

CCT5: A Code-Change-Oriented Pre-Trained Model

Bo Lin
linbo19@nudt.edu.cn
College of Computer Science,
National University of Defense
Technology
Changsha, China

Yepang Liu
liuyp1@sustech.edu.cn
Department of Computer Science and
Engineering, Southern University of
Science and Technology
Shenzhen, China

Shangwen Wang
wangshangwen13@nudt.edu.cn
College of Computer Science,
National University of Defense
Technology
Changsha, China

Xin Xia
xin.xia@acm.org
Zhejiang University
Hangzhou, China

Zhongxin Liu
liu_zx@zju.edu.cn
Zhejiang University
Hangzhou, China

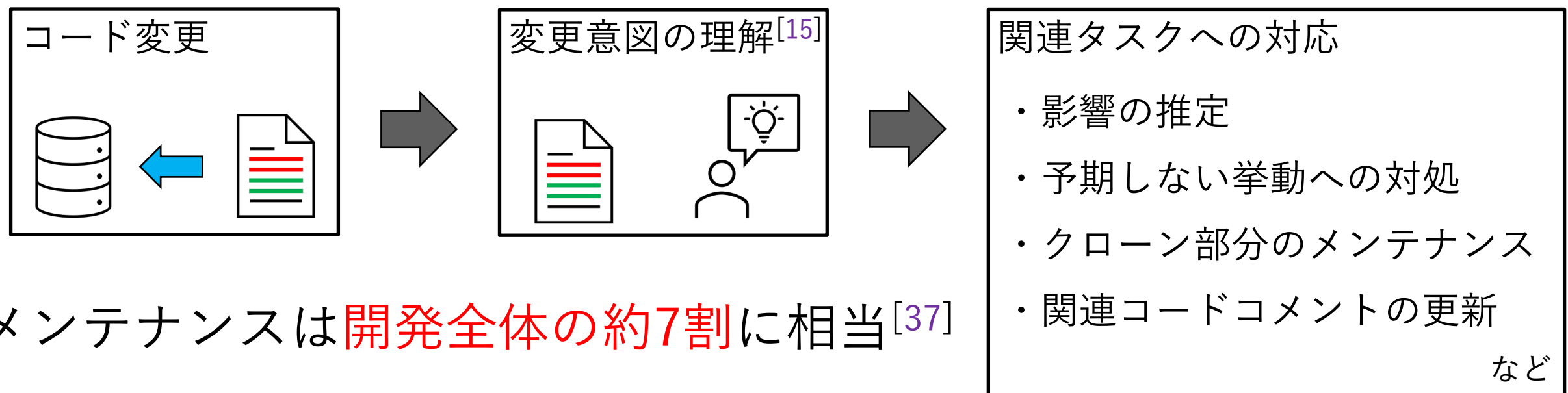
Xiaoguang Mao
xgmao@nudt.edu.cn
College of Computer Science,
National University of Defense
Technology
Changsha, China

In Proceedings of the 31st ACM Joint European Software Engineering Conference and
Symposium on the Foundations of Software Engineering

コード変更に係るタスク支援

ソフトウェアは継続的に変更が繰り返される

- バグ修正, ロジック変更, 処理の効率化, 機能追加 など



メンテナンスは開発全体の約7割に相当^[37]

コード変更に係るタスクへの支援が必要^[61]

- コミットメッセージ生成, 欠陥予測

[15] Thomas Fritz and Gail C Murphy. 2010. Using information fragments to answer the questions developers ask. In 2010 ACM/IEEE 32nd International Conference on Software Engineering, Vol. 1. IEEE, 175–184.

[37] I-H Lin and David A Gustafson. 1988. Classifying software maintenance. In 1988 Conference on Software Maintenance. IEEE Computer Society, 241–247.

[61] Jonathan Sillito, Gail C Murphy, and Kris De Volder. 2008. Asking and answering questions during a programming change task. IEEE Transactions on Software Engineering 34, 4 (2008), 434–451.

コードを扱う深層学習モデル

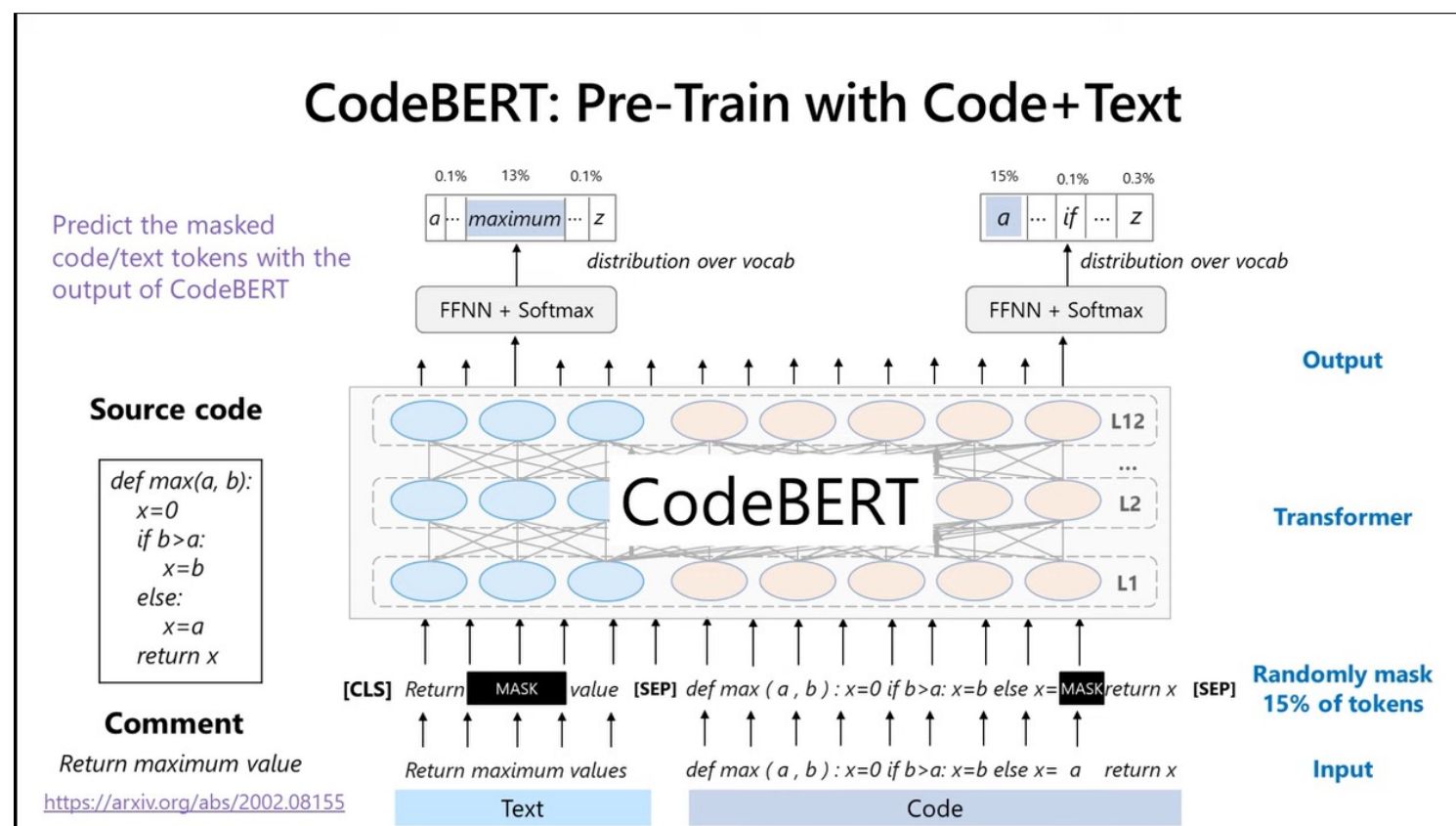
スクラッチモデル

- 対象となるタスクに特化したデータセットを用いて0から学習させるモデル
 - 十分なデータが存在するときには有効

事前学習済みモデル

- 大量のデータを学習し一般的な情報を記憶させるモデル
 - 教師データを必要としない
 - タスクに特化したデータセットでファインチューニングすることを前提
 - e.g. CodeBERT, GraphCodeBERT, CodeT5
- ファインチューニングすることでさまざまなタスクに応用可能
 - コード検索, コード要約, 欠陥検出 など

既存の事前学習済みモデルの問題点



https://miro.medium.com/v2/resize:fit:1400/1*vzOOQKtox7FuUUNvx-8O6w.png

Input :

テキスト

Comment

Return maximum value

コードスニペット

Source code

```
def max(a, b):  
    x=0  
    if b>a:  
        x=b  
    else:  
        x=a  
    return x
```

Architecture :

Transformer

(エンコーダ)

Training Task :

Masked Language Modeling

コードスニペットに関連する情報を学習しているため
変更差分の意味情報が欠落している

- コード変更を入力とするタスクへの応用は最適ではない^[67]

[67] Chaozheng Wang, Yuanhang Yang, Cuiyun Gao, Yun Peng, Hongyu Zhang, and Michael R. Lyu. 2022. No More Fine-Tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (Singapore, Singapore) (ESEC/FSE 2022).

研究目的

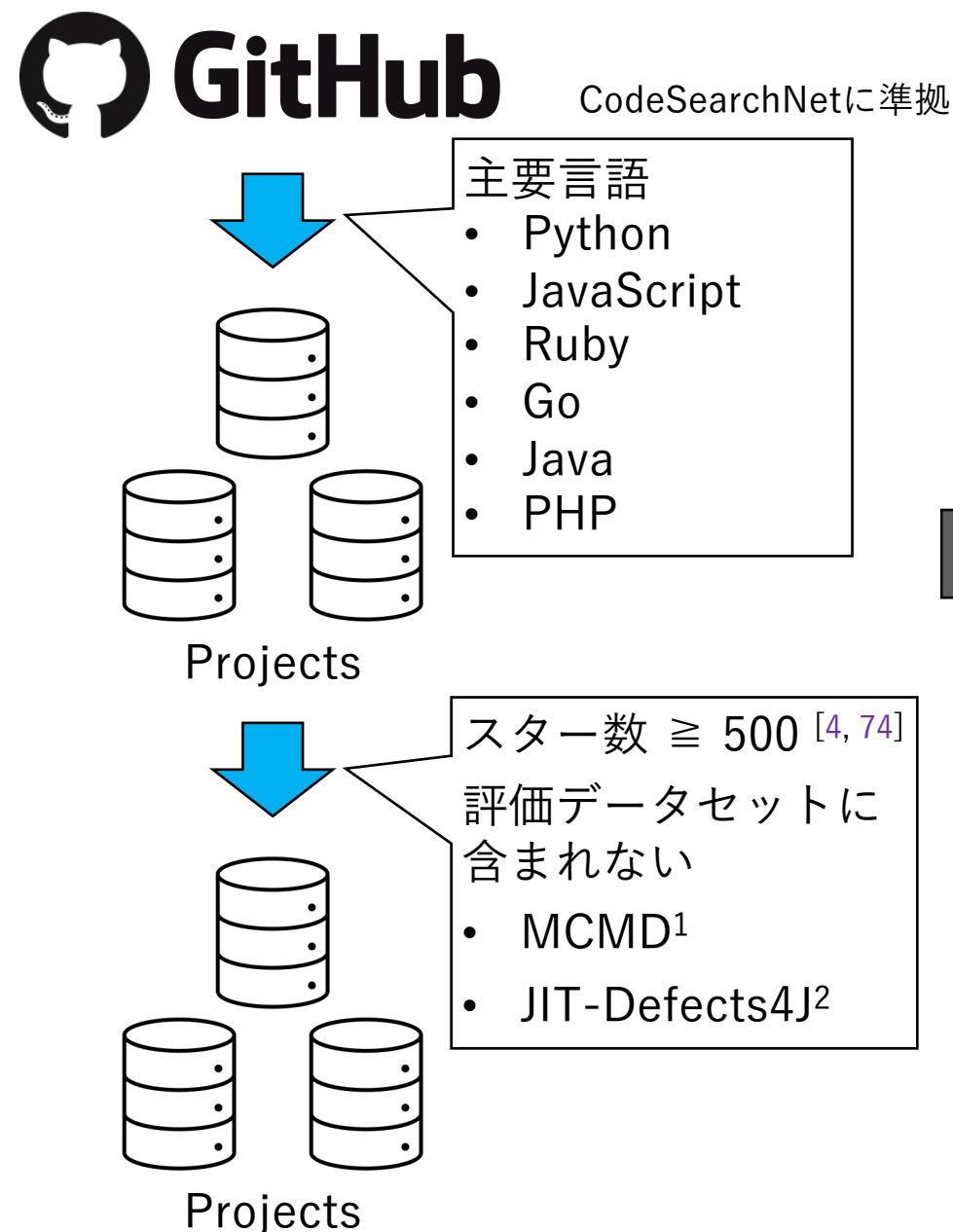
コード変更タスク用の事前学習済みモデル（CCT5）の構築

Code-Change-oriented pre-trained model with T5

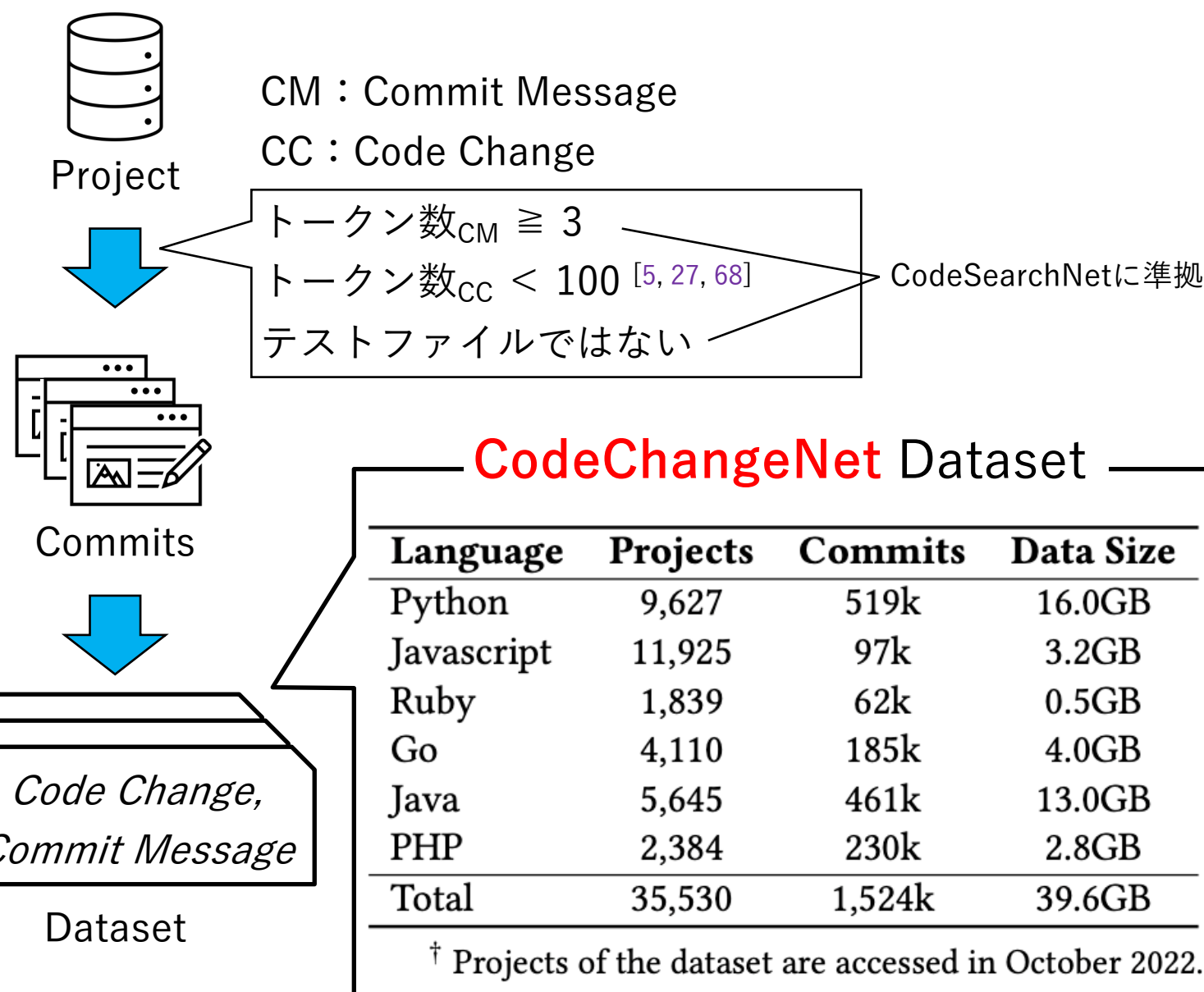
- ベースモデル：T5
 - 各12層のTransformerエンコーダとデコーダ，220Mパラメータ
- 学習データセット：CodeChangeNet
 - 1.5M+ のコード変更とコミットメッセージのデータ
- 学習タスク：5 種類
 - Masked Language Modeling × 2 種類
 - Code Change-Commit Message Dual Generation × 2 種類
 - Code Diff Generation × 1 種類

学習データセット | CodeChangeNet

1. プロジェクトの抽出



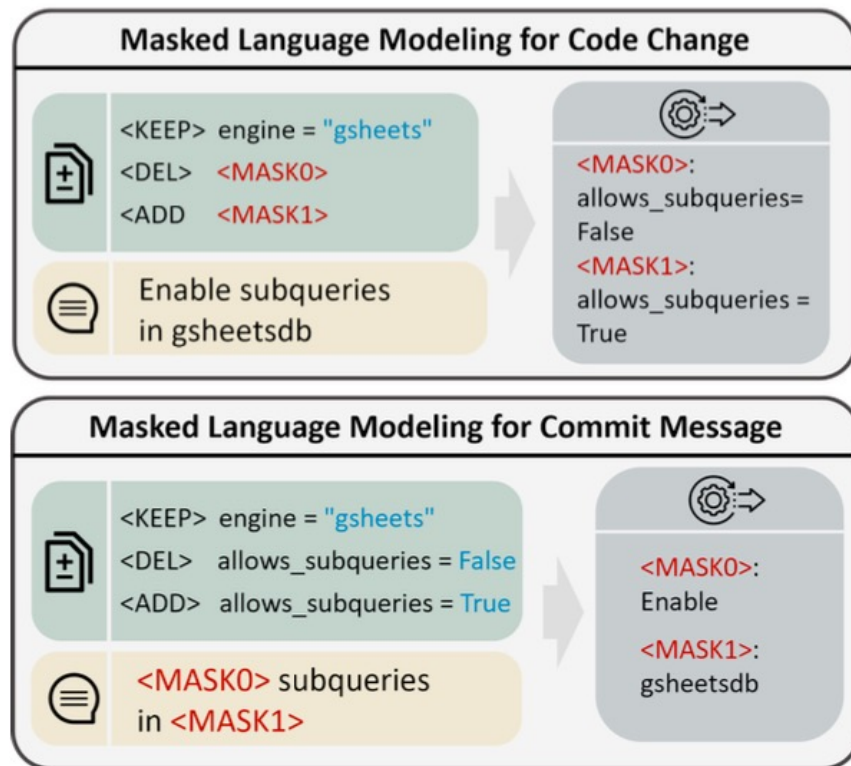
2. コミットの抽出



1. MCM (Multi-programming-language Commit Message Dataset) [63]
2. JIT-Defects4J [50]

CCT5

Masked Language Modeling for Code Change (MLM4CC)



Masked Language Modeling for Commit Message (MLM4CM)

Code Diff Generation (CDG)

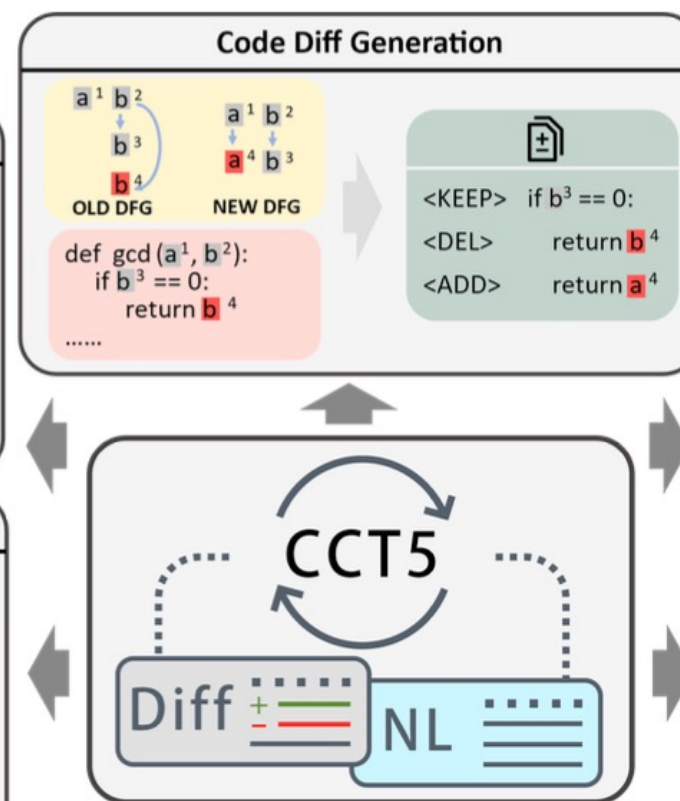
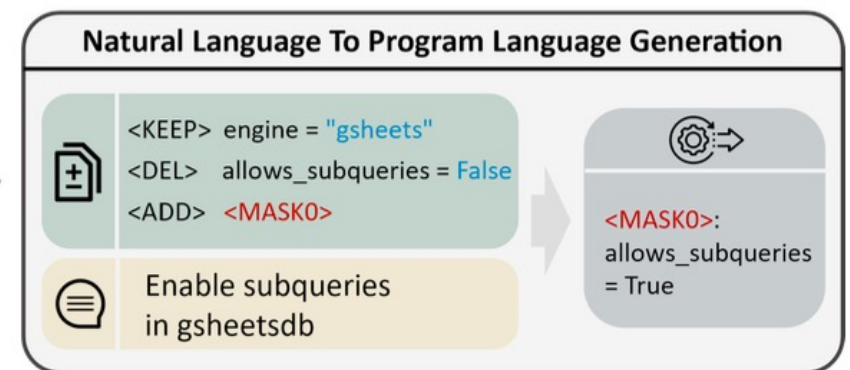
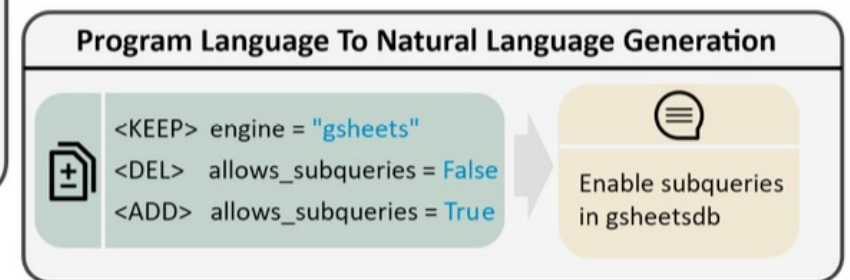


Figure 2: Pre-training tasks of CCT5.

Program Language to Natural Language Generation (PL2NL)

コミットメッセージ生成
コメント更新



Natural Language to Program Language Generation (NL2PL)

プログラム修正
欠陥予測

CCT5 | データフォーマット

基本フォーマット

$\{ [\text{CLS}], \underline{[\text{KEEP}], c_{11}, c_{12}, \dots, [\text{ADD}], c_{21}, c_{22}, \dots [\text{DEL}] , c_{31}, c_{32}, \dots}, \underline{[\text{MSG}], m_1, m_2, \dots} \}$

コード変更
コミットメッセージ

[CLS] : おまじない, 分類タスクの入力列の先頭につける

[KEEP]：変更に影響しないコード行のトークン列の先頭につける

[ADD]：追加コード行のトークン列の先頭につける

[DEL] : 削除コード行のトークン列の先頭につける

[MSG] : コードトークンとメッセージトークンの間につける

トークナイザ

- CodeT5を利用して各コード行を分割
 - ただし、コミットメッセージは不明

CCT5 | 損失関数

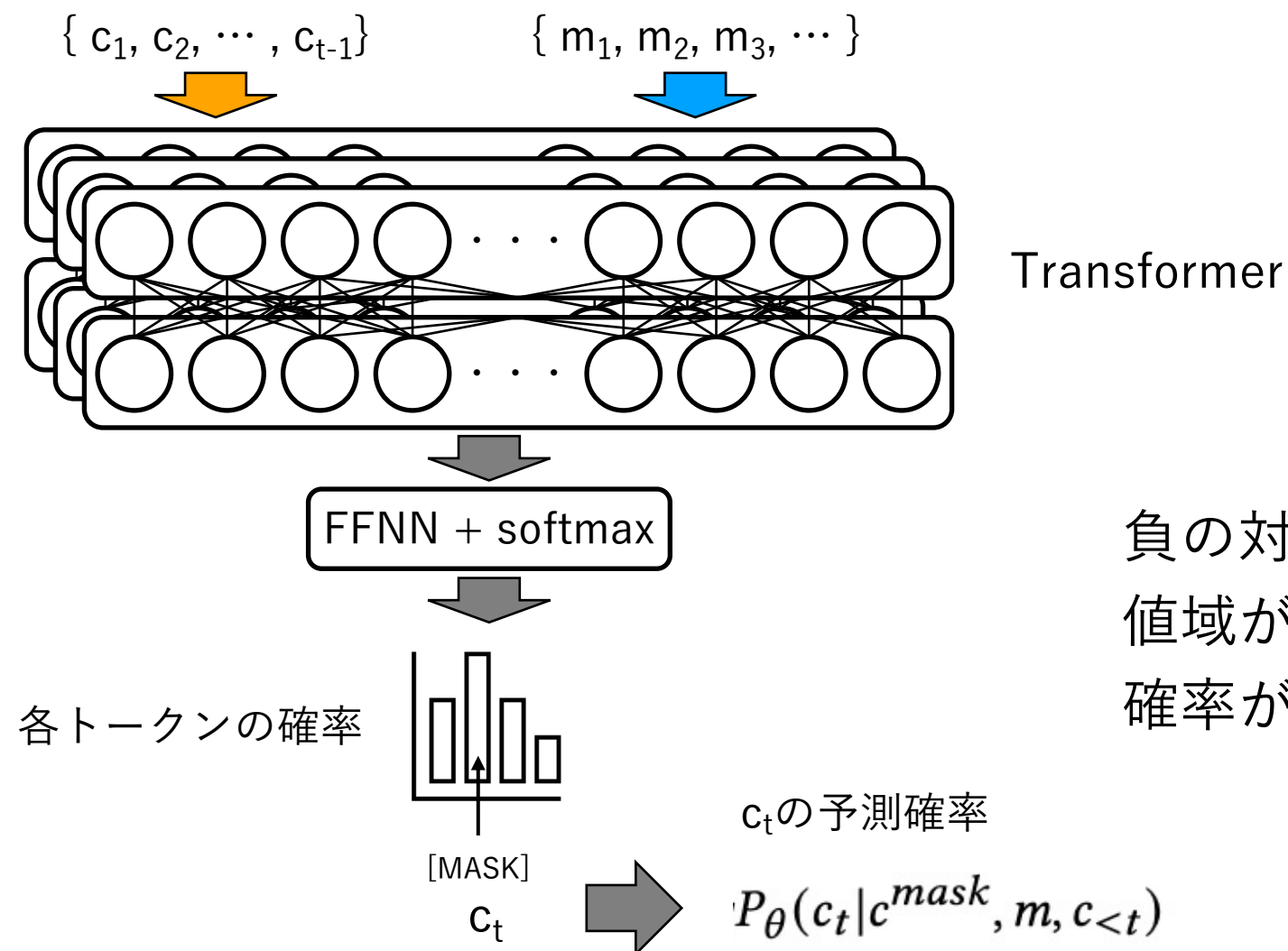
MLM4CC $\mathcal{L}_{MLM4CC}(\theta) = \sum_{t=1}^k -\log P_{\theta}(c_t | c^{mask}, m, c_{<t})$

c_t : t番目のコードトークン

c^{mask} : マスク加工コードトークン列

m : メッセージトークン列

$c_{<t}$: t番目以前のコードトークン列



負の対数に変換することで
値域が $P: 0 \rightarrow 1$ から $-\log P: 0 \rightarrow \infty$ になり
確率が高いときの損失が0になる

CCT5 | 損失関数

MLM4CC	$\mathcal{L}_{MLM4CC}(\theta) = \sum_{t=1}^k -\log P_{\theta}(c_t c^{mask}, m, c_{<t})$	c : コードトークン列 m : メッセージトークン列
MLM4CM	$\mathcal{L}_{MLM4CM}(\theta) = \sum_{t=1}^k -\log P_{\theta}(m_t m^{mask}, c, m_{<t})$	c_t : t 番目のコードトークン m_t : t 番目のメッセージトークン
NL2PL	$\mathcal{L}_{NL2PL}(\theta) = \sum_{t=1}^k -\log P_{\theta}(c_t c^{mask'}, m, c_{<t})$	c^{mask} : マスク加工コードトークン列 m^{mask} : マスク加工メッセージトークン列
PL2NL	$\mathcal{L}_{PL2NL}(\theta) = \sum_{t=1}^k -\log P_{\theta}(m_t c, m_{<t})$	$c_{<t}$: t 番目以前のコードトークン列 $m_{<t}$: t 番目以前のメッセージトークン列
CDG	$\mathcal{L}_{CDG}(\theta) = \sum_{t=1}^k -\log P_{\theta}(c_t oc, odf, ndf, c_{<t})$	oc : 変更前のコードトークン列 odf : 変更前の変数依存関係 ndf : 変更後の変数依存関係

最適化問題（損失最小化問題）

$$\begin{aligned} \min_{\theta} \mathcal{L}_{Loss}(\theta) &= \mathcal{L}_{MLM4CC}(\theta) + \mathcal{L}_{MLM4CM}(\theta) + \mathcal{L}_{NL2PL}(\theta) \\ &\quad + \mathcal{L}_{PL2NL}(\theta) + \mathcal{L}_{CDG}(\theta) \end{aligned}$$

評価（結果省略）

3つのタスクにファインチューニングさせて評価

- Commit Message Generation
- Just-in-Time Comment Update
- Just-in-Time Defect Prediction

結果

- コードスニペットを学習させた事前学習済みモデルと比較してCCT5は良いパフォーマンスを発揮した

まとめ・所感

コード変更に関連したタスク用の事前学習済みモデル CCT5を提案

- 5つのタスクを用いて事前学習させた
- 3つのタスクにファインチューニングし評価をした

所感

- Transformerを用いた事前学習済みモデルの仕組みがだいたい理解できた
 - 複数タスクを用いた事前学習の話は知っていたが、具体的にどのように学習させているのかわからなかったなので、すごくためになった
 - プロンプト然り、基本的にデータの性質に合わせてトークンを作成/挿入し、データに意味付け（パターン化）が重要な気がした