arXiv Comment Metadata Prediction

Brandon Edmunds
Computer Science and Engineering
The Pennsylvania State University
Pennsylvania, USA
bie5065@psu.edu

***Abstract*—arXiv paper comments are predicted using arXiv paper abstracts. A T5 transformer is used to make the predictions. The ROUGE metric will be used to evaluate performance.**

Keywords—transformers, arXiv, text generation

I. Introduction

arXiv papers are all stored with metadata on the arXiv site. The metadata contains information such as who the authors are, the abstract of the paper, and some additional comments about the paper. The paper comments often include information such as the number of pages in the paper, the number of figures, where the paper was published, and changes between versions of the paper. The task to be analyzed is to use text generation to predict paper comments from the paper abstract.

The task presents multiple challenges. The first challenge is the model must learn the type of content that will be present in comments given an abstract. For example, some comments that have both a page number count and a figure count are formatted as "37 pages, 15 figures; published version", whereas other comments may be formatted as "submitted to ApJ. Comments are welcome". The model will need to learn the type of content a comment has based on information such as the way the abstract was written and what the abstract was about. Another challenge is that once the model identifies the type of content of the comment, the model must determine some relevant values for the comment, such as the number of figures and the journal the paper was submitted to. Clearly, the task is specific to the data set and tries to solve novel problems. For example, determining the number of figures in a paper from a paper abstract has not been previously studied. To predict the number of figures, the model will need to use the abstract to make decisions such as the amount of experimentation that the abstract suggests the paper will have, and whether the type of experimentation will yield figures. To determine the type of comment, the model will need to learn to make some conclusions about the authors based on the way the abstract was written in order to predict how the authors will make comments. Another connection could be that the content of the abstract may make the journal the paper was published in clearer, and there may be some connection between the journal in which a paper was published to the comment.

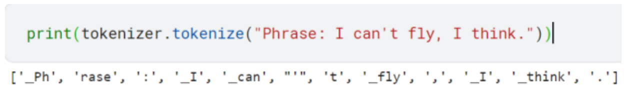
Also, the task is useful for a number of reasons. Firstly, it may not be the case that a reader of a paper has access to metadata about the paper, and someone browsing papers may come across the abstract of a paper, but not the metadata or other information about the paper (such as the journal a paper was published in). The user of interest may have access to additional information, such as a title, or authors, but copy-pasting only the abstract into a textbox, for example, and not worrying about other information could be easier. Such a user may want to know additional information about the paper from the abstract, such as the length of the paper, whether or not the paper was published, how many versions of the paper there are, what changed between versions of paper, and other information found in the comments. Additionally, the success of the model in completing the task could provide additional insights into how much information about the authors can be gathered from a paper abstract, since the content of a comment is partially dependent on the choices of the authors. Similarly, insights relating to how much information abstracts provide about papers (such as page length and extent of experiments indicated by number of figures), and whether an abstract can indicate if a paper was published or determine the journal a paper was published in can be gained, alongside additional insights in a similar vein.

II. Methodology

The data set used is taken from [1], which provides 1.7 million papers' metadata. Due to memory and compute constraints, the dataset is randomly subsampled to 100,000 data points. After removing data points with missing values, 76,560 data points remain. Of the data points used, 54% of them contain a figure count (contain the word "figure"), 76% of them contain a page count (contain the word "page"), 10% of them contain version information (contain the word "version"), and 9% of them contain journal information (contain the word "publish" or "submit"). There are also some more unique comments. For example, 0.5% of the comments do not contain a space character.

To predict comments from paper abstracts using text generation, a sequence-to-sequence model will be used. Specifically, the small, encoder-decoder T5 transformer will be used. The encoder-decoder architecture of the T5 transformer makes it a good candidate for the task. Since abstracts and comments are clearly different bodies of text, the model needs an encoder and a decoder to learn the more fine-grained traits of each type of text, where the encoder will build representations for the abstracts, and the decoder will use the meaning from the abstract representations to build comments. The T5 transformer is pretrained on a variety of text generation tasks, including translation, summarization, question answering, fill-in-the-blank tasks, and more, where the novelty of the T5 transformer was its unified view of text generation tasks. The T5 transformer takes some directive as an input alongside the text necessary to complete the task. For example, “summarize:” could be inserted at the beginning of a text sequence to be summarized. Note that the task of predicting comments from paper abstracts does not cleanly fall into any of the directives that the T5 transformer was pretrained on. On the other hand, due to the variety of tasks that the T5 transformer was trained on, the transformer has seen success in generalizing to other text generation tasks when a directive is not appended to the input text ([2]), which is what will be studied for this task. The small version of the T5 transformer is used due to memory and compute constraints, where larger transformers caused memory errors and took an unreasonable amount of time to run. A transformer is being used due to transformers achieving state-of-the-art (SOTA) results for a variety of text generation tasks in recent years ([3]), and since the T5 transformer is readily available and well-documented in [4].

To train the T5 model, both the abstracts and comments were first tokenized. The corresponding tokenizer from [4] was used. The T5 tokenizer tokenizes the input by turning words, for example, into indices in order to map words to embeddings. Specifically, the tokenizer uses SentencePiece, introduced in [5]. SentencePiece tokenization is especially useful for tokenizing different languages, which may not use a space to separate words. An example is provided in Fig. 1.



```
print(tokenizer.tokenize("Phrase: I can't fly, I think."))
['_Ph', 'rase', ':', '_I', '_can', "'", 't', '_fly', ',', '_I', '_think', '.']
```

Fig. 1. Python SentencePiece Tokenization

SentencePiece tokenization, and other methods of tokenization, work well for this task especially because they separate punction from words, which will help the model to learn about the structure of the comments, which use commas and semi-colons to separate pieces of information, as shown previously. SentencePiece in particular is chosen since the T5 transformer was trained using a tokenizer using SentencePiece, allowing for the fine-tuned model to better use the information that the pre-trained model had previously learned.

The input for the T5 model also requires relative positional embeddings to track the positions of tokens. To train the model, the model is first given the unique index of each token in the input for encoding, whereas the target sequence is prepended by a start-sequence token and appended with the end-sequence token. Also, the token used for padding is set to the start-sequence token.

To evaluate the model, the primary metric used is the ROUGE score, which is generally used in summarization and translation tasks. The ROUGE score considers evaluations such as matching unigrams and bigrams, and also considers evaluations such as the longest common subsequence (LCS). The ROUGE score is used because, similarly to other text generation tasks, the quality of the output of the model cannot be easily determined by more common metrics, such as accuracy, due to the difficulty of the task. The ROUGE score instead allows comparing sub-parts of the output to check for correctness. N-grams are important for this task because, for example, correctly predicting that the comment includes information about the number of pages or figures is important, and predicting the actual number of pages or figures is important, and so is ordering the terms into a correct structure. The other ROUGE metrics, such as LCS, are important for similar reasons (degree of correct output structure and content).

III. Experiments

First, the small pre-trained T5 model introduced in [3] was loaded from [4]. Similarly, the base tokenizer for the small T5 model was loaded from [4]. The abstracts and comments were tokenized and padded to have up to 1024 tokens. 1024 tokens is sufficiently long to ensure none of the inputs were truncated. The data points were separated such that 80% of the data was used for training and 20% was used for testing. Using cross validation, hyperparameters that performed well are given in Table 1.

TABLE I. Hyperparameters

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	2E-05
Weight Decay	0.01
Epochs	5
Train Batch Size	16
Test Batch Size	16

The results after training the model are given in Table 2, where Rouge1 is for unigrams, Rouge2 is for bigrams, and RougeLCS is for the longest common subsequence.

TABLE II. ROUGE Results

Metric	Value
Rouge1	0.216100
Rouge2	0.053600
RougeLCS	0.208200

As an example, one of the model outputs is “10 pages, 5 figures, accepted for publication in ApJ”. To further investigate the weaknesses of the model, the precisions, recalls, and F1-scores are given in Table 3 for a variety of cases.

TABLE III. Comment Type Results

Phrases	Precision	Recall	F1-score
Figure	0.593700	0.524600	0.557000
Page	0.847700	0.560900	0.675100
Version	0.145800	0.004100	0.008000
Publish or Submit	0.081400	0.041300	0.054800
No Spaces	0	0	0

In Table 3, each row calculates the given metric for instances where the phrase is or is not in the actual comment versus the prediction. Also, note that “Publish or Submit” indicates that either publish or submit were present, and “No Spaces” indicates that the comment had no spaces in it. Table 3 indicates that for abstracts that had comments with the word figure, or page, the performance was more reasonable. From the analysis performed previously, the majority of the comments had the word figure or page present, allowing the model to better learn which abstracts yielded comments with such information. On the other hand, comments containing the word version had poor performance due to the low recall, indicating that the model had many false negatives, indicating the predictions frequently did not have the word version in them even though the labels did. Both the version and publish or submit cases performed comparatively poorly, likely due to the smaller number of examples of each case. The scores of 0 for the case of the comment not having spaces indicates that the model was not able to learn when comments would only be a single word from the abstract, indicating that more unique comments were difficult, or even impossible to predict from abstract information alone. The poor recall scores indicate that the model was not able to easily distinguish between comments with different types of information (such as having page count information instead of version information) from the abstracts.

Next, Table 4 shows the average number of pages and figures predicted by the model compared to the actual number of pages and figures when both the model predicted that the comment would contain a page or figure count, and when the actual comment contained a page or figure count. Analyzing the data in this way allows for the performance of the model with regard to predicting the figure and page counts to be considered without considering the models performance in determining the appropriate comment structure (such as having information about the number of pages), which was previously analyzed. Additionally, Table 4 shows the R^2 value for each case.

TABLE IV. PAGE AND FIGURE COUNT RESULTS

Phrases	**Predicted Average**	**Label Average**	**R^2**
Page	8.284400	17.286700	-0.275100
Figure	3.823700	6.132800	-0.107200

Firstly, Table 4 shows that the R^2 values in both cases are negative, meaning that the model was not able to learn the page or figure count distributions from the abstract. Similarly, the model was not able to achieve appropriate averages regarding the number of pages and the number of figures, though the model did learn that the page count average is higher than the figure count average. The poor performance of the model indicates that paper abstracts do not provide much information to infer paper length or figure counts. The lack of performance in determining the associated values aids in explaining the low ROUGE scores.

IV. CONCLUSION

The T5 transformer was able to learn about the structure of the comments primarily from the label distribution. The model was able to learn some information about the comment structure from the abstracts, as indicated by the ROUGE and F1-scores, but the model was not able to learn more fine-grained details such as the number of figures or the structure for less common comments, such as comments not containing any spaces. The results indicate that the abstract alone does not provide much information regarding the structure or content of a comment on an associated paper. A limitation is in the size of the transformer and data used, where using a larger transformer and additional data could perform better given more compute.

REFERENCES

[1] C. University, "ArXiv dataset," *Kaggle*, 15-Apr-2023. [Online]. Available: https://www.kaggle.com/datasets/Cornell-University/arxiv. [Accessed: 19-Apr-2023].

[2] S. Shleifer, "T5 Finetuning Tips," *Hugging Face Forums*, 11-Aug-2020. [Online]. Available: https://discuss.huggingface.co/t/t5-finetuning-tips/684. [Accessed: 19-Apr-2023].

[3] C. Raffel *et al.*, 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer', *CoRR*, vol. abs/1910.10683, 2019.

[4] "Hugging face – the AI community building the future.," *Hugging Face* –. [Online]. Available: https://huggingface.co/. [Accessed: 20-Apr-2023].

[5] T. Kudo and J. Richardson, 'SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing', *CoRR*, vol. abs/1808.06226, 2018.