Zhong Wu, Rybak V.A.

EVALUATION METHODS FOR CODE GENERATION MODELS

This article discusses evaluation methods for code generation models, focusing on BLEU scores and Pass@k metrics.BLEU scores are used to measure the similarity and consistency between generated code and reference code, while Pass@k metrics evaluate the pass rate of generated code on predefined test cases. Some other assessment methods are also presented.These assessment methods are important for improving and optimising code generation models

Introduction

Code generation models are a key class of AI models that show great potential for automatic source code generation. However, these models need to be effectively evaluated in order to ensure the quality and accuracy of the generated code. Evaluating the quality of code generation models is a complex and critical task, and researchers and developers need to consider a variety of metrics and methods. In this paper, we will focus on two commonly used evaluation metrics: the BLEU score and the Pass@k metric.

I. BLEU SCORE

BLEU (Bilingual Evaluation Understudy) is a commonly used evaluation metric primarily used to assess the quality of machine translation tasks. It measures the similarity and consistency between the automatically generated text and the reference text [1].

The calculation of the BLEU score is based on n-gram matching, where "n"represents the length of the n-gram. The score is calculated by comparing the matches between the n-grams in the generated text and the n-grams in the reference text. Here are the specific steps involved in calculating the BLEU score:

N-gram match counting: For each n-gram (typically ranging from 1 to 4 words), the number of matches between the n-gram in the generated text and the n-gram in the reference text is counted.

N-gram count truncation: To avoid over-reliance on longer n-gram matches, the count of each n-gram is truncated or limited to the maximum count in the generated text and the reference text.

Short-text penalty: To address the issue of excessively short generated texts receiving higher BLEU scores, a penalty term is introduced to prevent short texts from inflating the scores.

Comprehensive calculation: The results of the n-gram matching calculation are combined to obtain the final BLEU score.

The BLEU score typically ranges from 0 to 1, with a score closer to 1 indicating a higher degree of similarity and consistency between the generated text and the reference text.

While BLEU is widely used in machine translation, it can also be applied to other generative tasks such as automatic summarization and text generation. However, it's important to note that the BLEU score does not fully capture the semantic accuracy and fluency of the generated text. Therefore, when evaluating code generation models, it is recommended to combine BLEU with other metrics and methods to obtain a more comprehensive evaluation result.

II. PASS@K METRICS

The Pass@k metric is a metric for evaluating code generation models that focuses on whether the generated code can pass a set of predefined test cases. It is used to measure the pass rate of generated code for a given set of test cases [2].

The Pass@k metric is calculated as follows:

Predefined set of test cases: defines a set of test cases that cover the functional requirements and boundary conditions that the generated code should have.

Execution of Generated Code: Apply the generated code to the set of test cases, execute and observe how the generated code passes on each test case. If the generated code can pass the test cases, it is considered to pass; otherwise, it is considered to fail.

Pass Rate Calculation: Calculate the percentage of generated code that passes on the set of test cases. The pass rate can be calculated in different ways depending on the specific requirements and evaluation objectives, for example, calculating the ratio of the number of passed test cases to the total number of test cases.

The advantage of the Pass@k metric is that it directly measures the functionality and usefulness of the generated code. By focusing on how well the generated code passes on actual test cases, it can provide important information about the quality and accuracy of the generated code. However, the Pass@k metric has some limitations, such as its inability to capture other quality characteristics of generated code, such as code style, readability, and maintainability.

When evaluating code generation models, Pass@k metrics can be combined with other evaluation methods, such as static analysis of code and manual evaluation, to obtain more comprehensive evaluation results.

III. Other evaluation methods

When evaluating code generation models, in addition to BLEU and PASS@k, there are several other commonly used evaluation metrics and criteria, including:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Used to evaluate the similarity between automatically generated summaries and reference summaries. ROUGE metrics include ROUGE-N (matching of N-grams) and ROUGE-L (matching of the longest common subsequence), among others.

METEOR (Metric for Evaluation of Translation with Explicit Ordering): Used to assess the quality of machine translation tasks. METEOR combines exact matching and partial matching, considering word sense and syntactic information.

CIDEr (Consensus-based Image Description Evaluation): Used to evaluate the quality of image captioning tasks. CIDEr considers the diversity and consensus of generated captions and uses a consensus measure to assess the consistency between generated and reference texts.

Perplexity: Used to evaluate the quality of language models. Perplexity measures the model's predictive ability for a given text, with lower values indicating a better fit to the data.

F1-Score: Used to evaluate the accuracy and recall of entity recognition or syntax analysis tasks in the generated code.

Compilation Success Rate: Used to evaluate whether the generated code successfully passes compilation checks by a compiler.

These evaluation metrics and criteria can be chosen and used based on the specific task and requirements to provide a more comprehensive and holistic evaluation of code generation models.

IV. Summary

To summarize, evaluating code generation models is crucial to ensure the quality and accuracy of the generated code. Two commonly used evaluation metrics are the BLEU score and the Pass@k metric.

The BLEU score is primarily used for machine translation tasks and measures the similarity and consistency between the generated text and reference text. It calculates the score based on n-gram matching and ranges from 0 to 1, with higher scores indicating greater similarity.

The Pass@k metric focuses on whether the generated code can pass a set of predefined test cases. It measures the pass rate of the generated code for the test cases, providing insights into the functionality and usefulness of the code.

Other evaluation methods include metrics like ROUGE, METEOR, CIDEr, Perplexity, and F1-Score. These metrics assess different aspects of the generated code, such as similarity to reference text, image captioning quality, language model performance, and accuracy of entity recognition or syntax analysis.

When evaluating code generation models, it is recommended to combine multiple metrics and methods to obtain a comprehensive evaluation result. This may involve considering factors such as semantic accuracy, fluency, code style, readability, maintainability, and manual evaluation, along with the quantitative metrics.

By employing a variety of evaluation metrics and methods, researchers and developers can gain a better understanding of the strengths and weaknesses of code generation models and make informed decisions regarding their quality and applicability.

1. Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 'BLEU: A Method for Automatic Evaluation of Machine Translation'. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02, 311. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001. https://doi.org/10.3115/1073083.1073135.
2. Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, et al. 'Evaluating Large Language Models Trained on Code'. arXiv, 14 July 2021. http://arxiv.org/abs/2107.03374.

Zhong Wu, master's student in the Faculty of Information Technology and Management of BSUIR, 2921123673@qq.com.

Rybak V.A., PhD, Associate Professor, vice-Rector for Academic Affairs, Belarusian State University of Informatics and Radioelectronics.