Architecture

T5 model uses standard sort of vanilla encoder - decoder architecture proposed in the original transformer paper.

Figure 1 shows the overview of the T5 model.

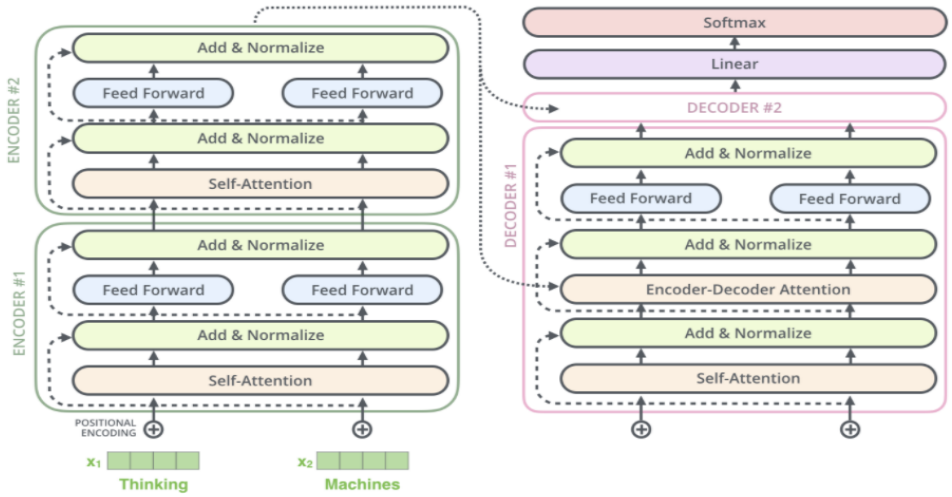

Figure 1: Overview of T5 model[1]

I use the pretrained T5 model (t5-small) from the hugging face library. During training phase, the input to the model are the *input_ids* and *attention_mask* obtained from the input text, as well as the labels i.e *input_ids* of the gold summary. All of these are generated using the T5 tokenizer. These are given as input to the model and the model returns dictionary of parameters including loss. I use this language modeling loss to calculate the gradients and optimize the model.

During inference, I use the model.generate() method which returns the token ids of the words generated by the trained model. This method takes care of encoding the input and feeding the encoded hidden states via cross-attention layers to the decoder and auto-regressively generates the decoder output. The generated token ids are decoded (mapped to actual words) by the tokenizer to return the generated summary sentence. Next, I compute the rouge score based on this generated summary and ground truth summary for every sentence. Finally I take the average of the rouge_2 score i.e of f, p, r (f1 score, precision, and recall) values for all the sentences.

I use the following set of hyper parameters:

- batch_size: 8
- epochs: 3
- learning_rate: 2e-4
- clip: 5
- input_length: 512
- output_length: 100

[1]From: Jay Alammar's blog

Here input length is the maximum length of the input text. Output length is the maximum length of the generated summary. I use gradient clipping with threshold clip value as 5 to prevent exploding gradients.

5 Results

For the evaluation we use the Rouge_2 score on validation and test data. Rouge score returns the f, p, r values for every sentence. So we report the average of the f, p, r values for all the sentences in validation and test data.

Given table shows the performance of the model on the validation and test data.

Table 1: Results

Mode	F1-score	Precision	Recall
Val	9.73%	8.35%	12.2%
Test	9.5%	8.16%	11.93%

Here are few examples of summaries generated by the model:

Gold Summary: `Bournemouth striker Callum Wilson said it was "great to feel like a football player again" as he returned to action following a long-term injury.`

Generated Summary: `Bournemouth striker Ryan Wilson says he feels like a footballer again after`

Here we can see that the generated summary tries to cover the semantic meaning when compared to the ground truth summary. Although it does not include all the information like the player getting injured but still does a good job of preserving the context.

Here's one more example:

Gold Summary: `The collapsed retailer BHS is to re-launch as an online shop, selling some of the most popular items previously available on its website.`

Generated Summary: `A new online shop for the former BHS department store is to be sold to a new`

Here, although the generated summary includes words from the ground truth summary, but it fails to preserve the contextual meaning of the summary. The literal meaning of the text does not align well with the gold summary. We analyze this in the next section.

6 Analysis

For some input text, our model generates the perfect summary. for eg:

Input Text: `summarize: If you have a picture you would like to share, please see below the images for details on how to submit yours. If you have a picture you'd like to share, email us at england@bbc.co.uk, post it on Facebook or tweet it to @BBCEngland. You can also find us on Instagram - use englandsbigpicture to share an image there. When emailing pictures, please make sure you include the following information: Please note that whilst we welcome all your pictures, we are more likely to use those which have been taken in the past week. If you submit a picture, you do so in accordance with the BBC's Terms and Conditions. In contributing to England's Big Picture you agree to grant us a royalty-free, non-exclusive licence to publish and otherwise use the material in any way that we want, and in any media worldwide. It's important to note, however, that you still own the copyright to everything you contribute to England's Big Picture, and that if your image is accepted, we will publish your name alongside. The BBC cannot guarantee that all pictures will be used and we reserve the right to edit your comments. At no time should you endanger yourself or others, take any unnecessary risks or infringe any laws collecting any kind of media.`

Gold Summary: `Each day we feature a photograph sent in from across England - the gallery will grow during the week`

Generated Summary: `Each day we feature a photograph sent in from across England - the gallery will grow during`

Here the generated summary completely aligns with the gold summary and thus has rouge_2 value as 1. Likewise there are examples where the model generates poor or 0 f values.

So, I analyse the rouge_2 f scores of all generated summaries from test data. Next, I plot the count of the summaries along with their rouge_2 f scores. The figure 2 gives an overview of the analyses performed:

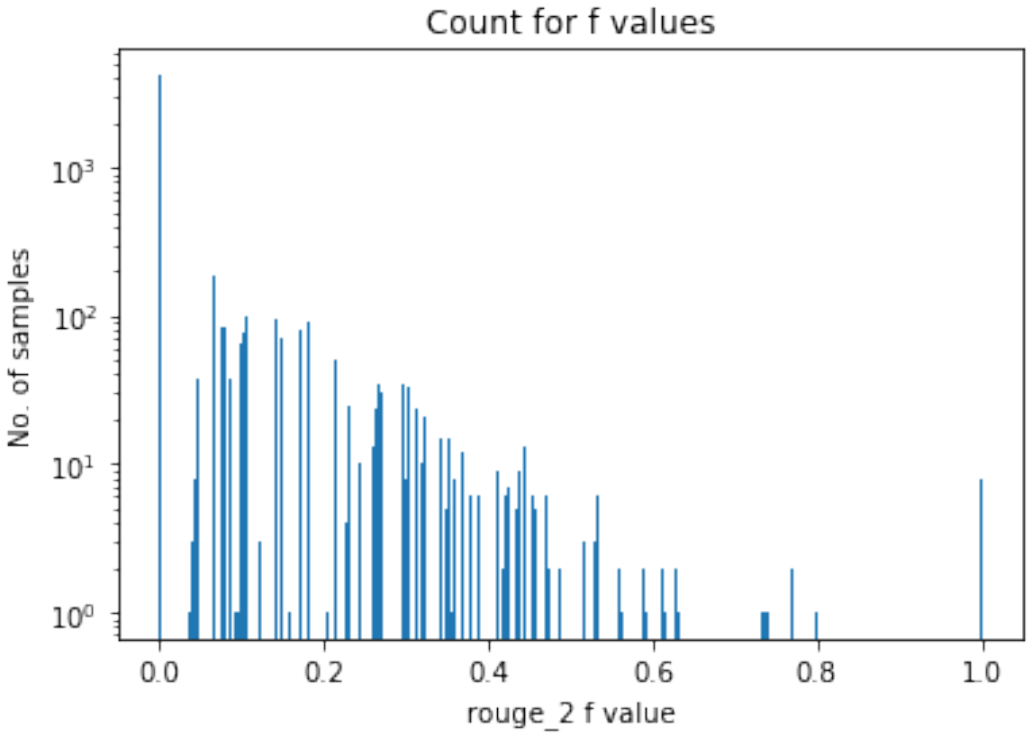

Figure 2: Analysis of summaries based on the rouge_2 f scores

The above figure shows that more than 1000 (4143 summaries) have rouge_2 f score of 0.0. On the other hand about 12 summaries have perfect rouge_2 f score. There are about 100s to 1000s of summaries in total which have f scores ranging from 5% to 70% while only few summaries (about 30) have f score above 70%

The following table shows the exact count of summaries within the mentioned f score range:

Table 2: No of summaries with f scores on Test data

No. of summaries	F1-score range
4143	0.0
19	2% - 4%
2693	4% - 8%
1873	8% - 15%
1893	15% - 30%
621	30% - 50%
64	50% - 70%
15	70% - 90%
12	90% - 100%

The above table gives a good insight of number of summaries and their f score range. Having a high count (4143/11,333) for 0 f scores indicates that the model is unable to generate the required summary for almost 30% of the test data . On the other hand, over 2500 summaries have more than 15% f score.

Drawback: Simply judging the model based on the Rouge_2 score might not be the best idea. Since rouge_2 score using n grams, it looks for exact word matching between the gold summary and the predicted summary. For instance consider the following example:

Gold Summary: `Such a beautiful weather for trekking`

Generated Summary: `Perfect day for hiking`

Here if we calculate the rouge 2 score, then the value would be 0. However if we analyse both the summaries, we know that the contextual meaning of the summaries are similar. Hence it should be assigned a higher value instead of 0 as the generated summary semantically aligns with the gold summary. This a major drawback of n gram based metrics. We instead need a semantic based metric for evaluation of text generation tasks

Once such metric is BERTscore[1] which uses word based contextual similarity as an evaluation metrics. It considers Bert based word embeddings of every token in both the summaries, then calculates cosine similarity and finally takes weighted average based on IDF (inverse document frequency).

I tested BERT score for the above example and it gives f score of 0.91 or 91% which intuitively makes sense as the predicted summary is contextually same as gold summary.

To further analyse this, consider the following example of generated summary from test dataset:

Gold Summary: `A bill giving limited access to abortion has been referred to an advisory body by Ireland's president`

Generated Summary: `The President of the Republic of Ireland is to ask the Supreme Court to consider a bill that`

Here the rouge_2 score is 0 as there's no bigram words that matches in the ground summary. However if we read the text, then we understand that the summary talks about new bill by the president and since there's a mention of bill in the generated summary, so it contextually aligns little bit with the gold summary and hence it should be assigned some score.

Evaluating this on BERTscore gives us f score of 88% which is again too high considering the entire sentence. The below Table 3 shows the BERT score when evaluated on the complete test dataset:

Table 3: BERTscore and Rouge on Test Dataset

Metric	F1-score	Precision	Recall
Rouge	9.5%	8.16%	11.93%
BERTScore	88.37%	89.32%	87.45%

As we can see from the above table, although BERTscore uses contextual approach to assign f scores, it sometimes overscores the summaries.

WCN: BERTscore captures the semantic or contextual meaning underlying in the summary while the ngram based metrics capture the word order and exact presence of the words. Both of them are important in order to effectively judge a summary

So I propose ***Weighted Contextual N gram*** **(WCN)** as a method to evaluate generated texts. Instead off simply relying on n gram or contextual based evaluation metrics we can compute the *weighted average of the rouge_1 score, rouge_2 score, and BERTscore* as a final metric. This metric will thus preserve the contextual as well as word order which leads to a better evaluation judgement.

Lets consider the same previous example:

Gold Summary: `A bill giving limited access to abortion has been referred to an advisory body by Ireland's president`

Generated Summary: `The President of the Republic of Ireland is to ask the Supreme Court to consider a bill that`

Here rouge_1 score is 0.129, rouge_2 score is 0, and BERTscore is 88%. Using the average of above scores will result in final score of 33.6% which seems fair for the above generated summary.

Following table 4 shows the result of WCN metric on the test dataset as well as final comparison of all the metrics.

Table 4: BERTscore and Rouge on Test Dataset

Metric	F1-score	Precision	Recall
Rouge_1	34.1%	36.93%	32.63%
Rouge_2	9.5%	8.16%	11.93%
BERTScore	88.37%	89.32%	87.45%
WCN	35.79%	37.11%	35.22%

WCN method overcomes the drawbacks of rouge_1 and rouge_2 scores which are solely based on exact word matching and word order. By incorporating contextual meaning through BERTscore, WCN becomes a more robust method to evaluate text generation based tasks. However, designing a single metric that can truly evaluate text generation tasks with human level precision is still an open research problem.

7 Conclusion

Through this assignment, I fine tune T5 model on Extreme Summarization (XSum) Dataset achieveing a **rouge_2 f score of 9.5%** on test data. Further I discuss the drawbacks of ngram based metrics as well as contextual word metrics. Finally, I propose use of **WCN** – an alternative metric which can be more effective for evaluation of text generation tasks

References

[1] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.