Reproducibility report of CoSQA: 20,000+ Web Queries for Code Search and Question Answering (2) for ML Reproducibility Challenge 2021

Iman Barati*, **Yeganeh Morshedzadeh***, **Ghazaleh Mahmoodi***
School of Computer Engineering
Iran University of Science and Technology, Tehran, Iran
{iman_barati, y_morshedzadeh, gh_mahmoodi}.iust.ac.ir

Reproducibility Summary

Abstract/Summary of the Original Paper (2)

The goal of the authors in the paper is finding relevant code pieces that matches a user's query. More specifically, they focused on two tasks, code question answering and code search. They gathered and used a new dataset named CoSQA since for the described problem a richer and bigger dataset than the previous ones is needed. Additionally, they introduced CoCLR which incorporates code contrastive learning into the siamese network with CodeBERT (1). In code constrastive method the goal is to learn the word representations in a way that similar objects are as close as possible whereas the disimilar objects are kept as far as possible. Therefore, two augmentation methods were used, In-Batch Augmentation (IBA) and Query-Rewritten Augmentation (QRA). In conclusion, by using these models and dataset they achieved better quality inn query-code matching tasks.

Scope of Reproducibility

- The main claim of the paper that we tried to reproduce is that the siamese network with CodeBERT (1) and CoCLR models improve overall performance on both tasks (code search and code question answering) by using the CoSQA dataset, particularly on the CodeXGLUE WebQueryTest (3) [2]. Specifically, CoCLR increased the results by 15.6% the WebQueryTest.
- Therefore, in this work, we tried to show:
  1. The performance of the siamese network with the CodeBert model on the CodeSearchNet + CoSQA as the training and testing dataset. To do that we used the same metrics as the paper that are accuracy for code question answering task and MRR for code search task.
  2. We performed previous experiments for the CoCLR (siamese network + CodeBERT using code constrastive learning) model as well.
  3. We trained CoCLR with different variations of augmentations (IBA and QRA) for code search task:
     (a) in-batch + query-rewritten (delete)
     (b) in-batch + query-rewritten (copy)
     (c) in-batch + query-rewritten (switch)
  4. We showed the performance of the CoCLR with various code componets on code search task:
     (a) complete code
     (b) header only
     (c) doc only
     (d) no header

*All authors contributed equally to this work.
[2]https://github.com/microsoft/CodeXGLUE

Submitted to ML Reproducibility Challenge 2021. Do not distribute.

(e) no doc
(f) no body

Methodology

- We used the authors' code provided on Jun-jie-Huang's GitHub.
- Our hardware resource was GPU.RTX2060 Super.xlarge and the specifications is as below:
  - RAM: 23.4GB
  - VCPUs: 6 VCPU
  - Disk: 200GB
- We added ten bash files that includes all the experiments done by authors. These can be all be run by a single bash file which simplifies automating running process significantly.
- The total budget in terms of GPU hours per task is as below:

Task	Approximate time for each epoch	Number of epochs for training	Total approximate time
Code Question Answering	15 minutes	10	2.5 hours
Code Search	30 minutes	10	5 hour

Table 1: GPU computation time

Results

The results are summarized in the below tables:

- Overall results:
  - Code Question Answering
    - Test hyperparameters:
      - max_seq_length: 200
      - per_gpu_eval_batch_size: 2
    - Train hyperparameters:
      - mmax_seq_length: 200
      - per_gpu_train_batch_size: 8
      - per_gpu_eval_batch_size: 16
  - Code Search
    - Test hyperparameters:
      - max_seq_length: 200
      - per_gpu_retrieval_batch_size: 67
    - Train hyperparameters:
      - max_seq_length: 200
      - per_gpu_train_batch_size: 8
      - per_gpu_retrieval_batch_size: 67

Model	Code Question Answering (Accuracy)	Code Search (MRR)
CodeBERT (Paper's)	52.87	54.41
CodeBERT + CoCLR (Paper's)	63.38	64.66
CodeBERT (Our's)	46.75	54.60
CodeBERT + CoCLR (Our's)	66.92	64.28

Table 2: Evaluation on two tasks. The data is CodeSearchNet + CoSQA.

- Augmentation results:
  - Test hyperparameters:
    - max_seq_length: 200
    - per_gpu_retrieval_batch_size: 67
  - Train hyperparameters:
    - max_seq_length: 200
    - per_gpu_train_batch_size: 8
    - per_gpu_retrieval_batch_size: 67

Augmentations	Train (MRR)	Test (MRR)
Delete (Paper's)	-	63.41
Copy (Paper's)	-	63.97
Switch (Paper's)	-	64.66
Delete (Our's)	62.15	63.87
Copy (Our's)	63.32	64.39
Switch (Our's)	64.34	64.28

Table 3: Performance of CodeBERT with different augmentations (in-batch + query-rewritten) in COCLR on code search.

- Code components results:
  - Test hyperparameters:
    - max_seq_length: 200
    - per_gpu_retrieval_batch_size: 67
  - Train hyperparameters:
    - max_seq_length: 160
    - per_gpu_train_batch_size: 8
    - per_gpu_retrieval_batch_size: 67

Augmentations	Our's (MRR)	Paper's (MRR)
Complete code	64.28	64.66
w/o header	60.06	62.01
w/o body	58.31	59.11
w/o documentation	57.57	58.54
w/o header & body	51.67	52.89
w/o header & documentation	42.16	43.35
w/o body & documentation	43.47	42.71

Table 4: Performance of CoCLR-incorporated CodeBERT trained and tested with different code components on code search.

- Reducing the max sequence length resulted in the poor performance of the model.
- Changing batch size had little impact on the performance of the model with requiring less RAM but it increased the running time. In addition, in case of the question-answering task, it resulted in worsening the accuracy.

What was easy

- Good README file and step-by-step instructions on how to use the code they provided in their GitHub repository.
- Running commands as a bash file which helps automate the testing and training process.
- They used command-line arguments to change the training and testing conditions/inputs which reduce unintentional mistakes and confusion.
- Providing log file for each command which gives good information about running process.

What was difficult

- Our resource was not as powerful as their resource since we had to change hyper-parameters such as decreasing the batch_size in order to train the models. Specifically, we got poorer results for the question-answering task.
- Not enough instruction to write the command with the correct argument to get the first four rows of results in table 6 which just uses the IBA or QRA methods.
- The confusion between the names of the code and their corresponding part in the paper. For example, the term vanilla model was used as the base model which is equal to the term CodeBERT trained with CSN+CoSQA.
- Difficulty to download the datasets since they were on google drive. Wget/curl command was not available to download these materials.

Conclusion

In conclusion, we achieved quite the same results as the original paper. Our reproducibility study concurs with their main claim in most cases. More precisely, except in one case that is for CodeBERT model on code question answering we obtained poorer results due to modifications to some hyperparameters to make the code executable and model trainable on our hardware. For all the other cases what we achieved was similar to what was reported by paper.

References

[1] FENG, Z., GUO, D., TANG, D., DUAN, N., FENG, X., GONG, M., SHOU, L., QIN, B., LIU, T., JIANG, D., AND ZHOU, M. Codebert: A pre-trained model for programming and natural languages, 2020.

[2] HUANG, J., TANG, D., SHOU, L., GONG, M., XU, K., JIANG, D., ZHOU, M., AND DUAN, N. CoSQA: 20,000+ web queries for code search and question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Online, Aug. 2021), Association for Computational Linguistics, pp. 5690–5700.

[3] LU, S., GUO, D., REN, S., HUANG, J., SVYATKOVSKIY, A., BLANCO, A., CLEMENT, C. B., DRAIN, D., JIANG, D., TANG, D., LI, G., ZHOU, L., SHOU, L., ZHOU, L., TUFANO, M., GONG, M., ZHOU, M., DUAN, N., SUNDARESAN, N., DENG, S. K., FU, S., AND LIU, S. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR abs/2102.04664* (2021).