Transformers For Markdown Article Rewriting

Stanford CS224N Custom Project

Scott Hickmann
Department of Computer Science
Stanford University
hickmann@stanford.edu

Abstract

The goal of this project is to be able to summarize and paraphrase full article, which may contain inline links, text in bold, images, etc. Inline markdown can be challenging to pass to transformers. In this paper, I describe how I am to rewrite articles containing rich text including inline markdown properly. More specifically, this paper delves into fine-tuning transformer models pretrained on summarization or paraphrasing by forcing them to learn markdown syntax, and encoding rich text tags to reduce the amount of training data necessary. I then introduce a novel method based on ROUGE and cosine similarity of words inside markdown tags to evaluate the quality of an NLP model at producing proper rich text.

1 Key Information to include

- Mentor: Manan Rai
- External collaborators: No
- Sharing project: No

2 Introduction

As of 2022, the state of the art approach when it comes to summarizing and paraphrasing text is to use transformer-based encoder-decoder models. Indeed, T5 and BART models have proven very performant at these tasks when pretrained on corpora of summarized and paraphrased text.

Unfortunately, these great pretrained models were trained on text that was properly cleaned, devoid of any form of style and formatting. Hence trying to apply these models to paraphrase articles containing rich text might bring up such issues:

- "A flock of ==frogs== were ==roaming== around the park in search of water once more." becomes "A flock of ==frogs===roaming===roaming===roaming====roaming====roaming====roaming..."
- "Once, a group of **frogs** were **roaming** around the forest in search of water." becomes "A flock of **frogs** were **roaming** around the park in search of water once more." whereas the same sentence without the markdown "**" syntax would become "A herd of frogs were wandering around the woods in search of water".

Yet summarizing and rewriting articles is key today in a world where everyone is bombarded with information. Markdown is helpful to make an article more digest, and therefore it is important to preserve it while paraphrasing and summarizing text. Hence the motivation behind this paper.

3 Related Work

This problem has already been encountered in the past, but no good solution has yet been determined. People like Edward Ross have experimented with several approaches using HTML tags to parse markdown, but these approaches have remained inconclusive [1]. However, in recent years, transformer models have proven very successful not only at doing various tasks, but also at learning quickly through fine-tuning thanks to multi-head attention [2]. Even with a limited amount of data, this paper will demonstrate how it is possible to summarize and paraphrase text containing inline markdown correctly thanks to these transformers.

4 Approach

The pretrained T5 transformer model used for paraphrasing can be found here [3].
The pretrained BART transformer model used for summarization can be found here [4].

4.1 Scraping Data

4.1.1 Paraphrasing

Using an simple web parser, I am collecting markdown rich sentences from the web. The scraping algorithm first goes through several Medium articles (converted from HTML to markdown) and downloads those. Then for each article, it selects a few sentences that are rich in markdown (contain at least one piece of markdown syntax in them). All these sentences are then stored in a `.csv` file [5].

4.1.2 Summarization

Using an simple web parser, I am collecting articles from the web. The scraping algorithm first goes through several Medium articles (converted from HTML to markdown) and downloads those. Then for each article, it groups sentences together into several parts. All these parts are then stored in a `.csv` file [5].

4.2 Preparing Data

4.2.1 Paraphrasing

For each sentence, the data preparation algorithm stores a copy of the sentence without any markdown. It then encodes these sentences and runs them through the T5 transformer model pretrained on paraphrasing diverse text but not yet fine-tuned. Then, a human (in this case me) is shown two sentence: the original sentence with markdown, and the paraphrased sentence without markdown. My task is simply to add the markdown from the original sentence back on the paraphrased sentence without markdown. For example, if the algorithm gives me:

- Input with markdown (fine-tune source): “Once, a group of **frogs** were **roaming** around the forest in search of water.”
- Output without markdown: “A herd of frogs were wandering around the woods in search of water”
- Human (fine-tune target): “A herd of **frogs** were **wandering** around the woods in search of water”

4.2.2 Summarization

Similarly to paraphrasing, for each part, the data preparation algorithm stores a copy of the part without any markdown. It then encodes these parts and runs them through the BART transformer model pretrained on summarizing diverse text but not yet fine-tuned. Then, a human (in this case me) is shown two parts: the original part with markdown, and the summarized part without markdown. My task is simply to add the markdown from the original sentence back on the paraphrased part without markdown.

4.3 Fine-Tuning

Now that we have collected paraphrased and summarized markdown data in an efficient way by using a hybrid version between automated and manual paraphrase, we can now move on to fine-tuning the transformer models. This is as simple as simply feeding in the input sentences and parts and back propagating the model's loss functions calculated by comparing the predicted and stored paraphrases and summaries.

4.4 Improvements

While the above approach works fine, there are many different markdown tags, and some work better than others with T5 and BART. In addition, we need enough fine-tuning data for every single markdown syntax ("*", "**", "_", links, etc.) for the models to perform well. Using HTML instead of markdown barely helps, even when defining the tags as special tokens. There is a trick to only train the model on a single tag. First, we can convert each markdown symbol into a special token with a number attached, such as "(MD1)", "(MD2)", etc. This way, the models only need to learn to preserve this single token in the correct markdown rules, instead of all the various markdown tags. I wrote an encoder and decoder to convert markdown to these special tokens and back. Throughout this paper, we will refer to text encoded in this manner as "MD-encoded" and the tags as (MDn) tags.

There is one edge case with this method: links. Link labels are kept in the text and surrounded by (MDn) tags just like everything else, but the URL itself is removed and can be added back after the summarization/paraphrasing by looking up the proper (MDn) tag in the output. This is done in order to prevent the model from being confused about "?", ".", and other special character in URLs that can for example trick the model into believing a sentence is over when it isn't.

5 Experiments

5.1 Data

The parts of articles, and sentences used to fine-tune the models were scraped from Medium articles. The dataset I collected can be found here [5].

5.2 Evaluation method

Using raw markdown articles that have been reserved for evaluation, we divide it into m subparts that are each get summarized or paraphrased individually.

We remove all markdown and feed that through the original model without any fine-tuning (text 1), and feed the markdown encoded version as discussed in 4.4 through the final model with fine-tuning then remove the markdown (text 2). We then compute ROUGE 2 and ROUGE L between these two texts. Let's call these scores R and L respectively. These scores evaluate how well the original summaries and paraphrases are preserved after fine-tuning, but not how well the markdown syntax is [6].

Therefore, let's compute a score to evaluate the success of the markdown. As there is no standard metric for that, I developed and used the following custom metric. Let's define the functions f, which computes an enclosure check (makes sure that the output is surrounded by (MD0) tags which should be present based on how the MD encoder and decoder work), and g, which computes sentence similarity between two phrases by taking the average of the word embeddings of all words in the two phrases, and calculating the cosine similarity between the resulting embeddings [7]:

$$f(\boldsymbol{v}) = \begin{cases} 1 & \text{if } \boldsymbol{v} = \text{“(MD0)\{entire content\}(MD0)”} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$g(\boldsymbol{v}, \boldsymbol{w}) = \frac{\text{avg}(\boldsymbol{v}) \cdot \text{avg}(\boldsymbol{w})}{\|\text{avg}(\boldsymbol{v})\| \|\text{avg}(\boldsymbol{w})\|} \tag{2}$$

Using these two functions, let's now define M, the markdown similarity score between the original text and the paraphrased/summarized text. Let's create $\boldsymbol{v}$, a 3D array containing the markdown tag contents for each (MDn) tag pair for each defined n for each paraphrased/summarized MD-encoded text. Let's also create $\boldsymbol{w}$, a 3D array containing the markdown tag contents for each (MDn) tag pair for each defined n for each original MD-encoded text (before paraphrasing/summarizing). M can be calculated in the following manner:

$$M = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{\text{len}(\boldsymbol{v}_i)}\left(f(\boldsymbol{v}_{i,1,1}) + \sum_{j=2}^{\text{len}(\boldsymbol{v}_i)}\left(\frac{1}{\text{len}(\boldsymbol{v}_{i,j})}\sum_{k=1}^{\text{len}(\boldsymbol{v}_{i,j})}\max\{g(\boldsymbol{v}_{i,j,k}, \boldsymbol{w}_{i,j,l}) : l = 1..\text{len}(w_{i,j})\}\right)\right)\right) \tag{3}$$

Finally, we can combine the resulting scores to get an overall score S in the following manner:

$$S = \frac{R + L + 2M}{4} \tag{4}$$

This is the overall score that will be reported in the results of the experiments.

5.3 Experimental details

Here are the hyperparameters used for each task that experimentally created the best results.

5.3.1 Paraphrasing

Learning Rate	1×10^{-4}
Train & Val Batch Size	2
Train Epochs	5
Max Length	128
Num Beams	15
Diversity Penalty	0.99

5.3.2 Summarization

Learning Rate	1×10^{-4}
Train & Val Batch Size	2
Train Epochs	3
Min Length	0
Max Length	512
Summary Max Length	142
Num Beams	4
Length Penalty	0.1

5.4 Results

The baseline charts have been obtained by using the non-fine-tuned model, while the best charts have been obtained using the fine-tuned model. You can find the corresponding tables in Appendix and articles 1 through 10 in Appendix B.

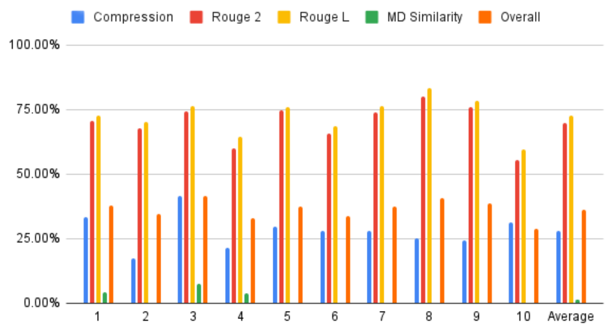

Figure 1: Summary Baseline

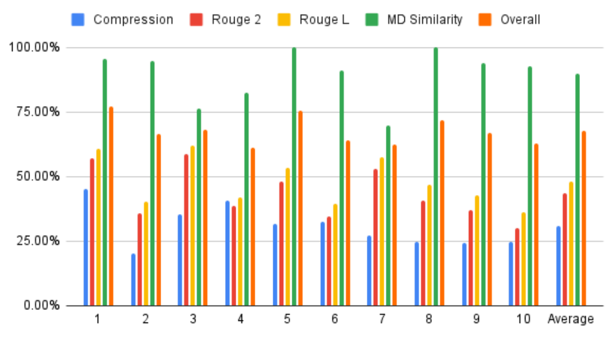

Figure 2: Summary Best

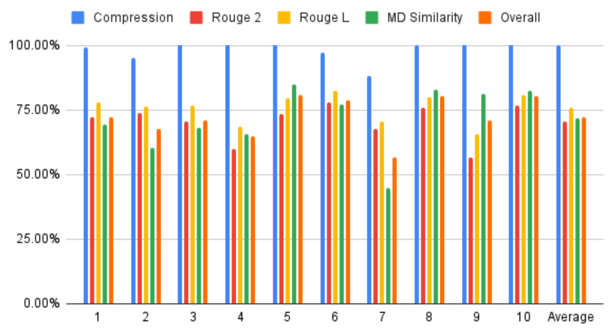

Figure 3: Paraphrase Baseline

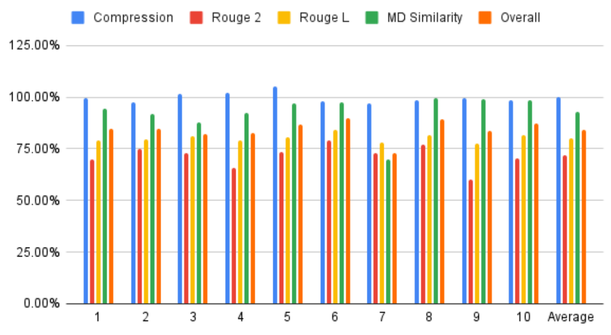

Figure 4: Paraphrase Best

The results are for the most part what I expected. By fine-tuning the model, we are able to significantly improve the MD similarity score. Markdown syntax is generally well preserved. However, for summarization specifically, the ROUGE scores decrease, which indicates that the model has a hard time preserving the same summaries that it had when not using markdown. From human evaluation, the summaries are still perfectly understandable so we do not consider that a big problem, apart from the fact that they differ slightly from the original summaries. This does not seem to be as much of an issue for paraphrasing, where ROUGE scores are high.

As a reminder, please note that these ROUGE score do not compare the model's output text to human generated text, but rather to machine generated text while omitting markdown. Also note that to collect the previous results, (MDn) tags where added as additional special tokens for the BART summarizing model, but not to the T5 paraphrasing model. The reason for why adding them as special tokens improves BART's performance but not T5's performance remains to be explained.

5.5 End-to-end

With the paraphraser and summarizer models fine-tuned, we can combine both of these to achieve a more developed form of article rewriting. Here are the steps used to fully rewrite an article.

- Divide the article into subparts, categorizing each of these parts into paragraph vs. other (image, code, figures, etc.).
- MD-encode the markdown in each paragraph subparts to (MDn) tags.
- Pass these MD-encoded parts into the BART fine-tuned summarizer model.
- Pass these results from BART into the T5 fine-tuned paraphrasing model.
- MD-decode the markdown in the resulting text from (MDn) tags to actual markdown.
- Combine all subparts back into a single document.

Examples of this method are available in Appendix B.

6 Analysis

The model fails most often when there is a lot of nested markdown, which is particularly present at the footer of articles when referencing the author(s). We might want to add more fine-tuning data regarding footer text to address this issue.

Overall, the results found are very promising considering the scarcity of data (manually collected) to achieve proper article rewriting. Collecting more data would definitely improve the performance especially when it comes to the ROUGE score of summarization, which is an inherently harder task than paraphrasing text.

7 Conclusion

The methods developed in this paper can be used to simplify or generate new and unique article rewrites. Collecting more data could help mitigate the slight decrease in ROUGE performance to achieve top-grade article rewriting.

As an avenue for future work, fine-tuning BART for too many epochs results in the model producing only (MD0), possibly by interpreting that token as an end token rather than a markdown token. It could be worth investigating why that is, as it could be a way to improve BART's performance at preserving markdown syntax, which is worse that T5's paraphrasing performance.

References

[1] Edward Ross. Using HTML in NLP. Jul 2020.

[2] Ashishm Vaswani, Noam Shazeer, Niki Parmar, Jacob Uszkoreit, Llion Jones, Aidan N. Gomez, and Łukasz Kaiser. Attention Is All You Need. 2017.

[3] Ramsri Goutham. High-quality sentence paraphraser using Transformers in NLP. Sep 2021.

[4] Sam Shleifer. Distilbart CNN 6-6. Sep 2021.

[5] Scott Hickmann. Metrics and Datasets. Feb 2022.

[6] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. 2004.

[7] Yves Peirsman. Comparing Sentence Similarity Methods. May 2018.

[8] Scott Hickmann. Results. Mar 2022.

A Appendix

Here are the results in table rather than graph form. The articles can be found in Appendix B.

Article Number	Compression	Rouge 2	Rouge L	MD Similarity	Overall
1	33.27%	70.65%	72.64%	4.17%	37.91%
2	17.55%	67.58%	70.05%	0.00%	34.41%
3	41.54%	74.50%	76.55%	7.69%	41.61%
4	21.41%	59.79%	64.29%	4.00%	33.02%
5	29.85%	74.56%	75.88%	0.00%	37.61%
6	27.90%	65.91%	68.70%	0.00%	33.65%
7	28.11%	74.02%	76.20%	0.00%	37.55%
8	25.32%	80.00%	83.12%	0.00%	40.78%
9	24.33%	75.99%	78.51%	0.00%	38.62%
10	31.17%	55.31%	59.61%	0.00%	28.73%
Average	**28.04%**	**69.83%**	**72.56%**	**1.59%**	**36.39%**

Table 1: Summary Baseline

Article Number	Compression	Rouge 2	Rouge L	MD Similarity	Overall
1	45.22%	57.21%	60.63%	95.61%	77.27%
2	20.37%	35.91%	40.28%	94.74%	66.42%
3	35.30%	58.91%	62.01%	76.19%	68.32%
4	40.82%	38.74%	41.93%	82.42%	61.38%
5	31.83%	48.27%	53.62%	100.00%	75.47%
6	32.57%	34.69%	39.69%	91.23%	64.21%
7	27.05%	53.13%	57.37%	70.00%	62.63%
8	24.84%	40.66%	46.95%	100.00%	71.90%
9	24.43%	36.91%	42.86%	94.12%	67.00%
10	24.55%	29.88%	36.41%	92.86%	63.00%
Average	**30.70%**	**43.43%**	**48.18%**	**89.72%**	**67.76%**

Table 2: Summary Best

Article Number	Compression	Rouge 2	Rouge L	MD Similarity	Overall
1	99.44%	72.37%	78.06%	69.52%	72.37%
2	95.16%	73.90%	76.39%	60.31%	67.73%
3	102.02%	70.73%	76.58%	68.35%	71.00%
4	103.71%	60.14%	68.42%	65.67%	64.97%
5	110.27%	73.54%	79.71%	84.85%	80.74%
6	97.26%	77.84%	82.48%	77.34%	78.75%
7	88.12%	67.61%	70.50%	44.73%	56.89%
8	99.98%	76.01%	80.18%	82.79%	80.44%
9	103.91%	56.65%	65.62%	81.32%	71.23%
10	101.38%	76.58%	80.79%	82.35%	80.52%
Average	**100.12%**	**70.53%**	**75.87%**	**71.72%**	**72.46%**

Table 3: Paraphrase Baseline

Article Number	**Compression**	**Rouge 2**	**Rouge L**	**MD Similarity**	**Overall**
1	99.64%	70.06%	79.23%	94.64%	84.64%
2	97.24%	75.03%	79.46%	91.97%	84.61%
3	101.75%	73.05%	80.87%	87.77%	82.36%
4	102.04%	65.73%	78.92%	92.52%	82.42%
5	105.22%	73.20%	80.75%	96.97%	86.97%
6	98.12%	78.98%	84.14%	97.66%	89.61%
7	97.19%	73.03%	78.17%	70.02%	72.81%
8	98.73%	77.15%	81.39%	99.73%	89.50%
9	99.71%	60.12%	77.63%	98.90%	83.89%
10	98.40%	70.34%	81.70%	98.47%	87.25%
Average	**99.80%**	**71.67%**	**80.22%**	**92.87%**	**84.41%**

Table 4: Paraphrase Best

B Appendix

The 10 original articles used to evaluate the model can be found here [5]. Their summarized only, paraphrased only, and summarized-then-paraphrased/end-to-end versions can be found here [8] under summarizer, paraphraser, and rewriter respectively.