Pull Request Title Generation

Present to
DR. EKAPOL CHUANGSUWANICH

By No.1 NLP

6230207021	แทนทวิช	พงศ์ทวิช
6331349721	อนวิต	เจตมงคลวงศ์
6331344521	วิริทธิ์พล	ลิมปวิทยากุล
6332016921	ตราภูมิ	สกุลปิยวงศ์
6332003721	คุณานนต์	รัตนโกเศศ
6332018121	ทินกฤต	อุตสา

2110572 NLP SYS
2st Semester, 2022

Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University

Table of contents

1) Motivation and overview

For every developer, there must be some incident where the developer cannot decide which name should be used for the pull request that they have been working on. When such an event occurred, the developer might end up wasting considerable time just to name the pull request. This project aims to eliminate this problem by automated the pull request naming process.

To achieve this goal, we use a language model to predict the suitable name for pull requests from commit messages. Our work in this project mostly involved finding a good approach to train our model.

2) Literature review/background of techniques related to your problem

BART (Bidirectional and autoRegressive Transformer) architecture is considered one of the top text generation models in the current era [1]. In our project, we leverage the power of BART and employ various preprocessing techniques such as textrank algorithms , separate tokens to enhance its performance.

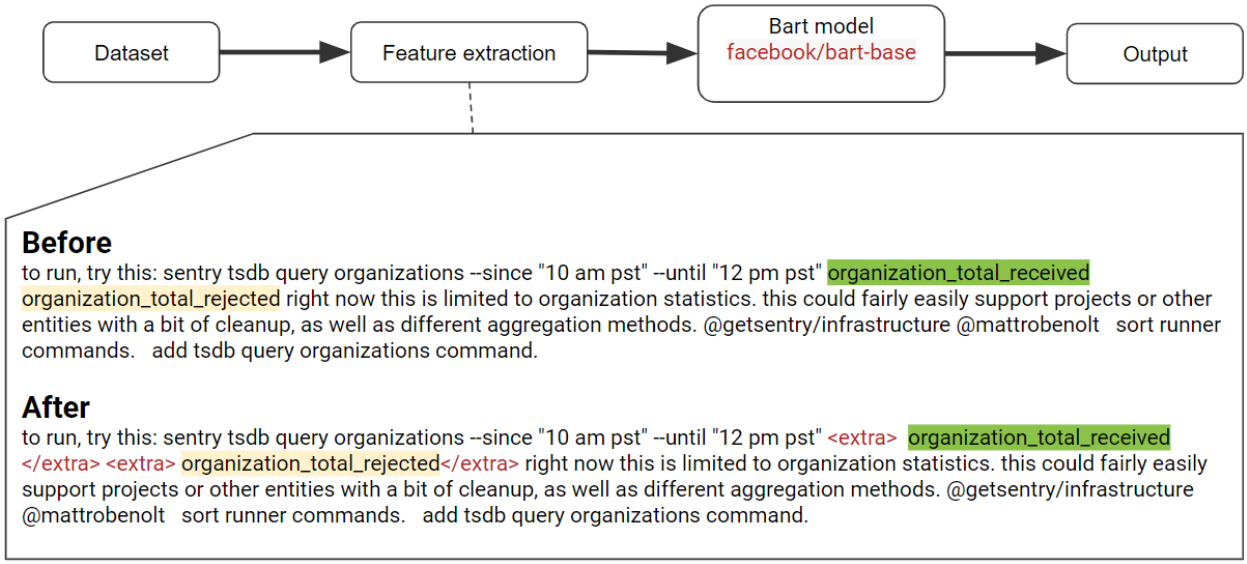
The techniques which may improve the performance is to combine TextRank with BART model [2]. We carried out an approach which concatenates the result from the BART model with TextRank then fed it to another BART model.

3) Describe your techniques/methods

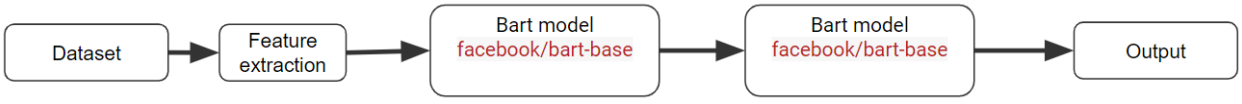
In order to attain the intended outcome, we have try four different approaches during the process of model training, the four approaches including:

1. Using the original BART model "Facebook/bart-base" with the dataset from the paper.

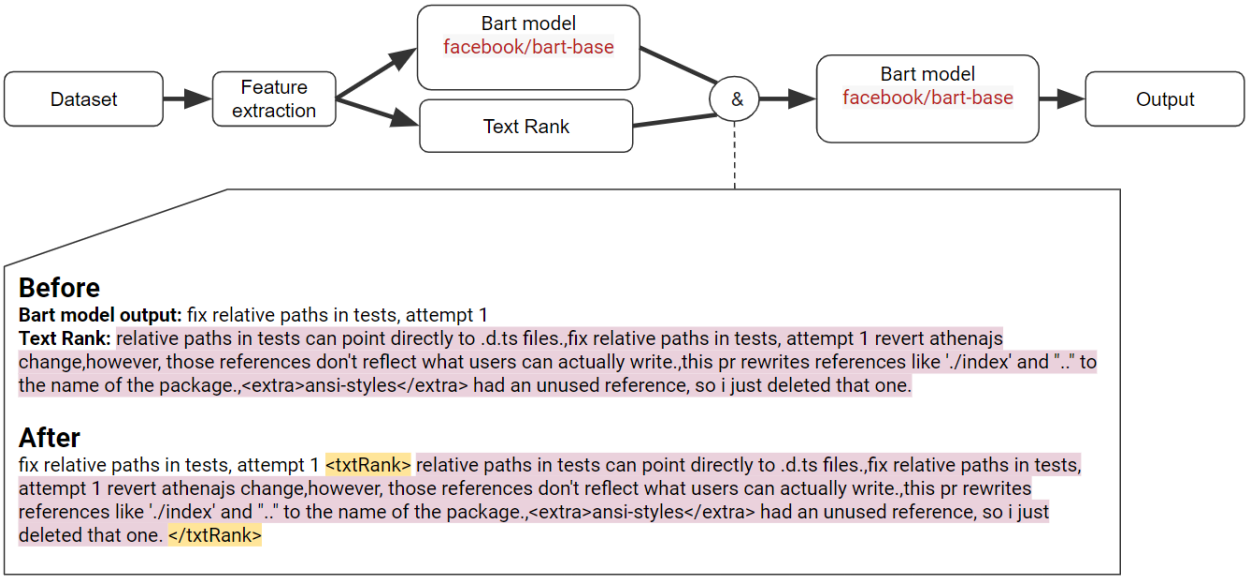
2. Perform feature extraction on the input text and concatenate it at the beginning of the input text as the new input. After concatenating the extracted features, we add separate tokens to mark the beginning and ending of the feature sequence. Then, proceed with a similar approach as in approach 1.



3. After obtaining the output from approach 2, we concatenate the summarization with the raw dataset and train the BART model once again.



4. In approach 2 , after performing feature extraction, we train two separate models: one using the BART model and the other using the TextRank algorithm. The outputs from both models are then concatenated with the output from the BART first and then the TextRank, and the resulting text is passed through the BART model once again.



Each approach was trained for 4 epochs with a batch size of 4. The maximum length of the encoder was set to 512, while the decoder had a maximum length of 64.

4) Data

PRTiger Dataset(Pull Request Title Generation) (link)

The first dataset that can be leveraged for PR title generation. Focus on helping contributors to compose non-trivial PR titles, which aims to help integrators to grasp the main changes made in the PRs. The dataset creator identified and applied several rules to keep the PR titles that suit the use scenario. In the end, they have 43,816 PR titles belonging to 495 GitHub repositories in PRTiger. The source sequence in the dataset is the concatenation of PR description, commit messages, and linked issue title, with an average length of 114 words. The target sequence is the corresponding PR title, with an average length of 7 words.

5) Metrics

ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of automatic summaries or machine-generated text compared to human reference summaries. It measures the overlap between the generated summary and the reference summary using various algorithms.

The Rouge-N Score is calculated by counting N-gram matched words, then dividing that value by the total N-gram that exists in the real sentence. Rouge-L Score is calculated by dividing longest common subsequence length by total length.

6) Results

Model	ROUGE-1	ROUGE-2	ROUGE-L
BART	46.8684	25.0862	42.9303
BART+ preprocessing	46.5925	24.7061	42.7126
BART+ preprocessing+ BART	45.6653	23.9111	41.6356
BART+ preprocessing+ TextRank + BART	46.3706	24.5945	42.4827

7) Discussion and Error Analysis

Example 1

Document: refactor typecheckfunctionbodyrequest to return the type-checked body, and remove typecheckabstractfunctionbody in favor of a method on abstractfunctiondecl. in addition, start returning an errorexpr body instead of a partially type-checked body if type-checking fails. then, start using gettypecheckedbody in silgen to allow lazy type-checking when emitting function definitions. [ast] remove newbodykind default a bunch of callers were accidentally misusing it. add abstractfunctiondecl::gettypecheckedbody refactor typecheckfunctionbodyrequest to return the type-checked body, and remove typecheckabstractfunctionbody in favor of a method on abstractfunctiondecl. in addition, start returning an errorexpr body instead of a partially type-checked body if type-checking fails. add some missing calls to setthrows we no longer run error checking for already type-checked synthesized functions, so add a couple of setthrows calls where they were previously missing. [silgen] use gettypecheckedbody use gettypecheckedbody to allow lazy type-checking when emitting function definitions.

Target: add and use abstractfunctiondecl::gettypecheckedbody

Human summary: refactor typecheckfunctionbodyrequest

ChatGPT summary: refactor typecheckfunctionbodyrequest, remove typecheckabstractfunctionbody, and improve error handling and lazy type-checking.

Model number	predicted results
1	use gettypecheckedbody to allow lazy type-checking when emitting function definitions
2	Use gettypecheckedbody instead of typecheckabstractfunctionbody in silgen
3	use gettypecheckedbody in silgen to allow lazy type-checking
4	use gettypecheckedbody in silgen to allow lazy type-checking

Example 2

Document: add or edit tests to reflect the change. (run with npm test.) follow the advice from the readme. avoid common mistakes. run npm run lint package-name (or tsc if no tslint.json is present). provide a url to documentation or source code which provides context for the suggested changes: increase the version number in the header if appropriate. (not applicable) if you are making substantial changes, consider adding a tslint.json containing { "extends": "dtslint/dt.json" }. (already available) updated typings to match vimeo's player.js v2.6.3, updated tests to reflect new methods, properties, and events cleanup of unused boilerplate lines setplaybackrate may return rangeerror options argument in constructor is optional removed typings for unexposed internals both playermap and readymap are internal weakmaps in vimeo's player.js and are never exposed, so there is no need to export their types. merge remote-tracking branch 'upstream/master'

Target: removing typings for unexposed internals

Human summary: Update tests to reflect the change

ChatGPT summary: Update Vimeo player typings for v2.6.3, including new methods/events, optional constructor argument, and cleanup.

Model number	predicted results
1	updated typings to match vimeo's player.js v2.6.3
2	updated definitions to match vimeo's player.js v2.6.3
3	updated typings for vimeo's player.js v2.6.3
4	updated typings to match vimeo's player.js v2.6.3

Analysis:

The models showed that the name entity word, e.g. v2.6.3, in the target also appears in the prediction of each model. In addition, all model predictions also have the similar meaning to the human, target and ChatGPT summary. But the difference between these models lied on the choices of words. Model 1 and model 2 have predicted longer sentence compare to those of model 3 and 4. And the summarized from model 3 displayed a similar pattern to summarized of model 4 with almost the same choice of words.

8) Conclusion

In conclusion, the Rouge score and human prediction show that the technique which perform in the experiment cannot improve the model. Therefore, the pure BART model is the best model for the Pull request title generate task.

9) Reference

[1] T. Zhang et al., “Automatic pull request title generation,” arXiv.org, https://arxiv.org/abs/2206.10430 (accessed May 24, 2023).

[2] Y. Chen and Q. Song, “News text summarization method based on Bart-TextRank model,” *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2021. doi:10.1109/iaeac50856.2021.9390683