Leveraging Generated Code Summaries in Question Answering System for Code

Georgy Andryushchenko

Motivation and Relevance

- The source code may be sometimes highly complex for understanding
- The text in natural language may help programmers understand the code
- A question answering system may answer specific questions from a programmer. (Ex.: "What is the purpose of this variable?", "In what cases does this code throws an exception?")
- As a developer in the Code QA team of Advanced Engineering School at Innopolis University, I work on implementing the tool for question answering over source code.

Problem Statement

At the moment, the implemented model for question answering over code demonstrates quite poor results according to BLEU, BERTScore, F1 metrics and the rate of exact match. The goal of the project is to try to further improve the results of the model.

Concepts

- **Code.** Only Python functions are considered as a code in the scope of this project.
- **Summary.** Textual description of the functionality of the code.
- **Question Answering System.** A system powered by a LLM that takes a question regarding code from the user as an input prompt and return an answer to the asked question.
- **Large Language Model(LLM).** Probabilistic model of a natural language that has a large number of parameters and able to achieve general-purpose understanding.

Concepts

- **Prompt.** Input text provided to the model to generate a specific output.
- **Fine-tuning.** Further training of a pre-trained model on a specific dataset or task to adapt it new domains or specialized tasks.
- **BLEU.** Approach to evaluate the quality of the generated text based on the concept of precision in n-gram matches between the generated translation and the reference translations.
- **BERTScore.** Leverages the pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity.

Relevant Existing Work

Datasets

- CodeQA contains a Java dataset with 119,778 question-answer pairs and a Python dataset obtained from code comment by parsing with syntactic rules.
- CS1QA is a dataset consisting of 9,237 question-answer pairs gathered from chat logs in an introductory programming class using Python
- Docstring dataset contains 150370 triples of function declarations, function docstrings, and function bodies of Python code

Relevant Existing Work

Models

- Impressive performance on text generation tasks has been demonstrated by Starcoder, which has been pretrained on a vast code corpus. This model is also used in a question answering system called StarChat.
- CodeT5 take advantages of novel identifier-aware pre-training task that enables the model to distinguish which code tokens are identifiers and to recover them when they are masked.

Methodology

Baseline Model Building

1. Collect and prepare CodeQA and CS1QA datasets to create a single unified dataset.
2. Fine-tune StarCoder model on the unified dataset to make a baseline model.
3. Generate testing answers and evaluate them by calculating BLEU, BERTScore, F1, and Exact Match metrics.

Methodology

Summarization Model Building

1. Collect and prepare Docstring dataset for code summarization
2. Fine-tune CodeT5 model on the Docstring dataset
3. Evaluate the summarization model with BLEU, ROUGE, and METEOR metrics

Methodology

Proposed Model Building

1. Fine-tune StarCoder model on the dataset that incorporates summaries generated by the summarization model
2. Evaluate the proposed model using BLEU, BERTScore, F1, and Exact Match metric and compare them with the baseline values.

Methodology

Reasons

Modification of the input prompt of a large language model may provide a positive impact to the generated results:

- Early works show that by designing appropriate prompts, LMs can achieve decent zero-shot performance
- Prompts can bring a giant leap in sample efficiency.

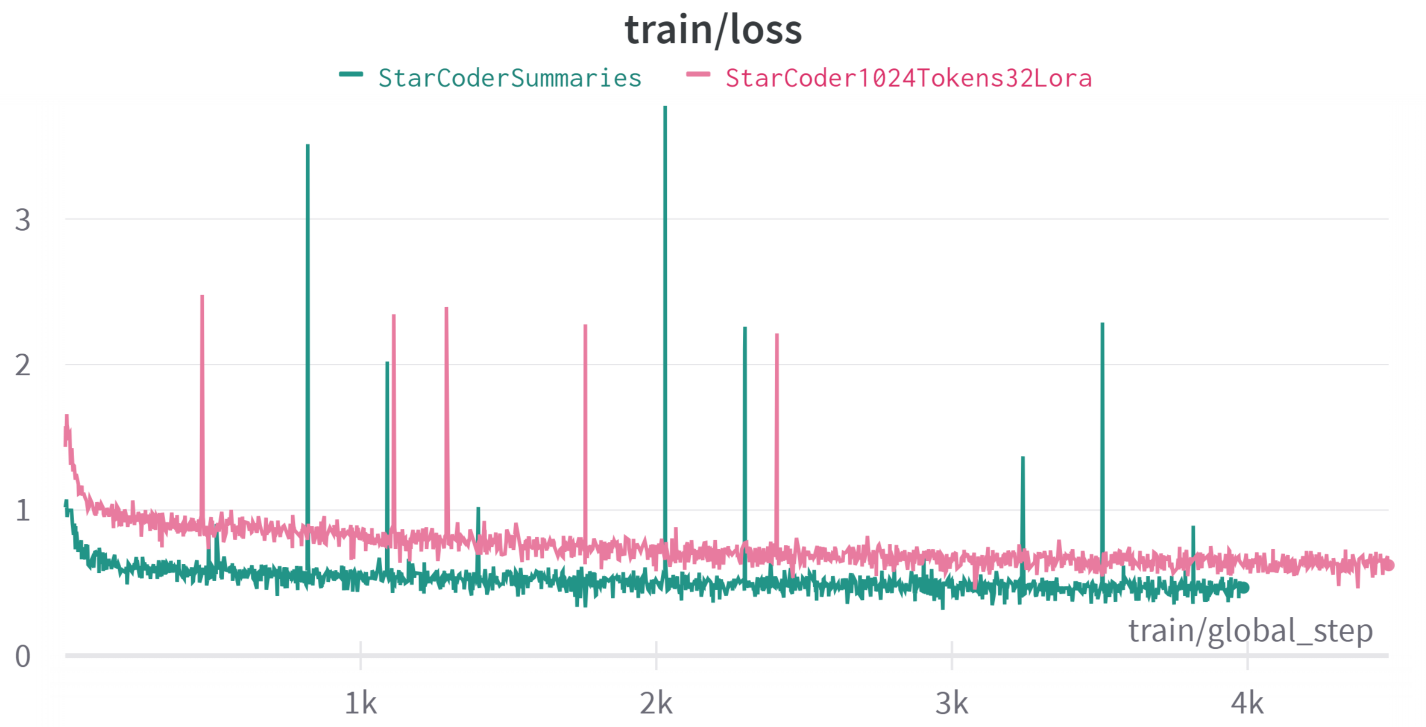
Results and Findings

Table I. Evaluation Metrics Values for Summarization Model

BLEU-4	METEOR	ROUGE-L
17.97%	37.91%	39.35%

Results and Findings

Fig. 1. Training Loss Plot for Baseline and Proposed Models



Results and Findings

Table II. Evaluation Metrics Values for Baseline and Proposed Question Answering Models

Model	BLEU4	BERTScore Precision	BERTScore Recall	BERTScore F1	F1	EM
Baseline	7.42%	68.56%	64.09%	65.89%	37.57%	9.89%
Model with Summaries	5.63%	67.29%	62.07%	64.20%	34.31%	8.34%

Analysis of Results

46% of the answers became worse in terms of BERTScore F1

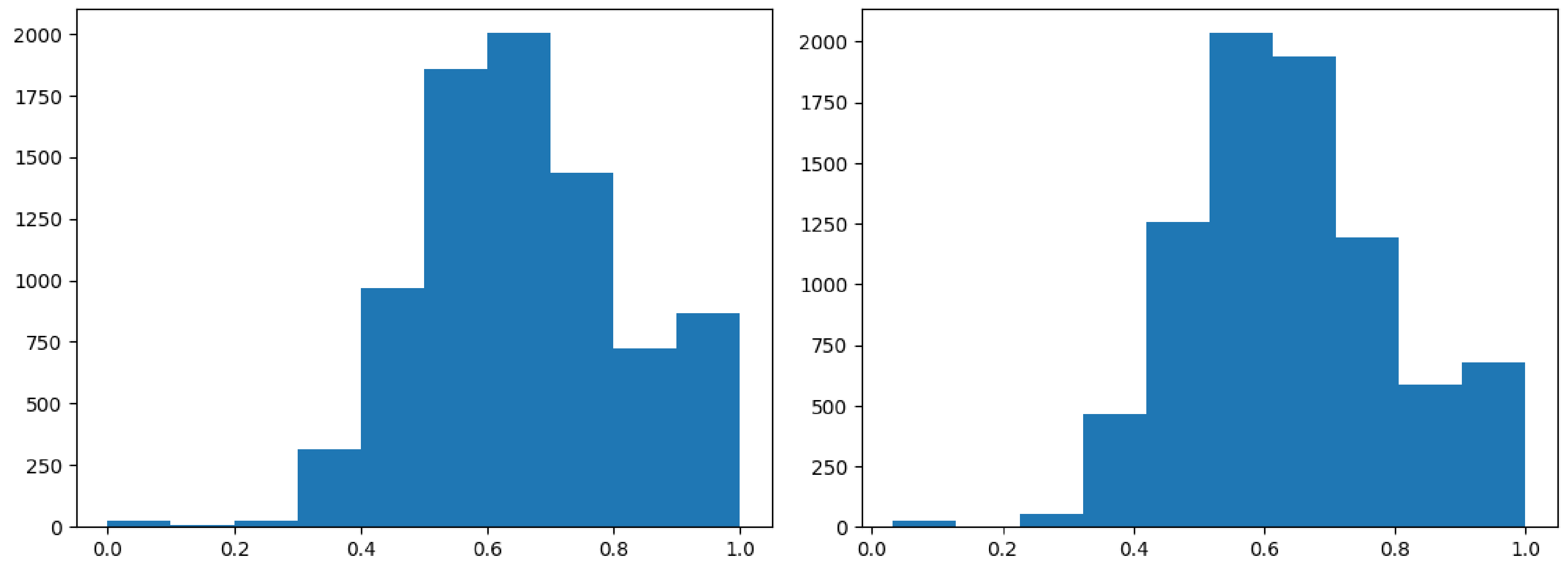


Fig 2. BERTScore F1 histogram for the baseline(left) and for the proposed(right) approaches

Analysis of Results

Takeouts from manual analysis of the generated answers:

1. Actually correct or better predictions (20%)
2. The prediction hasn’t been changed (19%)
3. The model was misled by the summary (15%)
4. Dataset problems (12%)
5. The model truncated the prediction (8%)
6. The model was confused (7%)
7. The model ignored a helpful summary (6%)
8. The model slightly changed the prediction (2%)

Proposed Improvements

1. The model does not change the prediction often. Maybe, trying different decoding techniques other than greedy decoding will give the model more freedom in choosing the words and, probably, making better predictions.
2. Quite many cases of incorrect predictions were caused by problems with the quality of the testing dataset. These problems include grammatical mistakes, unclear formulation of the question, irrelevant answer, excessively wordy answer. They may threaten the validity of the experiments. The models could be also tested on other datasets that do not have such problems.

Results of Changing Decoding Strategy

Decoding Type	BLEU4	BERTScore Precision	BERTScore Recall	BERTScore F1	F1	EM
Beam Search	3.92%	66.21%	59.24%	62.16%	32.63%	8.81%
Temperature	4.63%	64.62%	60.44%	62.09%	30.71%	6.82%
Sampling Top K	3.31%	59.76%	57.71%	58.32%	23.82%	4.36%
Sampling Top P	3.08%	59.30%	57.24%	57.85%	23.12%	4.23%
Sampling	2.87%	58.72%	56.89%	57.40%	22.75%	4.10%

Results of Changing Testing Dataset

Table IV. Evaluation Metrics Values for Baseline and Proposed Question Answering Models on a Higher Quality Subset

Model	BLEU4	BERTScore_P	BERTScore_R	BERTScore_F1	F1	EM
Baseline	9.73%	74.09%	69.66%	71.63%	43.27%	9.00%
Proposed Model	4.65%	70.23%	64.12%	66.75%	35.22%	7.00%

Results of Changing Testing Dataset

Table V. Evaluation Metrics Values for Baseline and Proposed Question Answering Models on ClassEval Benchmark

Model	BLEU4	BERTScore_P	BERTScore_R	BERTScore_F1	F1	EM
Baseline	4.24%	65.26%	52.72%	57.96%	24.86%	0.39%
Proposed Model	1.18%	59.37%	46.31%	51.66%	20.49%	0.34%

Conclusion

- Baseline question-answering StarCoder model and the proposed model were trained and tested.
- The proposed model happened to show poorer answers than the baseline model.
- 31% of the answers are actually the same or even better; the other 61% of the answer are actually worse
- Changing decoding strategies could not increase the metric values of the testing answers.
- The testing on other datasets showed no superiority of the proposed model over the baseline model.
- Implementing injection of the generated summaries in the input of the model leads to the quality decrease in the circumstances of the presented experiments.

Future Work

The problem still requires further investigation whether the results of this experiment were significantly affected by the threats such as flawed evaluation.

BERTScore is a decent metric for comparing the text based on their meaning. However, it is still quite sensitive to the formulations and the terms used. Maybe, the metric that is not such sensitive would also evaluate the answers in a different way.