Model Card for CodeT5: Identifier-Aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation

The CodeT5 programming language model employs a unified encoder-decoder framework to support multiple code intelligence downstream applications including both understanding and generation tasks.

On this model card, you can learn more about how this model was trained, its capabilities, its intended use, and its limitations.

Model Details

Organization
Salesforce Research

Model date
September 2, 2021

Model type
Programming language model

Input
Code, text, or both

Information about parameters
CodeT5-small (60 million), CodeT5-base (220 million)

Output
Code or text. The model supports 4 generation tasks (code summarization, code generation, translation, and refinement) & 2 understanding tasks (code defect and clone detection) in the CodeXGLUE benchmark.

Read the full paper here:
https://arxiv.org/abs/2109.00859

Access the public code here:
https://github.com/salesforce/CodeT5

Citation details:

Title:	CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation
Author:	Yue Wang, Weishi Wang, Shafiq Joty, Steven C.H. Hoi
Conference:	Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)
Year:	2021

License
BSD 3-Clause

Questions/Comments
codeT5@salesforce.com

Intended Use

Primary intended use

1. Deployment as an AI-powered coding assistant or reviewer to boost the productivity of software developers. Use cases include but are not limited to:
   a. Code summarization for better software understanding and maintenance;
   b. Code autocompletion or text-to-code generation to accelerate software development;
   c. Code defect detection and refinement for automatic bug repair.
2. Improvement of other code intelligence applications through fine-tuning on another task or other data, e.g., fine-tuning CodeT5 to generate API usage descriptions.
3. Enhancement in the field of programming language processing to push towards a better understanding of code, including how to better capture code-specific knowledge and balance many code intelligence tasks.

Primary intended users

- Software developers
- NLP and software engineering researchers

Out-of-scope use cases

- CodeT5 should not be used in real-world software development processes (such as synthesizing programs) without human supervision.
- It should not be used to collect, track, or create software to track, sensitive information such as:
  - financial information, such as credit or debit card numbers, any related security codes or passwords, and bank account numbers;
  - personal information, such as specific people's names and other profile data; or
  - sensitive data, such as government-issued identification numbers, racial or ethnic origin, political opinions, religious or philosophical beliefs, and health information.
- This software should not be used to promote or profit from:
  - violence, hate, and division;
  - environmental destruction;
  - abuse of human rights; or
  - the destruction of people's physical and mental health.

Training Data

The model is trained on 8.35 million code snippets: 3.16 million bimodal instances (a function and its natural language comment in English) and 5.19 million unimodal (function-only) instances. The training data contains 6 programming languages (Python, Java, JavaScript, PHP, Ruby, Go) from CodeSearchNet data and 2 additional programming languages (C and C#) from Google BigQuery data. See the paper for details.

Metrics

We follow the CodeXGLUE benchmark (described here) to measure different downstream tasks. We employ smoothed BLEU-4 for code summarization (19.77), exact match (EM) accuracy, BLEU-4, and CodeBLEU for code generation (22.7, 41.48, 44.10 respectively), exact match accuracy and BLEU-4 for code translation (65.90 EM, 84.03 BLEU for Java to C#) and code refinement (14.18 EM, 88.90 BLEU), accuracy for code defect detection (65.78) and F1 score for code clone detection (97.2). Please see the paper for more details.

Ethical Considerations

- **Dataset bias.** The training datasets in our study are source code including user-written comments from open-source and publicly-available Github repositories that do not tie to any specific application. However, the datasets possibly encode some stereotypes like race and gender from the text comments or the source code including its variables, functions, and class names. As such, social biases would be intrinsically embedded into the models trained on them.

- **Computational cost**. Our model pre-training requires non-trivial computational resources, though we have tried our best to carefully design our experiments and improve experiments to save unnecessary computation costs. In addition, we experimented on Google Cloud Platform which purchases carbon credits to reduce its carbon footprint -- training CodeT5-base produced around 49.25 kg CO_2 which was totally offset by the provider. Furthermore, we release our pre-trained models publicly to avoid repeated training for the code intelligence research community.
- **Automation bias**. As CodeT5 can be deployed to provide assistance like code generation to aid developers, automation bias of machine learning systems should be carefully considered, especially for developers who tend to over-rely on the model-generated outputs. These systems might produce functions that superficially appear correct but do not align with the developer's intents. If developers unintentionally adopt these incorrect code suggestions, it might require longer debugging time and even lead to some significant safety issues.
- **Security implications**. We trained CodeT5 on existing code corpus originally collected from public Github repositories. Pre-trained models might encode some sensitive information (e.g., personal identification numbers) from the training data. Though we have conducted multi-rounds of data cleaning to mitigate this before training our models, it is still possible that some sensitive information was not or cannot be completely removed. Besides, due to the non-deterministic nature of generation models like CodeT5, the model might produce some vulnerable code to harmfully affect the software and even be able to benefit more advanced malware development when deliberately misused.

Caveats and Recommendations

- We recommend that practitioners using CodeT5 in real-world scenarios bear in mind that its generation outputs should be only taken as references and that domain experts be engaged for further correctness- and security-checking.
- We also recommend that the data be further screened to fine-tune CodeT5, including sensitive data cleaning and bias mitigation.
- The model is trained on a limited number of programming languages: primarily Python, Java, JavaScript, PHP, Ruby, Go, C, and C#. A proposed future area of research would be to train the model on more languages.