TCEM: An exhaustive text-to-code evaluation metric covering edge cases for the code snippets

Wajahat Mirza
New York University
mwm356@nyu.edu

Bokang Jia
New York University
bj798@nyu.edu

Domnica Dzitac
New York University
did233@nyu.edu

Abstract

Abstract to be added at the end of the project when we have completed the development and deployment of our evaluation metric and model training.

1 Introduction

In last few years, natural language processing (NLP) and natural language understanding (NLU) techniques have been developed and used for generating code snippets (Cruz-Benito, 2020), code suggestion and completion (Franks et al., 2015), and neural code completion (Liu et al., 2016). Auto-filling or suggesting code has become a very popular feature among several IDEs whereas Graph-based statistical language model (Nguyen et al., 2015), and Deep Learning language model (Cruz-Benito et al., 2018) are other techniques used to auto-generate or predict code lines for developers. Generating code and converting a text string to a code has become a central topic of research among the NLP community. Text-to-code is a useful tool for novice developers, or even advanced users, who often forget syntactic implementation of simple code snippets. In efforts to enhance text-to-code techniques, several competitions have been designed as open source for anyone to make a contribution (Yu et al., 2018) (Spider: Yale Semantic Parsing and Text-to-SQL Challenge, 2018).

For this paper, authors have focused on the recently published dataset CoNaLa (Yin et al., 2018) that contains over 600k automatically labeled and over 2,379 manually labeled train samples (CoNaLa: The Code/Natural Language Challenge, 2018) [1]. The proposed models for the CoNaLa task are evaluated on the BLEU metric (Papineni et al., 2002) where the current baseline for this task has been established using the traditional sequence-to-sequence (seq2seq) model (Sutskever et al., 2014). Recent attempts have achieved state-of-the-art results where it was improved by up to 2.2% absolute BLEU score on the code generation (Xu et al., 2020). With the advent of modern transformer models, it is possible to improve on this baseline and demonstrate to the community the performance of these newer models, such as GPT-2 and GPT-3, on the task of text-to-code. We are exploring techniques for calibrating few-shot performance (Zhao et al., 2021) on our task, which could boost our performance from a base GPT-3 model.

While developing a model with newer transformers tweaked towards the task may improve the results and perhaps, surpass state-of-the-art, it is eminent to acknowledge that BLEU metric does not provide the most accurate evaluation of the model performance. It is to recognise that BLEU is very efficient in evaluating large corpora, however, "it is unreliable at the sentence or sub-sentence levels, and with a single reference" (Song et al., 2013) which is a pivotal element of the CoNaLa and any code snippet generating task. Hence, we would like to introduce a new evaluation metric, **TCEM**: An exhaustive text-to-code translation evaluation metric covering edge cases for the auto-generated code snippets. TCEM is more powerful and accurate than BLEU because it covers edge cases, evaluates the functionality of the code on each line level, and provides overall cumulative mean score based on the successfully passed unit tests.

The GitHub repository is available at `https://github.com/finnthedawg/text2snippet`.

[1] *Please note that we spoke with Pengcheng Yin who informed that the database has been down which is why only one result i.e. baseline is reflected. Other people have attempted the Task and achieved state-of-the-art results*

2 Literature Review

Previous related work on text-to-code tasks have evaluated their models with BLEU, a method for Automatic Evaluation of Machine Translation (MT) (Papineni et al., 2002). BLEU is the pioneer of automatic evaluation for MT that provided a quicker and cheaper alternative of evaluation other than using human subjects. The assumption behind BLEU is that good translations will share many `n-grams` with translations done by humans (Papineni et al., 2002). BLEU metric ranges from 0 to 1 (score 1 being attained almost never) and has three main components: `N-gram precision`, `Clipping` and `Brevity Penalty`. BLEU takes the geometric mean of a corpus' modified precision scores, after which it multiplies the output with an exponential brevity penalty factor (Papineni et al., 2002). This makes BLEU a precision-oriented metric with strong penalty methods.

While the BLEU metric passed the test of time, being widely used in evaluating MT tasks, including text-to-code, it has a multitude of limitations that were pointed out in previous related literature (Song et al., 2013). Firstly, the metric is not fit for sentence level evaluation (or even for short corpora). This is because the BLEU metric uses a geometric mean, which applied on short documents/sentences results in almost 0 precision. Previous related work proposes a solution to this issue by replacing the geometric mean with the arithmetic mean (Satanjeev et al., 2005). Others approached this issue from a different angle, for example, the `ORANGE` metric replaced the high `n-gram` precision with `uni-grams`, which seem to perform better on evaluating this task (Lin et al., 2002). More recent work integrated the above techniques and achieved a better correlation score (Song et al., 2013). However, we moved away from improving BLUE and we propose a new evaluation metric, **TCEM**: An exhaustive text-to-code translation evaluation metric, that focuses on covering edge cases for the auto-generated code snippets, instead of using `n-grams`.

3 Preliminary Model Training and Results

At this stage in the project, model development is under way whereas TCEM development is in the backlog. We obtained preliminary results for auto-generation of the code snippets using GPT-3 on CoNaLa sub-dataset. Two different prompts from GPT-3 using OpenAI API have been attempted to run the experiments.

Listing 1: Experiment Parameter Settings

```
temperature = 0.0
top_p = 1,
max_tokens=50,
stop='\n'
```

A temperature of 0.0 was chosen because of the deterministic nature of text-to-code task [2]. Since we are doing few-shot learning (Zhao et al., 2021), the primary focus will be on the prompt design.

Prompt	Engine	Examples	BLEU
1	Davinci	100	30.06
1	Davinci	20	38.82
1	Curie	100	18.24
2	Davinci	100	18.25

Table 1: Preliminary Results

The BLEU score of 30.06 on engine `Davinci` is already greater than the baseline results from CoNaLa task. However, these results can be further improved and solidified by understanding the importance and difficulties of designing suitable prompts that will result in good predictions (Zhao et al., 2021) (Gao et al., 2020). Further investigation on different prompt design and their effect on the task of text-to-code will conducted. However, as indicated earlier, the BLEU score may not be representing the best evaluation of the task due to aforementioned mentioned reasons. Therefore, once model training on the entire dataset is complete, we will be evaluating the model on TCEM for comparative analysis.

4 TCEM Evaluation Metric

Mathematical logic, and unit testing to be included here.

[2] https://beta.openai.com/docs/introduction/prompt-design-101

5 Result

6 Analysis

7 Conclusion

Plan of Action (POA) onwards

We will be finalizing model training on 2379 CoNaLa data points this week. We will proceed with Unit test documentation, and development.

Collaboration statement

All collaborators are working equally in the research and development of this project. The brainstorming and development of the plan has been done in collaboration with researcher Tal Schuster and prof. Sam Bowman. The research and literature review has been conducted as a collaborative work, split equally between all members. While Bokang did the initial training of the model with GPT-3 due to availability of API on his account, Mirza and Domnica have worked on improving the performance of the model on the sub-dataset. This work flow will be replicated for rest of model training. The team will start developing the unit test cases for the new evaluation metric, TCEM following this weekend.

Acknowledgments

We would like to extend our gratitude and appreciation towards Tal Schuster `(CSAIL, MIT)`, prof. Sam Bowman `(CDS, NYU)` and Angelica Chen `(CDS, NYU)` for their time and guidance throughout the process.

References

Cruz-Benito et al., Juan and Vishwakarma, Sanjay and Martin-Fernandez, Francisco and Faro, Ismael 2020. *Automated Source Code Generation and Auto-completion Using Deep Learning: Comparing and Discussing Current Language-Model-Related Approaches*. arXiv e-prints.

Yin, Pengcheng and Deng, Bowen and Chen, Edgar and Vasilescu, Bogdan and Neubig, Graham, 2018. *Learning to Mine Aligned Code and Natural Language Pairs from Stack Overflow*. arXiv e-prints.

Sutskever, Ilya and Vinyals, Oriol and Le, Quoc V., 2014. *Sequence to Sequence Learning with Neural Networks*. arXiv e-prints.

Papineni, Kishore and Roukos, Salim and Ward, Todd and Zhu, Wei-Jing, 2002. *Bleu: a Method for Automatic Evaluation of Machine Translation*. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.

Song, Xingyi and Cohn, Trevor and Specia, Lucia, 2013. *BLEU deconstructed: Designing a better MT evaluation metric*. International Journal of Computational Linguistics and Applications.

Zhao, Tony Z. and Wallace, Eric and Feng, Shi and Klein, Dan and Singh, Sameer, 2021. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. arXiv e-prints.

Yu, Tao and Zhang, Rui and Yang, Kai and Yasunaga, Michihiro and Wang, Dongxu and Li, Zifan and Ma, James and Li, Irene and Yao, Qingning and Roman, Shanelle and Zhang, Zilin and Radev, Dragomir, 2018. *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task*. arXiv e-prints.

R. -M. Karampatsis and H. Babii and R. Robbes and C. Sutton and A. Janes, 2020. *Big Code != Big Vocabulary: Open-Vocabulary Models for Source Code*. 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)s.

Raychev, Veselin and Vechev, Martin and Yahav, Eran, 2014. *Code completion with statistical language models*. Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation.

Gu, Xiaodong and Zhang, Hongyu and Zhang, Dongmei and Kim, Sunghun, 2016. *Deep API learning*. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering.

Chatzikoumi, Eirini, 2020. *How to evaluate machine translation: A review of automated and*

human metrics. Natural Language Engineering

Gao, Tianyu and Fisch, Adam and Chen, Danqi, 2020. *Making Pre-trained Language Models Better Few-shot Learners*. arXiv e-prints.

Franks, Christine and Tu, Zhaopeng and Devanbu, Premkumar and Hellendoorn, Vincent, 2015. *CACHECA: A Cache Language Model Based Code Suggestion Tool*. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering.

Liu, Chang and Wang, Xin and Shin, Richard and Gonzalez, Joseph E and Song, Dawn, 2016. *Neural code completion*.

Nguyen, Anh Tuan and Nguyen, Tien N., 2015. *Graph-Based Statistical Language Model for Code*. Proceedings of the 37th International Conference on Software Engineering - Volume 1.

Cruz-Benito, Juan and Faro, Ismael and Martín-Fernández, Francisco and Therón, Roberto and García-Peñalvo, Francisco J., 2018. *A Deep-Learning-Based Proposal to Aid Users in Quantum Computing Programming*. Learning and Collaboration Technologies. Learning and Teaching. isbn="978-3-319-91152-6"

Xu, Frank F. and Jiang, Zhengbao and Yin, Pengcheng and Vasilescu, Bogdan and Neubig, Graham, 2020. *Incorporating External Knowledge through Pre-training for Natural Language to Code Generation*. arXiv e-prints.

Lin, Chin-Yew and Och, Franz Josef, 2004. *ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation*. COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics.

Banerjee, Satanjeev and Lavie, Alon, 2005. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization.

Tony Z. Zhao and Eric Wallace and Shi Feng and Dan Klein and Sameer Singh, 2021. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. arXiv e-prints.

Tianyu Gao and Adam Fisch and Danqi Chen, 2020. *Making Pre-trained Language Models Better Few-shot Learners*. arXiv e-prints.

Reference Links

Sequence to Sequence Learning with Neural Networks. 2018. https://conala-corpus.github.io/

CodaLab - Competition. 2018. competitions.codalab.org/competitions/19175

Next chapter in artificial writing. 2020. Nature Machine Intelligence. https://doi.org/10.1038/s42256-020-0223-0

OpenAI API. 2020. https://openai.com/blog/openai-api/

unittest — Unit testing framework — Python 3.9.2 documentation. https://docs.python.org/3/library/unittest.html

Spider: Yale Semantic Parsing and Text-to-SQL Challenge. https://yale-lily.github.io/spider