修士学位論文

題目

事前学習済みモデルを用いた
Stack Overflow と言語ドキュメントの紐づけ手法

指導教員
肥後 芳樹 教授

報告者
鬼塚 仙太郎

令和 6 年 2 月 1 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

令和 5 年度 修士学位論文

事前学習済みモデルを用いた Stack Overflow と言語ドキュメントの紐づけ手法

鬼塚 仙太郎

内容梗概

近年，IT の発展に伴う IT 人材の需要の増加や文部科学省による情報教育の推進を背景として，プログラミング学習の需要が増加している．プログラミング学習時に利用する代表的な Web 上のリソースとしては，Stack Overflow と言語ドキュメントがある．Stack Overflow はプログラミングに特化した Q&A コミュニティサイトであり，プログラミング学習中に直面した課題の解決において有益なリソースである．しかし，プログラミング学習者が投稿内容を十分に理解できない可能性がある．特に，プログラミング言語の基本機能はプログラミング学習の初期段階で学ぶ事項であるため，プログラミング言語の基本機能が投稿の理解を阻む場合は，投稿を理解するための追加情報を自ら検索することも難しいと考えられる．言語ドキュメントはプログラミング言語の公式組織が提供しているプログラミング言語の基本機能について記載されたドキュメントのことであり，プログラミング言語の基本機能について学ぶ際に有益なリソースである．しかし，言語ドキュメントだけでは問題解決能力を身に着けることができないことが示唆されており，別のリソースを併用して学習する必要がある．

Stack Overflow と言語ドキュメントは，それぞれ異なる性質の学習リソースであり，互いにないものを補い合う関係にあると考えられる．そのため，プログラミング学習において Stack Overflow を利用する際の問題とプログラミング学習において言語ドキュメントを利用する際の問題の両方を解決する策として，Stack Overflow と言語ドキュメントを紐づけることが役立つと考えられる．Stack Overflow から言語ドキュメントの参照が可能になることで，Stack Overflow の投稿を理解するために必要なプログラミング言語の基本機能を簡単に知ることができるようになる．言語ドキュメントから Stack Overflow の参照が可能になることで，実践的な学習ができるようになり学習の理解度を高められる．

そこで，本研究では Stack Overflow と言語ドキュメントの紐づけ手法を提案する．提案手法では，Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応づいたデータセットで事前学習済みモデルをファインチューニングし，ファインチューニングした事前学習済みモデルを用いて Stack Overflow の投稿からその投稿に関連している言語ドキュメントの項目を推定することで紐づけを実現している．

提案手法の適用可能性を調べるために，提案手法を事前学習済みモデルに CodeBERT を用い，プログラミング言語 Python を対象として適用し紐づけの精度を調査した．適用実験を通じて，提案した Stack Overflow と言語ドキュメントの紐づけ手法がプログラミング言語 Python を対象とした場合では有効であることを確認した．

主な用語

Stack Overflow
言語ドキュメント
プログラミング言語
プログラミング学習
事前学習済みモデル

目 次

1　はじめに

近年，急速な情報技術の進歩による情報化の進展 [33] と同時に，様々な職業や分野で高度 IT 人材と呼ばれる高い IT スキルを持った人材が求められている [32, 34]．また，情報化やグローバル化，AI の進化といった社会変化が人間の予測を超えて進展している予測困難な社会においては，情報や情報技術を受け身で捉えるのではなく手段として活用していく力を育成していくことが求められている [35]．そのような背景から，情報技術を学び，活用する上での基礎となる技術であるプログラミングを学ぶ需要が増加している．

プログラミングの学習方法は多岐にわたるが [25]，どの学習方法においても必要となる情報を自分自身で調べることが必要となる．なぜなら，たとえ同じ題材のプログラミング問題を解いていたとしても直面するエラーは人によって異なっていたり，日々更新される膨大な技術情報をすべて覚えておくことは不可能であり都度検索することが一般的であったりするからである．必要となる情報を自分自身で調べる際に利用する代表的な Web 上のリソースとして Stack Overflow[2] があり，そこにはプログラマからの質問に対して他のプログラマが投稿した解決策や例が回答として掲載されている．

Stack Overflow はプログラミング学習において広く利用されている [19, 20, 25] と同時に，プログラミング学習において有益なリソースである [7, 22]．しかし，Stack Overflow の投稿内容は，投稿内容の要素すべてに対して説明があるわけではない．そのため，投稿では解説されていない要素がプログラミング学習者の知識にない場合は投稿の理解が十分にできない可能性がある．特に，プログラミング言語の基本機能はプログラミング学習の初期段階で学ぶ事項であるため，プログラミング言語の基本機能が投稿の理解を阻む場合は投稿を理解するための追加情報を自ら検索することも難しいと考えられる．本研究におけるプログラミング言語の基本機能とは，制御構造 (プログラミング言語 Python における if 文や for 文など) 及びプログラミング言語標準のデータ構造 (プログラミング言語 Python における list や set など) のことである．

また，エラーなどの特定の問題に対処するためではなく，プログラミング言語の基本機能について学ぶ際に利用する代表的な Web 上のリソースとしては，プログラミング言語の公式組織が提供しているプログラミング言語の基本機能について記載されたドキュメント (以降，言語ドキュメント) がある．言語ドキュメントは公式組織によって提供されている信頼性が高く洗練された情報であり，プログラミング言語の基本機能を一通り学ぶことができる．しかし，言語ドキュメントだけでは問題解決能力を身に着けることができないことが示唆されている [15, 13]．問題解決能力とは，問題の説明を受けそれを小問題に分解して実装し，最終的に 1 つの解決策としてまとめ上げる能力のことであり，高等教育機関では問題解決能力を身に着けた卒業生を輩出することが求められている [17]．そのため，言語ドキュメ

ントだけでなく問題解決能力を身に着けることができる別のリソースを併用して学習する必要がある．

Stack Overflow と言語ドキュメントは，それぞれ異なる性質の学習リソースであり，互いにないものを補い合う関係にあると考えられる．そのため，プログラミング学習において Stack Overflow を利用する際の問題とプログラミング学習において言語ドキュメントを利用する際の問題の両方を解決する策として，Stack Overflow と言語ドキュメントを紐づけることが役立つと考えられる．Stack Overflow から言語ドキュメントの参照が可能になることで，Stack Overflow の投稿を理解するために必要なプログラミング言語の基本機能を簡単に知ることができるようになる．言語ドキュメントから Stack Overflow の参照が可能になることで実践的な学習ができ学習の理解度を高められる．

そこで，本研究では Stack Overflow と言語ドキュメントの紐づけ手法を提案する．提案手法では，Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応づいたデータセットで事前学習済みモデルをファインチューニングし，ファインチューニングした事前学習済みモデルを用いて Stack Overflow の投稿からその投稿に関連している言語ドキュメントの項目を推定することで紐づけを実現している．

提案手法の適用可能性を調べるために，提案手法を事前学習済みモデルに CodeBERT を用い，プログラミング言語 Python を対象として適用し紐づけの精度を調査した．その結果，言語ドキュメントの項目に対応するタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿での評価では $F1\text{-}score@1$ で 0.897 を達成した．言語ドキュメントの項目に対応するタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 2 つの場合と 3 つの場合の投稿での評価では $F1\text{-}score@1$ はそれぞれ 0.917，0.951 を達成し，関連している項目の数が 2 つの時の $F1\text{-}score@2$ は 0.766，関連している項目の数が 3 つの時は $F1\text{-}score@3$ は 0.731 を達成した．また，言語ドキュメントの項目に対応するタグが付与されていない投稿での評価を行い，訓練データの数が十分にない項目をグランドトゥルースから除去した場合では，言語ドキュメントの項目に対応するタグが付与されていない投稿での紐づけの精度と言語ドキュメントの項目に対応するタグが付与された投稿での紐づけの精度が同程度であった．適用実験を通じて，提案した Stack Overflow と言語ドキュメントの紐づけ手法がプログラミング言語 Python を対象とした場合では有効であることを確認した．

以降，2 章では本研究の背景について説明する．3 章では本研究で提案する事前学習済みモデルを用いた Stack Overflow と言語ドキュメントの紐づけ手法について説明する．4 章では提案手法の適用実験の設定について説明し，5 章では実験結果と考察について述べる．6 章では提案手法の限界と妥当性への脅威について述べる．最後に 7 章で，まとめと今後の課題を述べる．

2　背景

2.1　プログラミング学習

近年の急速な情報技術の進歩により，経済 · 社会 · 生活のあらゆる場面で情報化が進展している [33]．様々な産業で IT が活用されており，様々な職業や分野で高度 IT 人材と呼ばれる高い IT スキルを持った人材が求められている [32, 34]．また，情報化やグローバル化，AI の進化といった社会変化が人間の予測を超えて進展している予測困難な社会においては，情報や情報技術を受け身で捉えるのではなく手段として活用していく力を育成していくことが求められている [35]．実際，近年学習指導要領が改定され，中学校だけでなく小学校において本格的な情報教育が行われるようになった [36]．そのような背景から，情報技術を学び，活用する上での基礎となる技術であるプログラミングを学ぶ需要が増加している．

プログラミングの学習方法は学校の講義の受講，オンラインのプログラミング学習コースの受講，本，ハッカソンへの参加など多岐にわたる [25]．しかし，どの学習方法においても，必要となる情報を自分自身で調べることが必要となる．なぜなら，たとえ同じ題材のプログラミング問題を解いていたとしても直面するエラーは人によって異なっていたり，日々更新される膨大な技術情報をすべて覚えておくことは不可能であり都度検索することが一般的であったりするからである．必要となる情報を自分自身で調べる際に利用する代表的な Web 上のリソースとして Q&A コミュニティサイトがある．Q&A コミュニティサイトとは，ある質問者が行った質問に対して不特定多数の回答者が回答を行うサイトである．中でもプログラミングに特化した Q&A コミュニティサイトとして Stack Overflow があり，そこにはプログラマからの質問に対して他のプログラマが投稿した解決策や例が回答として掲載されている．

また，エラーなどの特定の問題に対処するためではなく，プログラミング言語の基本機能について調査したり学んだりする際に利用する代表的な Web 上のリソースとしては，プログラミング言語の公式組織が提供しているプログラミング言語の基本機能について記載されたドキュメント (以降，言語ドキュメント) がある．本研究におけるプログラミング言語の基本機能とは，制御構造 (プログラミング言語 Python における if 文や for 文など) 及びプログラミング言語標準のデータ構造 (プログラミング言語 Python における list や set など) である．

2.2　Stack Overflow

Stack Overflow は 2008 年に設立されたプログラミングに関する質問とそれに対する回答を投稿・検索できる Q&A コミュニティサイトである [2]．2024 年 1 月現在，ユーザ数は約 2,200 万人，投稿は質問と回答を合わせて約 5,900 万件，そして 1 日に約 2300 の質問が投

稿されており広く普及している [1]．Stack Overflow における質問の例を図 1 に示す．Stack Overflow における投稿は，タイトル，質問，複数個のタグ，及び質問に対する複数の回答で構成される．また，各質問及び回答にはコメントを付与できる．質問及び回答では，自然言語による記述 (以降，質問のテキスト) だけでなくコードによる記述 (以降，質問のコード) が可能である．タグとは，質問のトピックを説明する単語またはフレーズとしてユーザが付与するものであり，興味のある質問を特定するためにも使用される [23]．タグとして “Python” 等の言語名や，“if-statement” のような文法事項，“pandas” のようなライブラリ名などが登録されている．また，質問者は回答の中から最も優れたものとしてベストアンサーを選択できる．加えて，ユーザは投稿に対してポジティブ (upvote) またはネガティブ (downvote) のどちらかに投票することができ，投稿に対して投票された upvote と downvote の差は投稿のスコアとしてみなされる．

Stack Overflow はプログラミング学習において広く利用されており，プログラミング学習者は Stack Overflow を利用する可能性が高い．Stack Overflow Developer Survey 2022[25] によると，コーディング方法を学ぶためのオンラインリソースとして Stack Overflow を使用すると答えた人の割合は 86.14% であり，技術文書と並んで最も利用されているリソースの 1 つである．Parnin ら [19] は，プログラミングに関するクエリにおいて，Stack Overflow の投稿が Google の検索結果リストの上位に高い確率で表示されることを示している．Robinson[20] は，大学生が Stack Overflow に投稿した質問の割合が授業期間に大幅に増加し休暇期間に低下することを示している．

Stack Overflow はプログラミング学習のリソースとして有益であることが主張されている．Staton[26] は Stack Overflow を人々が学びに行くべき “new Computer Science Departments” と呼んでいる．また，Dondio ら [7] は，Stack Overflow の教育目的での利用の有効性を評価し，Stack Overflow が従来の講義の教材と同等かむしろ高い効果が得られることを示している．さらに，Singh ら [22] は，プログラミングの教科書で教えられている概念や理論を Stack Overflow の実践的な事例で補強することで，複雑な概念に対する学生の理解度が高まり学習意欲が向上することを示している．

2.3 Stack Overflow 利用時の問題点

Stack Overflow はプログラミング学習者にとって有益なリソースであるが，Stack Overflow の投稿内容は，投稿内容の要素すべてに対して説明があるわけではない．そのため，投稿では解説されていない要素がプログラミング学習者の知識にない場合は投稿の理解が十分にできない可能性がある．その場合は投稿に関連する別のリソースをさらに参照する必要がある．

[1] https://stackoverflow.com/questions/419163/what-does-if-name-main-do

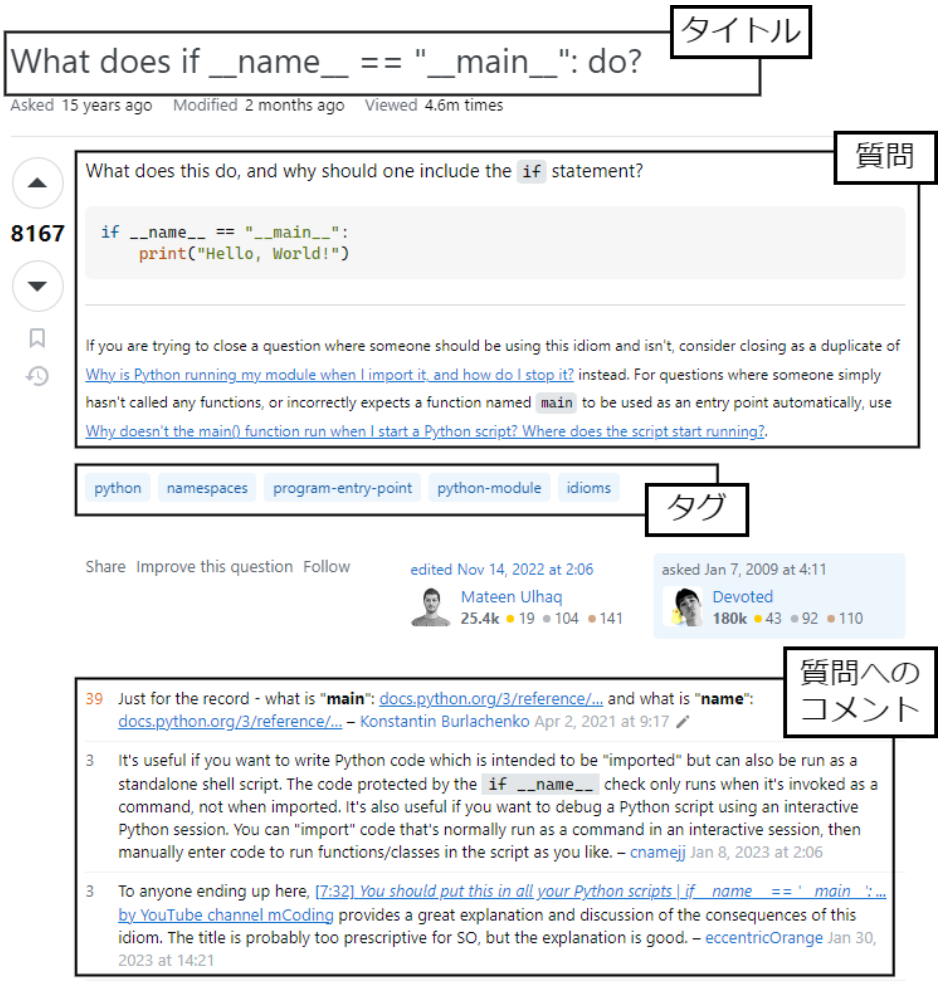

図 1: Stack Overflow における質問の例 (ID: 419163) [1]

Stack Overflow の投稿を理解するために必要な要素としては，API やプログラミング言語の基本機能が挙げられる．投稿を理解するために必要な要素のうち，API については投稿と関連する外部リソースを紐づける研究が存在しており理解支援がされている．具体的には，Stack Overflow の投稿に含まれるコードスニペットから API 要素を特定し，その API に該当する API ドキュメントへのリンクを作成する研究が存在している [27]．それに対し，Stack Overflow の投稿におけるプログラミング言語の基本機能の理解支援を行う研究はほとんどない．特に，プログラミング言語の基本機能はプログラミング学習の初期段階で学ぶ事項であるため，プログラミング言語の基本機能が投稿の理解を阻む場合は投稿を理解するための追加情報を自ら検索することも難しいと考えられる．そのため，プログラミング学習者が Stack Overflow の投稿を理解する上でプログラミング言語の基本機能が原因でつまず

いた場合，自分自身での解決が困難であることが予想される．

2.4 言語ドキュメント

本研究ではプログラミング言語の公式組織が提供している公式ドキュメントのうち，プログラミング言語の基本機能について記載されているドキュメントを言語ドキュメントと呼ぶ．Python[2]や C++[3]，PHP[4]など多くのプログラミング言語において言語ドキュメントが提供されている．言語ドキュメントでは，制御構造やプログラミング言語標準のデータ構造を一通り学ぶことできるようになっている．加えて，多くの言語ドキュメントではサンプルコードが提供されており，読むだけでなく実際に実行することでより理解を深められる．言語ドキュメントは公式組織によって提供されている信頼性が高く洗練された情報であり，また制御構造やプログラミング言語標準のデータ構造を一通り学ぶことができるため，プログラミング言語の基本機能を学ぶのに適しているリソースである．

本研究の適用実験で用いる Python の言語ドキュメント Python Tutorial[3] について説明する．Python Tutorial では 4 章 (制御構造) と 5 章 (データ構造) がプログラミング言語の基本機能に該当する．Python Tutorial の 5 章の項目一覧を図 2 に示す．Python Tutorial の各節のことを本研究では項目と呼ぶ．各項目は図 3 に示す “4.1 if Statements” 項目のように，タイトルと本文で構成される．本文は自然言語による記述 (以降，項目のテキスト) とコードによる記述 (以降，項目のコード) が存在する．

2.5 言語ドキュメント利用時の問題点

言語ドキュメントはプログラミング言語の基本機能を学ぶのに適している一方で，それだけでは問題解決能力を身に着けることができない可能性が高い．問題解決能力とは，問題の説明を受けそれを小問題に分解して実装し，最終的に 1 つの解決策としてまとめ上げる能力のことであり，高等教育機関では問題解決能力を身に着けた卒業生を輩出することが求められている [17]．Koulouri ら [15] は大学レベルのプログラミング入門講座での学習では問題解決能力が不足していることを指摘しており，それと同様のことが言語ドキュメントでの学習においても言える可能性が高い．また，Kember ら [13] は学習意欲を掻き立てる教育環境についての調査を行い，理論とそれに関連する実際の実践的な例との関連性を確立することが学習の動機づけにおいて重要であることを示唆している．これらの調査結果を踏まえると，問題解決能力を身に着けるためには言語ドキュメントだけでなく実践的な例を学ぶことがで

[2] https://docs.python.org/3/tutorial/index.html
[3] https://learn.microsoft.com/ja-jp/cpp/cpp/?view=msvc-160
[4] https://www.php.net/manual/ja/
[5] https://docs.python.org/3/tutorial/controlflow.html#if-statements

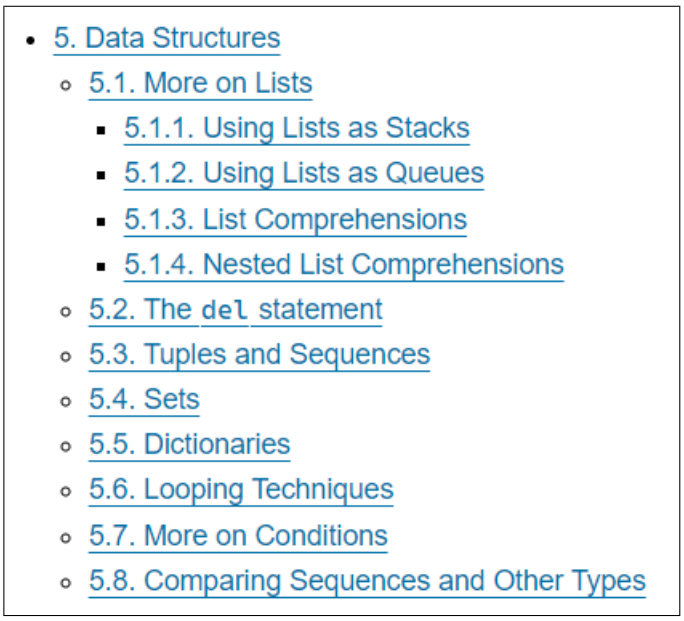

図 2: Python Tutorial の 5 章 (データ構造) の項目一覧

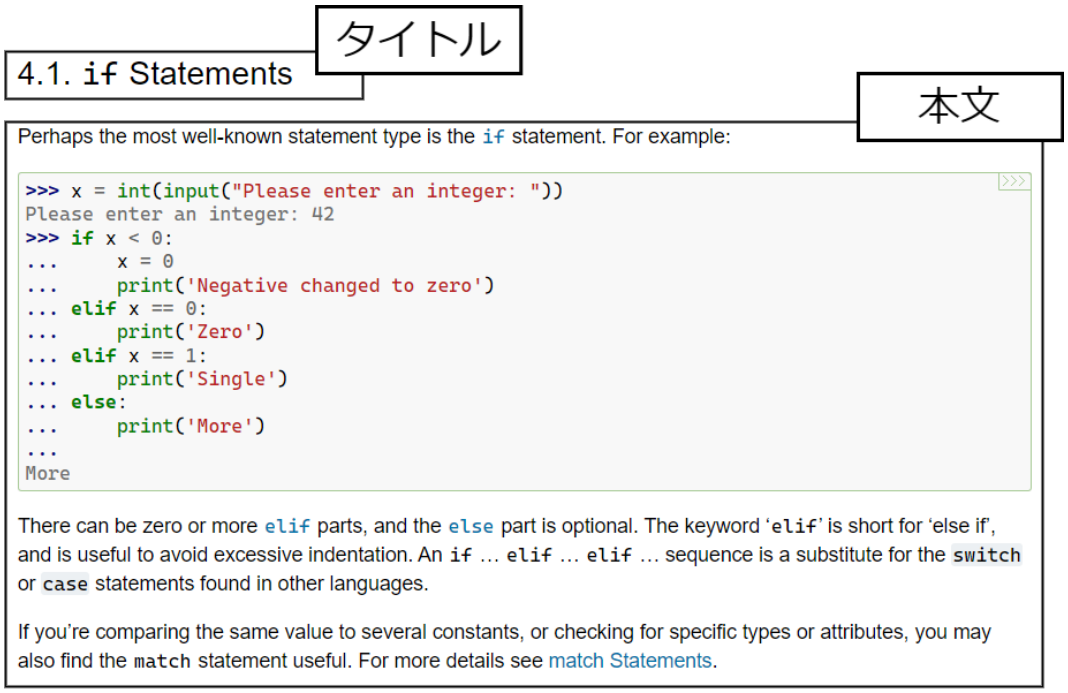

図 3: Python Tutorial の 4.1 節 [5]

きる別のリソースを併用して学習する必要がある．

2.6 Stack Overflow と言語ドキュメントの紐づけ

Stack Overflow と言語ドキュメントは，それぞれ異なる性質の学習リソースであり，互いにないものを補い合う関係にあると考えられる．そこで，2.3 節で述べた Stack Overflow 利用時の問題と 2.5 節で述べた言語ドキュメント利用時の問題の両方を解決する策として，Stack Overflow と言語ドキュメントを紐づけることが役立つと考えられる．両者を紐づけることで双方向での参照が可能となり，それぞれの問題を支援することができる．

2.6.1 Stack Overflow から言語ドキュメントの参照

Stack Overflow から言語ドキュメントの参照が可能になることで，Stack Overflow の投稿を理解するために必要なプログラミング言語の基本機能を簡単に知ることができるようになる．これにより，基本的な知識が定着していない場合やふと忘れた場合だけでなく，投稿を理解するために必要なプログラミング言語の基本機能に関する追加情報を自ら検索することが難しい場合でも，参照すべき外部リソースを容易に知ることができるようになる．

2.6.2 言語ドキュメントから Stack Overflow の参照

言語ドキュメントから Stack Overflow の参照が可能になることで，言語ドキュメントで学びながら関連した Stack Overflow の投稿に触れることができるようになる．Stack Overflow は教育目的での利用が有用であることや [7]，概念や理論を Stack Overflow で補強することで理解度を高めることできることが示されているため [22]，言語ドキュメントを用いたプログラミング言語の基本機能の学習における理解度を高めることができる．また，言語ドキュメントでプログラミング言語の基本機能を学びながら，プログラミング学習において広く利用されている Stack Overflow に慣れることができる．

2.7 紐づけ手法に関する調査

本節では，Stack Overflow と言語ドキュメントの名詞集合間の類似度に関して調査した研究の紹介，キーワードマッチングでの紐づけに関する調査の説明，及び事前学習済みモデルを用いた紐づけに関する調査の説明を行う．

2.7.1 Stack Overflow と言語ドキュメントの名詞集合間の類似度に関する調査

眞鍋 [37] は，Python における for 文と if 文を対象に Stack Overflow の投稿の質問と言語ドキュメントの項目の本文それぞれに登場する名詞集合間の類似度を調査している．こ

の研究では，Stack Overflow の “if-statement” 及び “python” タグが付いた質問のテキストと Python Tutorial の “4.1. if Statements” 項目のテキスト，Stack Overflow の “for-loop” 及び “python” タグが付いた質問のテキストと Python Tutorial の “4.2. for Statements” 項目のテキストを対象として，Stack Overflow の投稿の質問と言語ドキュメントの項目の本文それぞれの類似したトピック間で同じ名詞が登場しているかを調査している．その結果，for 文の場合は Stack Overflow と言語ドキュメントの紐づけに適する名詞が現れたものの，if 文の場合は現れなかったことがわかっている．この調査結果から，Stack Overflow と言語ドキュメントの紐づけは単純なキーワードマッチングでは達成できない可能性が高いことがわかる．

2.7.2 キーワードマッチングでの紐づけに関する調査

キーワードマッチングでの紐づけに関する調査として，Stack Overflow においてタグが質問のテキストに出現する要素を特徴づけたものになっているかを調査した [31]．調査対象の単語としては，Python Tutorial の項目のテキストで使用されていた名詞から，その項目に関連していてかつ Stack Overflow のタグとして存在する単語を選択した．具体的には，言語ドキュメントの項目として “4.2. for Statements”，“5.4. Sets” 及び “8.2. Exceptions” を選択し，各項目のテキストから調査対象の単語としてそれぞれ “iteration”，“set”，“exception” を選択した．選択した単語に関して調査を行った結果，プログラミング言語の基本機能に関するタグは質問のテキストの頻出単語でなく，また質問のテキストのプログラミング言語の基本機能に関する頻出単語はタグとして付与されないことがわかった．この調査結果から，単純なキーワードマッチングでは Stack Overflow の投稿から投稿に関連するプログラミング言語の基本機能を抽出することができないことがわかる．

2.7.3 事前学習済みモデルを用いた紐づけに関する調査

眞鍋の調査 [37] 及び 2.7.2 節で述べたキーワードマッチングでの紐づけに関する調査の結果から，キーワードマッチングによって Stack Overflow と言語ドキュメントを紐づけることが難しいことがわかる．そこで，キーワードマッチングのような単語単位ではなく，文章単位で Stack Overflow の投稿や言語ドキュメントの項目の特徴を捉えることが有効ではないかと考えた．文章単位で特徴を捉えるには機械学習を用いることが一般的であるため，機械学習を用いた紐づけ手法についての検討を行う．

機械学習を用いた紐づけ手法の検討として，CodeBERT を用いて Stack Overflow の投稿と言語ドキュメントの項目の類似度を調査した．2.8 節で紹介する事前学習済みモデル CodeBERT はソフトウェア工学関連のタスクで実用的であることが示されているため [11,

18, 30]，調査で使用する機械学習モデルとして選択した．類似度は，CodeBERT を用いて Stack Overflow の投稿の質問と言語ドキュメント項目の本文を分散表現に変換し，それらのコサイン類似度により算出した．分散表現への変換方法は以下の 3 つの手順で構成される．

1. CodeBERT を用いてテキストとコードそれぞれの単語埋め込みを生成する．
2. 単語埋め込みに対して Average Pooling を適用して文章埋め込みを生成する．
3. 生成されたテキストの文章埋め込みとコードの文章埋め込みを結合して 1 つの分散表現とする．

Stack Overflow 上の 3 つの項目 (制御構造 for 文, データ構造 set, 例外処理) それぞれに関連する投稿の質問と，Python Tutorial[3] の 3 つの項目 (“4.2. for Statements”，“5.4. Sets”，“8.3. Handling Exceptions”) ぞれぞれの本文との類似度を調査した．調査対象の Stack Overflow の投稿としては，各項目と関連した Stack Overflow のタグやキーワードを用いて検索して見つかったものの中からランダムに抽出した．調査結果を表 1(コサイン類似度は小数第三位で四捨五入) に示す．調査結果を見ると，どの質問においても “4.2. for Statements” 項目との類似度が最も高くなっている．このことから，CodeBERT を用いたコサイン類似度では Stack Overflow の投稿と言語ドキュメントの項目の関係を適切にとらえることができておらず紐づけが実現できないことがわかる．

この原因の 1 つとして，事前学習済みモデルである CodeBERT をファインチューニングせずに使用していることが考えられる．特定のタスクで CodeBERT を使用するには，そのタスクに応じたデータセットでファインチューニングする必要がある [6, 8]．つまり，機械学習を用いて Stack Overflow と言語ドキュメントの紐づけを実現するには，紐づけに適したデータセットを作成してファインチューニングを行う必要があると考えられる．

2.8 CodeBERT

CodeBERT[8] はトークンベースの事前学習済みモデルである．その特徴として，自然言語とプログラミング言語のマルチモーダルな情報を事前学習しており，両方の形式のデータを理解できる．CodeBERT はコードサーチ [11, 30] や自動プログラム修正 [18] などの様々なソフトウェア工学関連のタスクで実用的であることが示されている．

特定のタスクで CodeBERT を使用する際には，そのタスクに応じたデータセットでファインチューニングする必要がある [6, 8]．CodeBERT をダウンストリームタスクに適用した研究をいくつか紹介する．

- Zhou ら [30] は，コードサーチタスクに CodeBERT を適用することで，コードサーチに特化したモデルと比較して Mean Reciprocal Rank 指標の点で 31 %〜38 %上回る

ことを示している．評価は 2 つのデータセット (テキストとコードがペアになったデータセット) で行っている．ファインチューニングに用いた訓練データ数は，それぞれ 51176 個と 67864 個である．

- He ら [10] は，Stack Overflow のタグ推薦に CodeBERT を適用することで，最先端のアプローチであった深層学習ベースのタグ推薦アプローチを大差で上回ることを示している．また，Stack Overflow のタグ推薦において 5 つの事前学習済みモデル (BERT, RoBERTa, ALBERT, CodeBERT, BERTOverflow) での性能を比較し，CodeBERT が最高性能を達成することを示している．CodeBERT のファインチューニングの訓練データとしては，100000 件の Stack Overflow の投稿を用いている．

- JIN ら [12] は，コードサーチタスクに CodeBERT を適用し，ファインチューニングに使用するデータ数を変更して性能を調査している．データセットとして 80723 のコメントとコードのペアを作成し，そのデータセットのうちの 1%，5%，10%，20%，50%，及び 100%を用いてファインチューニングを行った場合の性能を調査している．その結果，ファインチューニングに使用するデータが増加するにつれて性能が上がることを示しており，ファインチューニングに使用するデータ数が多いほど特定のダウンストリームタスクに適合させられることを示唆している．また，データセットの 1%だけを

表 1: Stack Overflow の投稿の質問と言語ドキュメントの項目のコサイン類似度

言語ドキュメントの項目	項目と関連する質問の ID	各項目とのコサイン類似度		
		4.2 節	5.4 節	8.3 節
4.2. for Statements	1663807	0.91	0.84	0.80
	3162271	0.94	0.87	0.83
	2990121	0.89	0.82	0.77
	14785495	0.88	0.80	0.76
5.4. Sets	59825	0.89	0.82	0.77
	9792664	0.91	0.83	0.79
	17457793	0.86	0.79	0.73
	29648520	0.89	0.82	0.77
8.3. Handling Exceptions	6470428	0.92	0.89	0.83
	1483429	0.82	0.76	0.74
	843277	0.88	0.82	0.76
	16511337	0.89	0.80	0.79

ファインチューニング使用した場合でも，ファインチューニングなしの CodeBERT と比較すると劇的に精度が向上することを示しており，少量のデータでのファインチューニングでも特定のタスクに適応する能力が大幅に向上することを示唆している．

3　提案手法

本章では事前学習モデルを用いた Stack Overflow と言語ドキュメントの紐づけ手法を提案する．提案手法では Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応づいたデータセットで事前学習済みモデルをファインチューニングし，ファインチューニングした事前学習済みモデル用いて Stack Overflow の投稿からその項目に関連している言語ドキュメントの項目を推定することで紐づけを実現している．提案手法の概要図を図 4 に示す．提案手法は 2 つの STEP で構成されている．

STEP1: データセットの作成

この STEP では，Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応付けられたデータセットを作成する．まず，言語ドキュメントの各項目からその項目に対応する Stack Overflow のタグ (以降，ターゲットタグ) を選定し，そのタグをもとに Stack Overflow の投稿を抽出する．その後，Stack Overflow の投稿と言語ドキュメントの項目を対応付けることでデータセットを作成する．

STEP2: 事前学習済みモデルを用いた Stack Overflow と言語ドキュメントの紐づけ

この STEP では，事前学習済みモデルを用いて Stack Overflow の投稿からその投稿との関連度の高い言語ドキュメントの項目を推定する．事前学習済みモデルのファインチューニングには STEP1 で作成したデータセットを用いる．

3.1　STEP1: データセットの作成

Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応付けられたデータセットの作成手法は 3 つの STEP で構成されている．

STEP1-1: ターゲットタグの選定

STEP1-1 では，言語ドキュメントの項目と Stack Overflow のタグの集合を入力としてターゲットタグを選定する．ターゲットタグの選定の流れを図 5 に示す．ターゲットタグの選定における各構成要素の詳細は以下の通りである (箇条書きの番号と図 5 の各構成要素の番号が対応している)．

1. タイトルのトークナイズとレンマ化
   言語ドキュメントの項目のタイトルを単語に分割し各単語をレンマ化する．レンマ化とは，文脈と意味を考慮して単語を辞書の基本形に切り詰めることである．

2. ストップワードの除去
   得られた単語からストップワードを除外する．ストップワードは一般的な文章におい

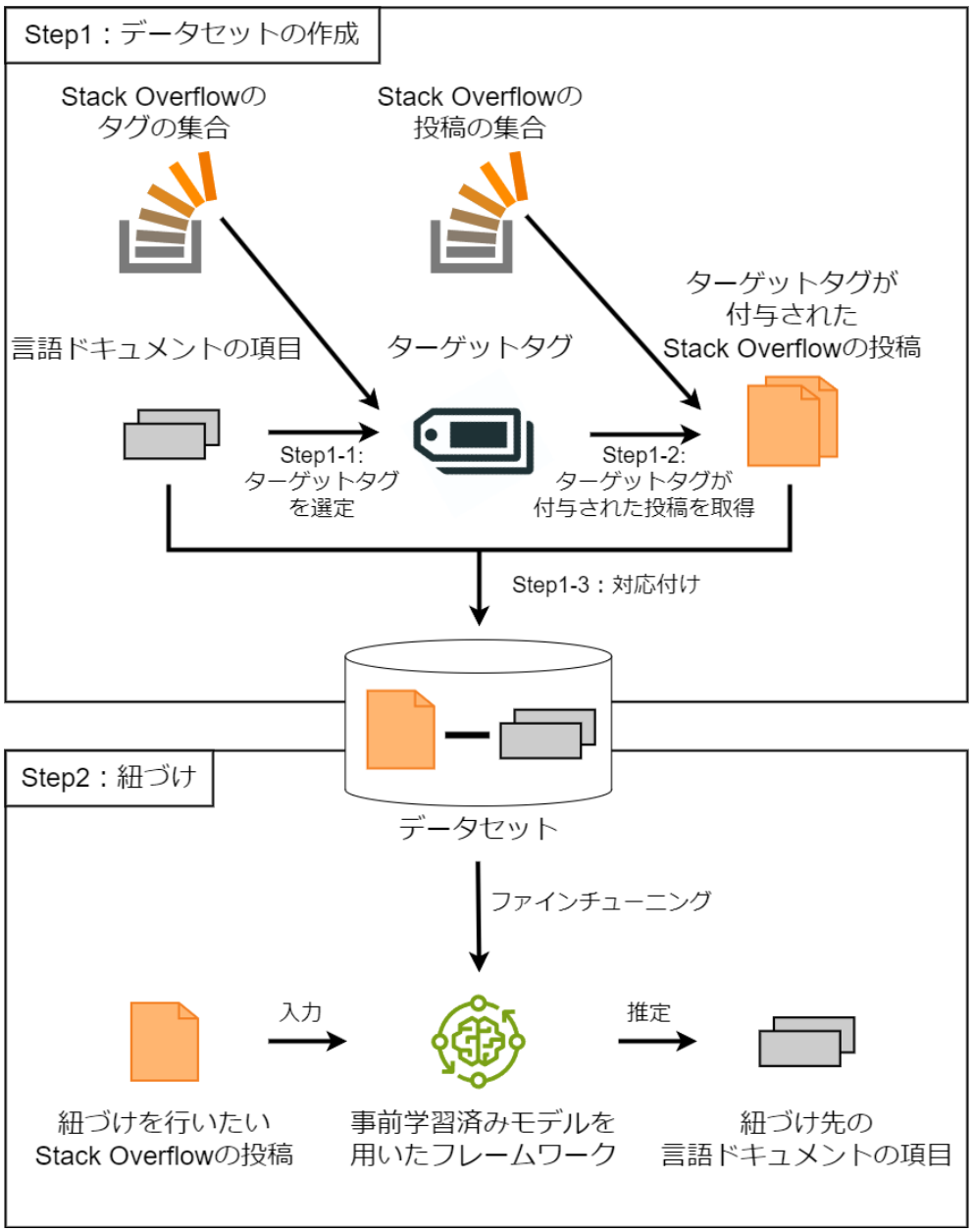
Step1：データセットの作成
Stack Overflowの
タグの集合
Stack Overflowの
投稿の集合
言語ドキュメントの項目
ターゲットタグ
ターゲットタグが
付与された
Stack Overflowの投稿
Step1-1:
ターゲットタグ
を選定
Step1-2:
ターゲットタグが
付与された投稿を取得
Step1-3：対応付け
データセット
Step2：紐づけ
ファインチューニング
入力
推定
紐づけを行いたい
Stack Overflowの投稿
事前学習済みモデルを
用いたフレームワーク
紐づけ先の
言語ドキュメントの項目

図 4: 提案手法の概要図

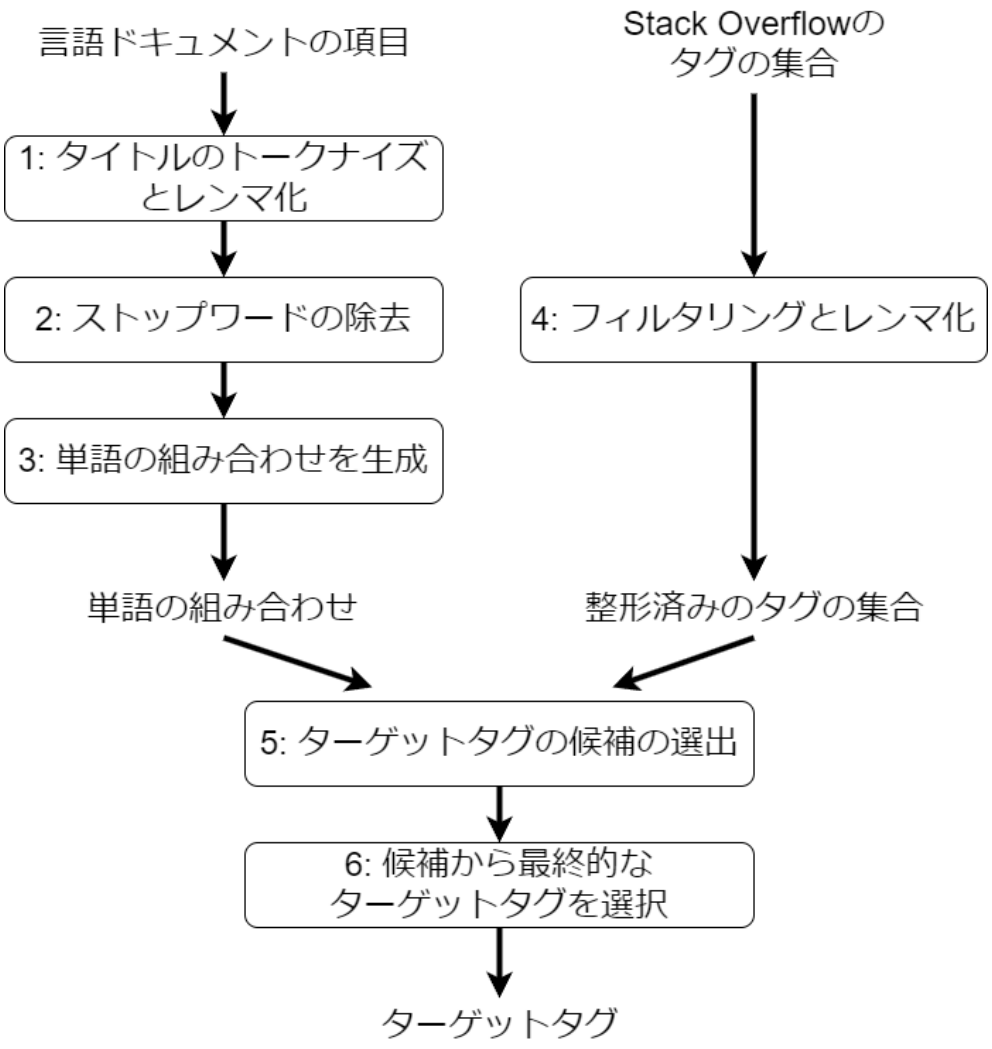

図 5: STEP1-1(ターゲットタグの選定) の流れ

て出現頻度が高くあまり意味を持たない単語のことである．ただし，タグに必要な単語やプログラミング言語において意味を持つ単語 (例: for, if) が失われるのを防ぐために，Stack Overflow のタグとプログラミング言語の予約語で使用されている単語はストップワードから除外する．

3. 単語の組み合わせを生成
   Stack Overflow のタグは 1 つの単語だけでなく複数の単語を含むことができる (例: sql-server) ため，得られた単語から 1~3 単語を組み合わせてハイフンで結合した unigram, bigram, trigram を生成する．組み合わせる際の単語の前後関係は全通りを考慮して生成する．また，動詞であればその動詞に対応する名詞 (例: define であれば definition) で置き換えた場合についても考慮する．例えば，得られた単語が “define” と “function” であった場合，組み合わせとして “define-function”，“definition-function”，“function-define”，“function-definition” の 4 つを生成する．

4. フィルタリングとレンマ化
   Stack Overflow のタグの集合に以下の 3 つの条件でフィルタリングを適用する．

   - 出現頻度の低いタグはユーザに広く認識されていない不適当なタグである可能性が高いと考えられるため，出現頻度が閾値より小さいタグを除外する．
   - Stack Overflow のタグにはそのタグの説明として tag wiki を付与できる．タグに説明がない場合，そのタグが意図したとおりに使用されない不適当なタグである可能性が高いと考えられるため，tag wiki が存在しないタグを除外する．
   - 対象とするプログラミング言語と関係のないタグを除外するため，対象とするプログラミング言語のタグ (例えば Python であれば “python” タグ) と一緒に質問に付与されたことがないタグを除外する．

   また，タグの表記ゆれに対応するためにフィルタリングしたタグの集合に含まれる各タグにはレンマ化を適用する．

5. ターゲットタグの候補の選出
   4 でフィルタリングとレンマ化を適用したタグの集合に含まれるタグのうち，3 で生成した単語の組み合わせと一致するものをターゲットタグの候補として選出する．比較の際には Stack Overflow におけるタグの同義語 (tag synonyms[24]) を考慮する．例えばタグ “for” はタグ “for-loop” の同義語として定義されているため，3 で生成された単語が “for” の場合にはタグ “for-loop” と一致する．

6. 候補から最終的なターゲットタグを選択
   5 で選出したタグの候補が複数ある場合以下の基準に従ってどのタグを選択するかを決定する．

   - 候補のタグがタイトルにおいて and または or で列挙された並列構造の場合 (3 つ以上の場合はカンマで区切られることもある) はそれぞれを選択する．
   - 候補のタグがタイトルにおいて並列構造となっていない場合は，それらの候補のうち他のドキュメントの項目のタグの候補として挙がっていないものを優先する．それでも複数の候補がある場合はその項目において重要度が高いタグを優先する．ただし，優先されたタグが付いた投稿と，優先されたタグと優先されなかったタグの両方が付いた投稿の内容を数件目視で確認し，言語ドキュメントの項目に対応するタグとしてどちらが使われているかを確認して適切なほうを選出する．

     項目におけるタグの重要度の指標としては，項目の本文に対する tf-idf[21] を用いる．ただし，ドキュメントの意味的アノテーションタスクにおいてドキュメントの本文を考慮せずタイトルだけを用いても有効である [9] ように，言語ドキュメントのタイトルに登場する単語がその項目において特徴的な単語である可能性が高いことがわかっている．そのことから，項目の本文で登場する回数 (tf) はそれほど重要とならないと考えられるため，項目の本文中に同じ単語が極端に繰り返し出現した場合に大きく影響を受ける一般的な tf-idf ではなく，tf には対数で計算した $1 + log(tf)$ を用いる．

STEP1-2: ターゲットタグが付与された Stack Overflow の投稿を取得

STEP1-2 では，STEP1-1 で選定されたターゲットタグを用いて Stack Overflow の投稿の集合からターゲットタグが付与された投稿を取得する．

STEP1-3: Stack Overflow の投稿と言語ドキュメントの項目の対応付け

STEP1-3 では，STEP1-2 で取得した Stack Overflow の各投稿と，その投稿を取得する際に使用したターゲットタグの選定元となった言語ドキュメントの項目を対応付ける．これにより，Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応付けられたデータセットを作成する．

3.2 STEP2: 事前学習済みモデルを用いた Stack Overflow の投稿と言語ドキュメントの項目の紐づけ

事前学習済みモデルを用いた Stack Overflow の投稿と言語ドキュメントの項目の紐づけは，He ら [10] の事前学習済みモデルを用いたタグ推薦フレームワークを応用することで実現する．

3.2.1 タグ推薦フレームワーク

He らの事前学習済みモデルを用いたタグ推薦フレームワークの概要図を図 6 に示す．このフレームワークは前処理，特徴抽出，分類の 3 つのステージで構成されている．前処理のステージでは，Stack Overflow の投稿をタイトル，質問のテキスト，質問のコードの 3 つの要素に分解し，その後事前学習済みモデルを用いて各要素をトークン化する．特徴抽出のステージでは，前処理されたデータを各事前学習済みモデルに入力して各要素 (タイトル，質問のテキスト，質問のコード) の特徴ベクトルを得る．その後，得られた特徴ベクトルをプーリング及び結合し投稿のベクトル表現を構築する．分類のステージでは，タグ予測器により投稿のベクトル表現をもとに各タグと関連している確率を出力する．

このフレームワークは，訓練データとして Stack Overflow の投稿 $x_i(1 \leq i \leq N)$ と x_i のグランドトゥルースタグ y 与えられたとき，式 1 の目的関数 $\mathcal{L}$ を最小化することで訓練される．$f(y_j|x_i)$ はタグ y_j が投稿 x_i に関連している確率である．

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} y_j \times \log(f(y_j|x_i)) + (1 - y_j) \times \log(1 - f(y_j|x_i)) \tag{1}$$

目的関数 $\mathcal{L}$ は 2 値交差エントロピー誤差をとらえ，バックプロパゲーションを用いた勾配降下によって最適化される．勾配は各要素 (タイトル，質問のテキスト，質問のコード) の事前学習済みモデルにも伝播し，事前学習済みモデルのパラメータは訓練プロセスにおいて更新される．

3.2.2 タグ推薦フレームワークの応用方法

このタグ推薦フレームワークを応用して，入力が Stack Overflow の投稿，出力が各言語ドキュメントの項目と関連している確率となるようにする．そのために，既存のフレームワークから以下 2 点を変更する．

- 正解ラベルの集合を Stack Overflow のタグから言語ドキュメントの項目に変更する．これにより，出力が各言語ドキュメントの項目と関連している確率となる．

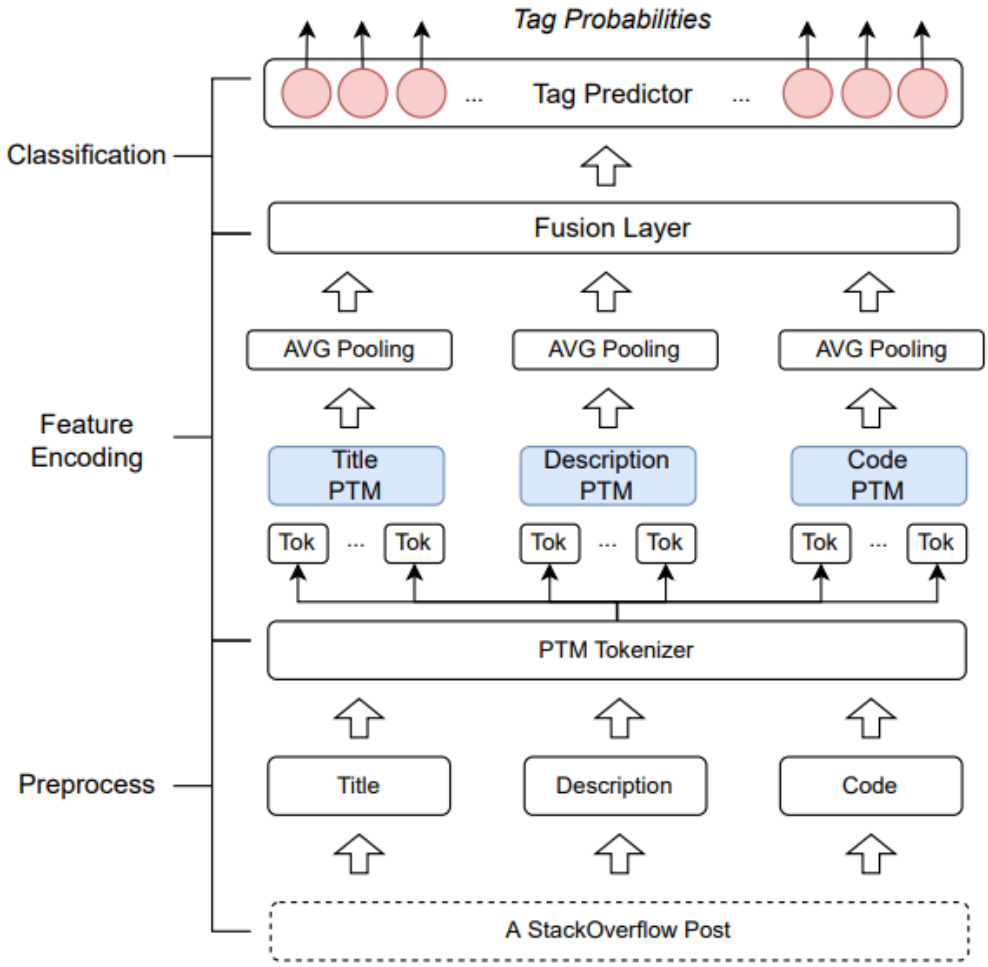

図 6: He らの事前学習済みモデルを用いたタグ推薦フレームワークの概要図 ([10] より引用)

- 出力を変更しているため，ファインチューニングに使用するデータセットの変更も必要となる．ファインチューニングに使用するデータセットを，Stack Overflow の投稿とタグが対応づいたデータセットから，STEP1 で作成した Stack Overflow の投稿とその投稿に関連する言語ドキュメントの項目が対応づいたデータセットに変更する．

変更したフレームワークを用いることで Stack Overflow の投稿を入力すると各言語ドキュメントの項目との関連度が出力されるため，出力された関連度の高い項目を入力した Stack Overflow の投稿と紐づける．

4　実験設定

提案手法に基づく Stack Overflow と言語ドキュメントの紐づけの精度を評価するために適用実験を実施した．提案手法では，事前学習済みモデルのファインチューニングに用いる訓練データはターゲットタグに基づいて作成している．そのため，適用実験ではターゲットタグが付与された投稿での評価だけでなく，ターゲットタグが付与されていない投稿での評価も実施する．以降，適用実験で対象としたプログラミング言語，言語ドキュメント，訓練データ，評価データ，評価指標，実験環境及び事前学習済みモデルについて説明する．

4.1　プログラミング言語と言語ドキュメント

実験対象のプログラミング言語としては Python を選択した．Python の言語ドキュメントとしては Python Tutorial[3] が該当する．Python Tutorial のうち，プログラミング言語の基本機能に該当する 4 章 (制御構造) と 5 章 (データ構造) の計 27 項目からプログラミング言語の基本機能と関係のない 3 項目を除いた 24 項目を対象とした．除いた 3 項目は “4.9. Intermezzo: Coding Style”，“5.6. Looping Techniques” 及び “5.8. Comparing Sequences and Other Types” である．

4.2　訓練データ

提案手法のデータセットの作成における STEP1-1 の手順に従い，言語ドキュメントの各項目のターゲットタグを選定した結果を表 2 に示す．表の「選定されたターゲットタグ」列の&区切りで書かれているタグは，&区切りのすべてのタグで 1 つのタグのセットであり，STEP1-2 においてタグのセットに含まれるすべてのタグが付与された投稿が取得される．表の「著者の評価」列は，選定されたタグが適切であるかを著者の経験とタグの tag wiki をもとに著者が確認した結果である．選定されたターゲットタグが適切であると評価した場合は「○」を書いている．適切でないと評価した場合は「×」を書いている．適切ではないと評価した場合の括弧内には，より適切なタグが見つかった場合はそのタグを書いており，見つからなかった場合はハイフンを書いている．

データセット作成の STEP1-2 の手順として，STEP1-1 の手順で適切なターゲットタグを選定できたと判断した 20 項目のうち，“4.6. match Statements ” 項目を除いた 19 項目を対象としてそのタグを持つ Stack Overflow の投稿を取得した．“4.6. match Statements” に該当するプログラミング言語の基本機能は，2021 年の 10 月にリリースされた Python 3.10 で新しく追加された機能であり，本実験で用いる SOTorrent[4](Version 2020-12-31) ではその機能に関する投稿を取得できないため対象外とした．SOTorrent は Stack Overflow の公式データダンプをもとに構築されたオープンデータセットである．

STEP1-3 では，STEP1-2 で取得した Stack Overflow の各投稿と，その投稿を取得する際に使用したターゲットタグの選定元となった言語ドキュメントの項目を対応付けた．結果として得られた言語ドキュメントの項目ごとの Stack Overflow の投稿数を表 3 に示す．表の「重複を除いた投稿数」とは，他の言語ドキュメントの項目とも関連している投稿を除いた投稿，つまり単一の言語ドキュメントの項目のみに関連している投稿の数である．単一の

表 2: Python Tutorial の各項目に対するターゲットタグの選定結果

言語ドキュメントの項目	選定されたターゲットタグ	著者の評価
4.1. if Statements	if-statement	○
4.2. for Statements	for-loop	○
4.3. The range() Function	range & function	○
4.4. break and continue Statements, and else Clauses on Loops	break, continue, if-statement & loops	○
4.5. pass Statements	statements	×(−)
4.6. match Statements	match	○
4.7. Defining Functions	function-definition	○
4.8.1. Default Argument Values	default-arguments	○
4.8.2. Keyword Arguments	keyword-argument	○
4.8.3. Special parameters	parameters	×(−)
4.8.4. Arbitrary Argument Lists	arguments	×(−)
4.8.5. Unpacking Argument Lists	argument-unpacking	○
4.8.6. Lambda Expressions	lambda	○
4.8.7. Documentation Strings	documentation	× (docstring)
4.8.8. Function Annotations	function & annotations	○
5.1.1. Using Lists as Stacks	stack	○
5.1.2. Using Lists as Queues	queue	○
5.1.3. List Comprehensions	list-comprehension	○
5.1.4. Nested List Comprehensions	list-comprehension & nested-lists	○
5.2. The del statement	del	○
5.3. Tuples and Sequences	tuples, sequence	○
5.4. Sets	set	○
5.5. Dictionaries	dictionary	○
5.7. More on Conditions	conditional-statements	○

言語ドキュメントの項目のみに関連している投稿を項目ごとに 3:1 で訓練データと評価データに分割した．複数の言語ドキュメントの項目に関連している投稿については投稿数が少ないため訓練データに含めずすべて評価データとした．2 つの項目と関連している投稿の数が 4934，3 つと関連している投稿のが 284 であった．

データセット作成において用いた技術は以下の通りである．

- レンマ化には Python ライブラリの NLTK[5] で提供されている WordNetLemmatizer を用いた．
- ストップワードについては NLTK で用いられているストップワードのリストを用いた．
- 出現頻度が低いタグを除外するための閾値には 50 を設定した．この閾値 50 は Stack

表 3: データセット

言語ドキュメントの項目	投稿数	重複を除いた投稿数
4.1. if Statements	6,377	4,729
4.2. for Statements	11,662	9,631
4.3. The range() Function	57	46
4.4. break and continue Statements, and else Clauses on Loops	991	379
4.7. Defining Functions	6	6
4.8.1. Default Argument Values	52	48
4.8.2. Keyword Arguments	616	540
4.8.5. Unpacking Argument Lists	84	61
4.8.6. Lambda Expressions	3,553	3,158
4.8.8. Function Annotations	13	13
5.1.1. Using Lists as Stacks	457	409
5.1.2. Using Lists as Queues	1,442	1,381
5.1.3. List Comprehensions	4,445	3,477
5.1.4. Nested List Comprehensions	51	0
5.2. The del statement	162	133
5.3. Tuples and Sequences	6,251	5,069
5.4. Sets	2,465	1,974
5.5. Dictionaries	34,580	31,866
5.7. More on Conditions	1,225	928

Overflow のタグ推薦の既存研究 [29, 16, 10] に従った．

- Stack Overflow のデータ (質問のテキスト，質問のコード，タグ，tag wiki) の取得には SOTorrent[4](Version 2020-12-31) を利用した．SOTorrent は Stack Overflow の公式データダンプをもとに構築されたオープンデータセットである．
- Stack Overflow におけるタグの同義語 (tag synonyms) の取得には Stack Overflow が提供する API を用いた．

4.3 評価データ

評価データとしては以下の 3 種類の Stack Overflow の投稿を用いる．

- 1 種類目: ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿 (以降，TAG_SINGLE 評価データ)
- 2 種類目: ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が複数ある投稿 (以降，TAG_MULTI 評価データ)
- 3 種類目: ターゲットタグが付与されていない投稿 (以降，NO_TAG 評価データ)

評価データの 1 種類目と 2 種類目はターゲットタグが付与された投稿であり，4.2 節の訓練データの作成で得られたデータセットのうち訓練データに使用しなかった投稿である．訓練データにはターゲットタグが付与された投稿のうち投稿に関連している言語ドキュメントの項目が 1 つである投稿のみを用いたため，評価データとしては投稿に関連している言語ドキュメントの項目が 1 つである場合と複数ある場合で区別している．評価データの数は，TAG_SINGLE 評価データが 15971 である．TAG_MULTI 評価データについては，2 つの項目と関連している投稿の数が 4886 であり，3 つと関連している投稿の数が 283 である．

評価データの 3 種類目はターゲットタグが付与されていない投稿である．事前学習済みモデルのファインチューニングに用いた訓練データはターゲットタグを用いて作成した．そのため，評価データの 1 種類目と 2 種類目だけでは，ターゲットタグが付与されていない投稿における紐づけの精度を評価することができない．そこで，ターゲットタグが付与されていない投稿に対して，投稿と投稿に関連する言語ドキュメントの項目が対応づいた評価データ (NO_TAG 評価データ) を作成した．NO_TAG 評価データは，言語ドキュメントの各項目と関連したキーワードを用いて検索して見つかった投稿の中から，ターゲットタグが付与されていない投稿をいくつか抽出して内容を目視で確認し，その投稿とその投稿に関連する言語ドキュメントの項目を対応付けることで作成した．NO_TAG 評価データの数は 134 である．NO_TAG 評価データにおける言語ドキュメントの項目ごとの Stack Overflow の投稿数を表

4 に示す．1 つの投稿が複数の投稿と関連している場合もあるため，表の「投稿数」列の合計は 134 にはならない．“4.2. for Statements” や “4.7. Defining Functions” は，その項目ではなく他の項目に関する投稿を探して見つかった投稿においても関連していることが多かったため投稿数が多くなっている．“5.1.1. Using Lists as Stacks” や “5.1.2. Using Lists as Queues” はその項目に関する投稿が見つからなかったため少なくなっている．NO_TAG 評価データのうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿の数は 81，2 つの項目と関連している投稿の数が 45，3 つと関連している投稿の数が 6，4 つと関連している投稿の数が 2 である．

表 4: NO_TAG 評価データ

言語ドキュメントの項目	投稿数
4.1. if Statements	14
4.2. for Statements	28
4.3. The range() Function	15
4.4. break and continue Statements, and else Clauses on Loops	9
4.7. Defining Functions	21
4.8.1. Default Argument Values	11
4.8.2. Keyword Arguments	11
4.8.5. Unpacking Argument Lists	9
4.8.6. Lambda Expressions	11
4.8.8. Function Annotations	5
5.1.1. Using Lists as Stacks	2
5.1.2. Using Lists as Queues	3
5.1.3. List Comprehensions	16
5.1.4. Nested List Comprehensions	4
5.2. The del statement	4
5.3. Tuples and Sequences	10
5.4. Sets	9
5.5. Dictionaries	10
5.7. More on Conditions	5

4.4 紐づけの精度の評価指標

事前学習済みモデルを用いたStack Overflowの投稿と言語ドキュメントの項目の紐づけの精度を評価する指標としては，$Precision@k$，$Recall@k$，$F1\text{-}score@k$の3つを用いた．これらはタグ推薦において一般的に使用されている指標である[10, 16, 29]．Stack Overflowの投稿$x_i(1 \leq i \leq N)$が与えられたとき，各投稿における$Precision@k_i$，$Recall@k_i$，$F1\text{-}score@k_i$の平均がそれぞれ$Precision@k$，$Recall@k$，$F1\text{-}score@k$として計算される．

$Precision@k$は，モデルが推定した関連度の高い上位kのドキュメントの項目のうちのグランドトゥルースの割合である．投稿x_iのグランドトゥルースをGT_i，モデルによって投稿x_iから推定された関連度の高い上位kのドキュメントの項目を$Item_i^k$とすると$Precision@k_i$は式2で計算され，$Precision@k$は式3で計算される．

$$Precision@k_i = \frac{|GT_i \cap Item_i^k|}{k} \tag{2}$$

$$Precision@k = \frac{\sum_{i=1}^{N} Precision@k_i}{N} \tag{3}$$

$Recall@k$は，グランドトゥルースのうち正しく予測されたグランドトゥルースの割合である．$Recall@k_i$は式4で計算され，$Recall@k$は式5で計算される．

$$Recall@k_i = \begin{cases} \frac{|GT_i \cap Item_i^k|}{k} & (|GT_i| > k) \\ \frac{|GT_i \cap Item_i^k|}{|GT_i|} & (|GT_i| \leq k) \end{cases} \tag{4}$$

$$Recall@k = \frac{\sum_{i=1}^{N} Recall@k_i}{N} \tag{5}$$

$F1\text{-}score@k$は$Precision@k$と$Recall@k$の調和平均であり，$F1\text{-}score@k_i$は式6で計算され，$F1\text{-}score@k$は式7で計算される．

$$F1\text{-}score@k_i = 2 \times \frac{Precision@k_i \times Recall@k_i}{Precision@k_i + Recall@k_i} \tag{6}$$

$$F1\text{-}score@k = \frac{\sum_{i=1}^{N} F1\text{-}score@k_i}{N} \tag{7}$$

4.5 実験環境

適用実験は，2.60GHz Intel(R)Xeon(R) E5-2623 v4 CPU，16GB NVIDIA Tesla P100 GPU上で実施した．

4.6 事前学習済みモデル

Stack Overflow の投稿から言語ドキュメントの項目を推定するために用いる事前学習済みモデルとしては CodeBERT[8] を用いた．これは，タグ推薦フレームワークにおいて 5 つの事前学習済みモデル (BERT, RoBERTa, ALBERT, CodeBERT, BERTOverflow) の中で CodeBERT が最高性能を達成しているからである [10]．CodeBERT のファインチューニングでは，バッチサイズを 32，エポック数を 5 に設定し，オプティマイザとして Adam[14] を利用した．また，初期学習率を 7E-05 に設定し線形スケジューラを適用した．ファインチューニングに要した時間は約 8 時間であった．

CodeBERT が処理できる最大トークン数は 512 であるため，512 トークンを超える入力に対してはトランケーションが必要になる．本実験では先頭の 510 トークン (先頭に挿入される $[CLS]$ トークンと末尾に挿入される $[SEP]$ トークンを除く) を入力トークンとみなす head-only トランケーション [28] を用いた．また，特徴ベクトルのプーリングには Average Pooling を用いた．

5　実験結果と考察

本章では，4.2 節で述べた訓練データでファインチューニングしたモデルを用いて，4.3 節で述べた 3 種類の評価データに対して紐づけの精度を評価した結果と，その結果に基づく考察について述べる．

5.1　ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿での評価

5.1.1　結果

ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿に対して紐づけの精度を評価した．評価には TAG_SINGLE 評価データを用いた．評価に用いた投稿数は 15971 であり，紐づけに要した時間は計約 10 分であった．各 epoch における $F1\text{-}score@1$ の結果 (小数第 4 位以下を切り捨て) を表 5 に示す．k が 1 の場合は $Precision@k$，$Recall@k$ 及び $F1\text{-}score@k$ が一致するため $F1\text{-}score@1$ のみを示す．結果を見ると，epoch1 で 0.897 と最高性能となっていることがわかる．

また，最高性能を達成した epoch1 のモデルを用いて言語ドキュメントの項目ごとに評価を行った．その結果 (小数第 4 位以下を切り捨て) を表 6 に示す．表において，“5.1.4. Nested List Comprehensions” はその項目だけに関連している投稿が存在しなかったため結果がない．結果を見ると，“4.2. for Statements” での 0.896 や “5.5. Dictionaries” での 0.962, “5.1.2. Using Lists as Queues” での 0.896 のように高精度を達成している項目があることが確認できる．一方で，“4.3. The range() Function”，“4.7. Defining Functions”，“4.8.1. Default Argument Values”，“4.8.5. Unpacking Argument Lists” 及び “4.8.8. Function Annotation” での 0 のように精度が低い項目も存在していることがわかる．

5.1.2　考察

実験結果から言語ドキュメントの項目によって紐づけの精度にばらつきがあることがわかった．本実験で訓練に用いたデータ数には項目ごとにばらつきがあり，それが実験結果に影響している可能性がある．そこで，訓練データの数と $F1\text{-}score@1$ に関係があるのかを

表 5: TAG_SINGLE 評価データでの評価結果

epoch	1	2	3	4	5
$F1\text{-}score@1$	0.897	0.889	0.875	0.815	0.822

調査した．訓練データの数と $F1\text{-}score@1$ の関係を表した散布図を図 7 に示す．散布図の各点は，各項目の訓練データの数と $F1\text{-}score@1$ に対応する．散布図を見ると，ある程度訓練データの数があると訓練データの数と $F1\text{-}score@1$ の間に相関は見られないが，訓練データの数が少ない場合に $F1\text{-}score@1$ が極端に低くなることがわかる．訓練データの数が 450 以下の項目に絞った散布図を図 8 に示す．散布図を見ると，訓練データの数が 50 以下になると $F1\text{-}score@1$ が 0 となることがわかる．このことから，訓練データの数は少なくとも 100 程度は必要になることがわかる．このような訓練データの数が少ない項目に対しては，その項目に関連する複数の投稿を組み合わせて新たな投稿を作成するなどのデータ拡張を行い訓練データの数を増やすことで精度が向上する可能性がある．

紐づけの精度の向上が見込める方法としては以下が挙げられる．

表 6: TAG_SINGLE 評価データでの項目ごとの評価結果

言語ドキュメントの項目	$F1\text{-}score@1$
4.1. if Statements	0.828
4.2. for Statements	0.896
4.3. The range() Function	0
4.4. break and continue Statements, and else Clauses on Loops	0.347
4.7. Defining Functions	0
4.8.1. Default Argument Values	0
4.8.2. Keyword Arguments	0.851
4.8.5. Unpacking Argument Lists	0
4.8.6. Lambda Expressions	0.809
4.8.8. Function Annotations	0
5.1.1. Using Lists as Stacks	0.844
5.1.2. Using Lists as Queues	0.896
5.1.3. List Comprehensions	0.792
5.1.4. Nested List Comprehensions	-
5.2. The del statement	0.558
5.3. Tuples and Sequences	0.876
5.4. Sets	0.833
5.5. Dictionaries	0.962
5.7. More on Conditions	0.437

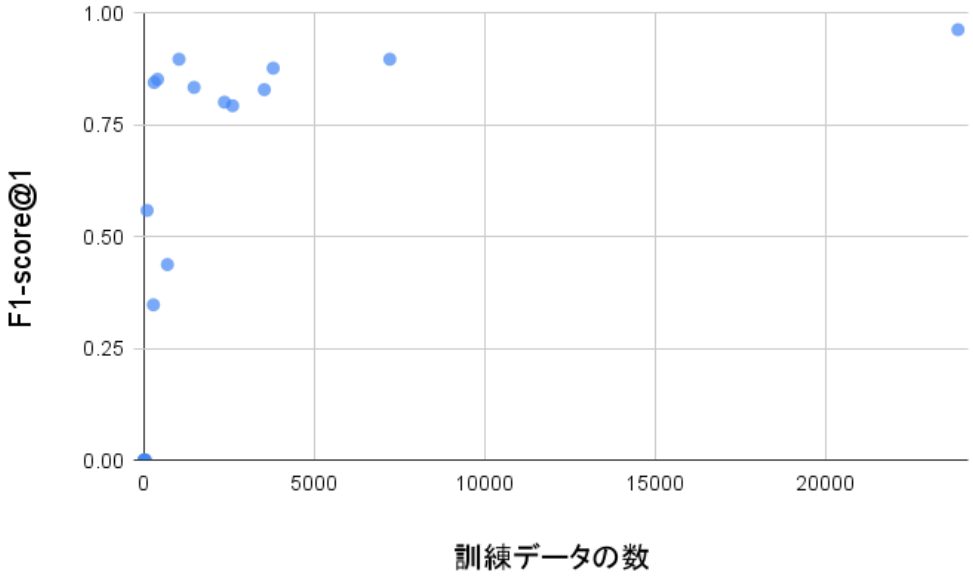

図 7: 訓練データの数と $F1$-$score$ の関係

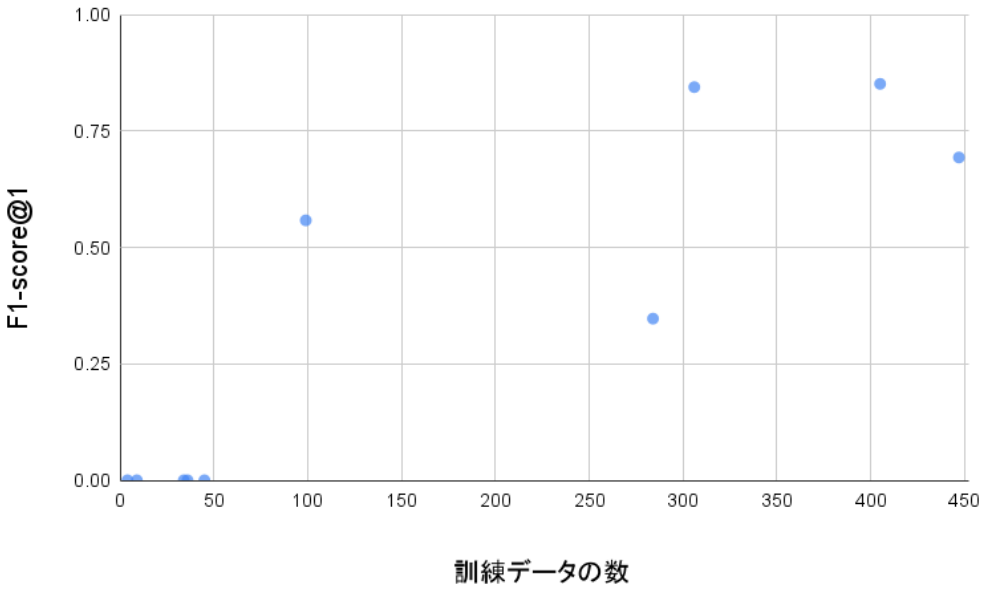

図 8: 訓練データの数と $F1$-$score$ の関係 (訓練データの数を 450 以下に限定)

- 本実験ではファインチューニングにおいてハイパーパラメータのチューニングを行っていないため，ハイパーパラメータのチューニングにより紐づけの精度が向上する可能性がある．

- 本実験では Stack Overflow の投稿の要素のうち 3 つの要素 (タイトル，質問のテキスト，質問のコード) のみを使用している．そのため，Stack Overflow の投稿の質問へのコメント，回答のテキスト，回答のコード，回答へのコメントなどの要素を考慮することで紐づけの精度が向上する可能性がある．

- 本実験ではトランケーションとして head-only トランケーションを用いた．その他のトランケーション手法として，末尾の 510 トークンを使用する tail-only トランケーション [28] や，先頭の 128 トークンと末尾の 382 トークンを使用する head+tail トランケーション [28]，Hierarchical method[28] などがあるため，それらを利用することで精度が向上する可能性がある．

- 本実験ではプーリング手法として Average Pooling を用いた．その他のプーリング手法として Max Pooling や CodeBERT の特殊トークンである $[CLS]$ トークンのベクトルを用いる方法があるため，それらを利用することで精度が向上する可能性がある．

5.2 ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が複数ある投稿での評価

5.2.1 結果

ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が複数ある投稿に対して紐づけの精度を評価した．ファインチューニング済みのモデルには，5.1 節の評価で最高性能を達成した epoch1 のモデルを用いた．評価には TAG_MULTI 評価データを用いた．TAG_MULTI 評価データのうち，投稿に関連している言語ドキュメントの項目が 2 つの投稿の数は 4886，3 つの投稿の数は 283 であり，紐づけに要した時間はそれぞれ計約 3 分，計約 10 秒であった．評価結果 (小数第 4 位以下を切り捨て) を表 7 に示す．$k = 1, 2$ のときは $Precision@k$，$Recall@k$ 及び $F1\text{-}score@k$ が一致するため $F1\text{-}score@k$ のみを示す．結果を見ると，複数の項目のうちの 1 つの項目と紐づける精度を表す $F1\text{-}score@1$ では，関連している項目の数が 2 つの時は 0.917，関連している項目の数が 3 つの時は 0.951 を達成している．関連している項目すべてと紐づける精度については，関連している項目の数が 2 つの時の $F1\text{-}score@2$ は 0.766，関連している項目の数が 3 つの時の $F1\text{-}score@3$ は 0.731 を達成している．

5.2.2 考察

訓練データには複数の項目に関連している投稿が含まれていなかったことを考慮すると，投稿に関連している言語ドキュメントの項目が 1 つである投稿での学習が，複数の項目に関連している投稿における紐づけにも有効であることがわかる．

訓練データの数を十分に確保できなかったことから，投稿に関連している言語ドキュメントの項目が複数ある投稿は訓練データに含まれていなかった．そのため，複数の項目に関連している投稿のデータセットを構築してそれを訓練データに含めることで精度向上が見込める．例えば，複数の投稿を組み合わせて 1 つの投稿として扱うなどのデータ拡張を行うことで複数の項目に関連している投稿のデータ数を増やし，それらで学習することが考えられる．

5.3 ターゲットタグが付与されていない投稿での評価

5.3.1 結果

ターゲットタグが付与されていない投稿に対して紐づけの精度を評価し，その結果をターゲットタグが付与された投稿に対する紐づけの精度と比較した．ターゲットタグが付与されていない投稿における評価には NO_TAG 評価データを用いた．ターゲットタグが付与された投稿の評価データとしては，TAG_SINGLE 評価データと TAG_MULTI 評価データを合わせた評価データ (以降，TAG 評価データ) を用いる．ファインチューニング済みのモデルには，5.1 節の評価で最高性能を達成した epoch1 のモデルを用いた．NO_TAG 評価データでの評価結果と TAG 評価データでの評価結果を表 8 に示す．結果を見ると，NO_TAG 評価データでの $F1\text{-}score@k$ は TAG 評価データでの結果と比較すると低いことがわかる．$F1\text{-}score@1$ では 0.171 低く，$F1\text{-}score@2$ では 0.121 低く，$F1\text{-}score@3$ では 0.083 低い．

しかし，NO_TAG 評価データにおける各言語ドキュメントの項目の割合と，TAG 評価データにおける各言語ドキュメントの項目の割合が異なる．特に，5.1 節の言語ドキュメントの項目ごとでの評価において $F1\text{-}score@1$ が 0 であった 5 つの項目 (以降，低精度項目) の評価データの割合が，NO_TAG 評価データでは 36.56%であるのに対し，TAG 評価データでは 0.38%である．このため，低精度項目に関連する投稿の評価データが，ターゲットタグ

表 7: TAG_MULTI 評価データでの評価結果

関連している項目の数	$F1\text{-}score@k$				$Precision@k$		$Recall@k$	
	$F@1$	$F@2$	$F@3$	$F@4$	$P@3$	$P@4$	$R@3$	$R@4$
2	0.917	0.766	0.570	0.456	0.684	0.609	0.855	0.913
3	0.951	0.872	0.731	0.613	0.731	0.700	0.731	0.817

が付与された投稿での評価にはほとんど影響しないのに対して，ターゲットタグが付与されていない投稿での評価には大きく影響する．そこで，低精度項目をグランドトゥルースから削除した NO_TAG 評価データ及び TAG 評価データを用いて，ターゲットタグが付与された投稿と付与されていない投稿それぞれでの紐づけの精度を評価した．評価結果を表 9 に示す．低精度項目をグランドトゥルースから削除したことで低精度項目のみに関連している投稿が評価データから除去されるため，TAG 評価データの数は 21140 から 21094 になり，NO_TAG 評価データの数は 134 から 106 になった．結果を見ると，NO_TAG 評価データでの評価結果と TAG 評価データでの評価結果が，どの評価指標においても同程度であることがわかる．また，低精度項目をグランドトゥルースから削除していない評価データでの評価結果 (表 8) と比べると，NO_TAG 評価データの結果はどの評価指標でも高くなるのに対し，TAG 評価データでの評価結果はほとんど変化していないことがわかる．

5.3.2 考察

ターゲットタグが付与された投稿に対する紐づけの精度は，グランドトゥルースから低精度項目を削除した場合と削除しない場合でほとんど変化していなかった．また，グランドトゥルースから低精度項目を削除した場合，ターゲットタグが付与されていない投稿での紐づけの精度とターゲットタグが付与された投稿での紐づけの精度が同程度であった．以上の結果を考慮すると，提案手法での紐づけはターゲットタグが付与されていない投稿に対してもターゲットタグが付与された投稿と同程度の性能で適用可能であるといえる．

表 8: NO_TAG 評価データと TAG 評価データでの評価結果

評価データ	$F1\text{-}score@k$			$Precision@k$			$Recall@k$		
	$F@1$	$F@2$	$F@3$	$P@1$	$P@2$	$P@3$	$R@1$	$R@2$	$R@3$
NO_TAG	0.731	0.429	0.303	0.731	0.496	0.399	0.731	0.630	0.645
TAG	0.902	0.550	0.386	0.902	0.670	0.535	0.902	0.911	0.942

表 9: NO_TAG 評価データと TAG 評価データでの評価結果 (低精度項目をグランドトゥルースから削除)

評価データ	$F1\text{-}score@k$			$Precision@k$			$Recall@k$		
	$F@1$	$F@2$	$F@3$	$P@1$	$P@2$	$P@3$	$R@1$	$R@2$	$R@3$
NO_TAG	0.924	0.542	0.380	0.924	0.657	0.523	0.924	0.886	0.911
TAG	0.904	0.551	0.387	0.904	0.672	0.536	0.904	0.914	0.944

6　手法の限界と妥当性への脅威

6.1　手法の限界

提案手法のデータセットの作成において，ターゲットタグを選定する手順 (STEP1-1) がある．本実験では，STEP1-1 における以下の 2 つの部分を自動化できていない．

- 「動詞に対応する名詞 (例: define であれば definition) を生成する」ことである．本実験においては，動詞に対応する名詞を生成することができる自然言語処理ツールやライブラリを見つけることができなかったため動詞から名詞の生成は手動で行った．動詞の語幹に特定の接尾辞 (例: tion, sion, ion, ment) を付加する方法も考えられたが，単純に接尾辞だけを付与するのでは不十分であったため断念した．例えば，“transform” から “transformation” への変換には接尾辞 “tion” のみを付与するのでは不十分であり，“ation” を付与する必要がある．

- 「複数のタグの候補がある場合に優先されたタグが付いた投稿と，優先されたタグと優先されなかったタグの両方が付いた投稿の内容を数件目視で確認し，言語ドキュメントの項目に対応するタグとしてどちらが使われているかを確認して適切なほうを選出する」ことである．例えば，言語ドキュメントの項目 “5.1.4. Nested List Comprehensions” であれば，タグの候補として “nested-lists” と “list-comprehension” が生成され，優先されるタグは “nested-lists” である．しかし，“nested-lists” タグが付いた投稿と，“list-comprehension” タグと “nested-lists” タグが付いた投稿を目視で確認して，この項目には “list-comprehension” タグと “nested-lists” タグの両方をつけるほうが適切であると判断した．本実験においては，このような判断を機械的に行う手法を実現できなかったため目視で確認した．

また，本実験ではターゲットタグの選定 (STEP1-1) において，ターゲットタグとして適切なタグを選定することができなかった項目が存在した．この原因は以下の 2 つが挙げられる．

- 提案手法は Stack Overflow のタグに依存しているため，言語ドキュメントの項目に対応するタグが Stack Overflow に存在しない (“4.5. pass Statements” や “4.8.3. Special parameters”，“4.8.4. Arbitrary Argument Lists”) 場合にはターゲットタグとして適切なタグを選定することができなかった．これは，本提案手法が Stack Overflow のタグに依存していることによる限界であり，この問題を解決するにはタグに依存しない手法を考案する必要がある．

- 提案手法では単語の組み合わせによりタグの候補を生成しているため，言語ドキュメントの項目 “4.8.7. Documentation Strings” から “docstring” タグのような単語が省

略されている形式のタグを選定することができなかった．これは，本提案手法が単語の組み合わせによりタグの候補を生成しているためであり，この問題を解決するには “documentation string” と “docstring” が同じ意味で使用されることを判断できる機構を考案する必要がある．

6.2 外部妥当性への脅威

本研究では，プログラミング言語 Python を対象として提案手法を適用したが，その他の言語については適用していない．そのため，提案手法が他のプログラミング言語にも適用可能であることを十分に主張できない．

本実験では事前学習済みモデルとして CodeBERT を用いたが，CodeBERT が学習しているプログラミング言語 (Python, Java, JavaScript, PHP, Ruby, Go) 以外のプログラミング言語に適用した場合には精度が大きく下がることが予想される．そのため，それらの言語に対して適用する場合は CodeBERT 以外の事前学習済みモデルを用いる必要がある．

6.3 内部妥当性への脅威

事前学習済みモデルを用いた Stack Overflow の投稿から言語ドキュメントの項目の推定には He ら [10] のタグ推薦フレームワークを応用した．このフレームワークを正しく実装するために，He らによってリリースされているパッケージ[6]を再利用した．

提案手法におけるデータセットの作成では，Stack Overflow のタグをもとに各言語ドキュメントの項目と関連している投稿を取得しており，Stack Overflow のタグが正しくつけられていることを前提としている．そのため，誤ってタグ付けされている場合には，Stack Overflow の投稿と言語ドキュメントの項目が対応付けられたデータとして不適切なデータが含まれる可能性がある．

5.3 節のターゲットタグが付与されていない投稿での評価で用いた評価データの数は 134 であり，ターゲットタグが付与された投稿における評価データの数 21140 と比較すると小さい．そのため，ターゲットタグが付与されていない投稿の評価データを増やした場合にその結果が大きく変化する可能性がある．また，ターゲットタグが付与されていない投稿の評価データにおける投稿と投稿に関連する言語ドキュメントの項目の対応付けは著者 1 人で行ったため，評価データの信頼性が十分ではない．

[6] https://github.com/soarsmu/PTM4Tag

7　おわりに

本研究では事前学習済みモデルを用いた Stack Overflow と言語ドキュメントの紐づけ手法を提案した．提案手法の適用可能性を調べるために，提案手法を事前学習済みモデルに CodeBERT を用い，プログラミング言語 Python を対象として適用実験を実施し，Stack Overflow の投稿と Python Tutorial の項目との紐づけの精度を調査した．その結果，ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 1 つである投稿での評価では $F1$-$score$@1 で 0.897 を達成した．ターゲットタグが付与された投稿のうち，投稿に関連している言語ドキュメントの項目が 2 つの場合と 3 つの場合の投稿での評価では $F1$-$score$@1 はそれぞれ 0.917，0.951 を達成し，関連している項目の数が 2 つの時の $F1$-$score$@2 は 0.766，関連している項目の数が 3 つの時は $F1$-$score$@3 は 0.731 を達成した．また，ターゲットタグが付与されていない投稿での評価を行い，低精度項目をグランドトゥルースから削除した場合にはターゲットタグが付与されていない投稿での紐づけの精度とターゲットタグが付与された投稿での紐づけの精度が同程度であった．適用実験を通じて，提案した Stack Overflow と言語ドキュメントの紐づけ手法がプログラミング言語 Python を対象とした場合では有効であることを確認した．

今後の課題としては，提案手法が他のプログラミング言語にも適用可能であるかを調査するために，Python 以外のプログラミング言語を対象として提案手法を適用することが考えられる．また，ターゲットタグが付与されていない投稿に対する紐づけの有効性をより詳細に調査するために，ターゲットタグが付与されていない投稿の評価データを複数人で議論して作成することでより信頼性の高い評価データを作成し，そのデータで評価することが考えられる．さらに，紐づけの精度を向上するために，ハイパーパラメータのチューニングや，本研究の実験で用いた 3 つの要素 (タイトル，質問のテキスト，質問のコード) 以外の Stack Overflow の投稿の要素を用いること，トランケーションやプーリング方法を変更することが考えられる．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 肥後 芳樹 教授には，研究室での発表の機会において鋭い意見を数多く賜りました．研究に関するご指摘は研究の意義を見つめ直す際の指針となり，研究の質を向上させる手助けとなりました．肥後 教授に心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授には，研究室での発表の機会において数多くの御意見・御助言を賜りました．松下 准教授に心より深く感謝いたします．

福知山公立大学情報学部情報学科 眞鍋 雄貴 講師には，研究テーマの設定から本論文の完成まで，数多くの相談に乗っていただくとともに数多くの指針を示していただきました．また，論文執筆や発表練習など多くの場面で御支援を賜りました．眞鍋 講師に心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 神田 哲也 助教には，学部時代の研究から始まり，これまでの研究生活の数多くの場面で大変お世話になりました．日々の研究の相談に対しても常に丁寧に対応くださり，的確なご助言を数多く賜わりました．神田 哲也 助教の御支援のおかげで本論文の完成に至りました．心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻肥後研究室 事務職員 軽部 瑞穂 氏には，研究室での日々の生活で数多くのご支援を賜りました．軽部 瑞穂 氏の御支援のおかげで研究室での生活を快適に過ごすことができました．心より深く感謝いたします．

最後に，その他様々な御指導及び御助言を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻肥後研究室の皆様に心より深く感謝いたします．

参考文献

[1] All Sites - Stack Exchange. https://stackexchange.com/sites?view=list#traffic, Accessed: 2024-01-13.

[2] Stack Overflow. https://stackoverflow.com/, Accessed: 2024-01-13.

[3] The Python Tutorial. https://docs.python.org/3/tutorial/, Accessed: 2024-01-28.

[4] Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. SOTorrent: reconstructing and analyzing the evolution of stack overflow posts. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pp. 319–330, 2018.

[5] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media Inc, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[7] Pierpaolo Dondio and Suha Shaheen. Is stackoverflow an effective complement to gaining practical knowledge compared to traditional computer science learning? In *Proceedings of the 11th International Conference on Education Technology and Computers*, pp. 132–138. Association for Computing Machinery, 2020.

[8] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1536–1547, 2020.

[9] Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. Using titles vs. full-text as source for automated semantic document annotation. In *Proceedings of the 9th Knowledge Capture Conference*, 2017.

[10] Junda He, Bowen Xu, Zhou Yang, DongGyun Han, Chengran Yang, and David Lo. PTM4Tag: Sharpening tag recommendation of stack overflow posts with pre-trained models. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, pp. 1–11. Association for Computing Machinery, 2022.

[11] Junjie Huang, Duyu Tang, Linjun Shou, Ming Gong, Ke Xu, Daxin Jiang, Ming Zhou, and Nan Duan. 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*, Vol. 1: Long Papers, 2021.

[12] Huan JIN and Qinying LI. Fine-tuning pre-trained codebert for code search in smart contract. *Wuhan University Journal of Natural Sciences*, Vol. 28, No. 3, pp. 237–245, 2023.

[13] David Kember, Amber Ho, and Celina Hong. The importance of establishing relevance in motivating student learning. *Active Learning in Higher Education*, Vol. 9, pp. 249–263, 2008.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[15] Theodora Koulouri, Stanislao Lauria, and Robert D. Macredie. Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education*, Vol. 14, No. 4, 2015.

[16] Can Li, Ling Xu, Meng Yan, and Yan Lei. TagDC: A tag recommendation method for software information sites with a combination of deep learning and collaborative filtering. *Journal of Systems and Software*, Vol. 170, p. 110783, 2020.

[17] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, Beth Simon, and Lynda Thomas. A multi-national study of reading and tracing skills in novice programmers. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pp. 119–150, 2004.

[18] Ehsan Mashhadi and Hadi Hemmati. Applying codebert for automated program repair of java simple bugs. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories*, pp. 505–509, 2021.

[19] Chris Parnin and Christoph Treude. Measuring api documentation on the web. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, pp. 25–30, 2011.

[20] David Robinson. How Do Students Use Stack Overflow? - Stack Overflow Blog. https://stackoverflow.blog/2017/02/15/how-do-students-use-stack-overflow/, Accessed: 2024-01-28.

[21] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, Vol. 26, No. 11, pp. 1022–1036, 1983.

[22] Gaurav Kumar Singh, Varun Kumar, Savita Bhat, and Niranjan Pedanekar. Automatically augmenting learning material with practical questions to increase its relevance. In *2015 IEEE Frontiers in Education Conference*, pp. 1–7, 2015.

[23] StackOverflow. What are tags, and how should I use them? https://stackoverflow.com/help/tagging, Accessed: 2024-01-17.

[24] StackOverflow. Tag Synonyms - Stack Overflow. https://stackoverflow.com/tags/synonyms, Accessed: 2024-01-24.

[25] StackOverflow. Stack Overflow Developer Survey 2022. https://survey.stackoverflow.co/2022, Accessed: 2024-01-28.

[26] Michael Staton. Disaggregating the components of a college degree. *Stretching High. Educ. Dollar*, 2012.

[27] Siddharth Subramanian, Laura Inozemtseva, and Reid Holmes. Live api documentation. In *Proceedings of the 36th International Conference on Software Engineering*, pp. 643–652, 2014.

[28] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pp. 194–206, 2019.

[29] Bowen Xu, Thong Hoang, Abhishek Sharma, Chengran Yang, Xin Xia, and David Lo. Post2Vec: Learning distributed representations of stack overflow posts. *IEEE Transactions on Software Engineering*, Vol. 48, No. 9, pp. 3423–3441, 2022.

[30] Xin Zhou, DongGyun Han, and David Lo. Assessing generalizability of CodeBERT. In *2021 IEEE International Conference on Software Maintenance and Evolution*, pp. 425–436, 2021.

[31] 鬼塚仙太郎, 神田哲也, 眞鍋雄貴, 肥後芳樹. Stack Overflow と言語ドキュメントの紐づけ手法の検討. 信学技報, 第 123 巻, pp. 98–1032, jul 2023.

[32] 経済産業省. IT 人材需給に関する調査. https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf, 2019. Accessed: 2024-01-27.

[33] 総務省. 令和 4 年版 情報通信白書, 2022.

[34] 独立行政法人情報処理推進機構 (IPA) 社会基盤センター. IT 人材白書 2020, 2020.

[35] 文部科学省. 「教育の情報化に関する手引」について. https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/mext_00117.html. Accessed: 2024-01-27.

[36] 文部科学省. 小学校プログラミング教育に関する資料. https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416328.htm. Accessed: 2024-01-27.

[37] 眞鍋雄貴. プログラミング学習における状況に応じた学習要素の提示方法の検討. 信学技報, 第 122 巻, pp. 43–48, jul 2022.