情報処理学会論文誌 Vol.59 No.1 1–13 (Jan. 2018)

Stack Overflowと言語ドキュメント間の相互参照のための紐づけ手法

鬼塚 仙太郎[1] 神田 哲也[2,a)] 眞鍋 雄貴[3,b)] 肥後 芳樹[1]

受付日 2016年3月4日，再受付日 2015年7月16日 / 2015年11月20日，
採録日 2016年8月1日

概要：プログラミングの学習において，必要となる情報を自分自身で調べることは重要であり，そのためにStack Overflowや言語ドキュメントのようなWeb上のリソースが活用されている．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相補的な関係にあると考えられる．この関係を活かすには相互に参照できるようになることが望ましい．しかし，これらはともに自然言語で書かれたドキュメントであり，紐づけは容易ではない．本研究では，Stack Overflowの投稿とその投稿に関連する言語ドキュメントを相互に紐づけるための，事前学習モデルを用いた手法を提案する．具体的には，Stack Overflowの投稿と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を作成し，事前学習済みモデルをファインチューニングする．このモデルを用いて，Stack Overflowの投稿に関連する言語ドキュメントの項目を推定することで，紐づけが行われ相互参照が可能になる．提案手法の適用可能性を調べるため，事前学習済みモデルにCodeBERTを用い，プログラミング言語Pythonを対象とした場合の紐づけの精度を調査した．その結果，事前学習モデルを用いて一部の事項について高い精度で紐づけを実現できることを確認した．

キーワード：Stack Overflow，言語ドキュメント，プログラミング学習，事前学習済みモデル

Linking Method for Cross-Referencing between Stack Overflow and Language Documentation

Sentaro Onizuka[1] Tetsuya Kanda[2,a)] Yuki Manabe[3,b)] Yoshiki Higo[1]

Received: March 4, 2016, Revised: July 16, 2015/November 20, 2015,
Accepted: August 1, 2016

Abstract: When learning programming, it is important to find necessary information on your own, and web resources such as Stack Overflow and language documentatio are helpful for this. These two resources have different characteristics but can complement each other, so it would be helpful if they could be linked together. However, since both are written in natural language, linking them is not an trivial task. This study proposes a method using a pre-trained model to link Stack Overflow posts with related language documentation. We created a dataset that matches Stack Overflow posts with corresponding language documentation, and then fine-tuned a pre-trained model using this dataset. By using this model, we can predict related language documentation items from Stack Overflow posts, making linking and cross-referencing possible. To test our method, we used CodeBERT as the pre-trained model and checked the linking accuracy for the Python programming language. The results showed that the pre-training model can be used to achieve high accuracy for some documentation linking.

Keywords: Stack Overflow, language documentations, programming learning, pre-trained models

[1] 大阪大学
Osaka University
[2] ノートルダム清心女子大学
Notre Dame Seishin University
[3] 福知山公立大学
University of Fukuchiyama
[a)] kanda@m.ndsu.ac.jp

1. はじめに

プログラミングの学習方法は多岐にわたるが [1]，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必要がある．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るからである．日々更新される膨大な技術情報を全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，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 [1], [3], [4] と同時に，有益なリソースである [5], [6]．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．

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，プログラミング言語の公式組織が提供する，プログラミング言語の基本機能が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べる．しかし，言語ドキュメントだけでは問題解決能力を習得できないことが示唆されている [7].

そこで，本研究では Stack Overflow と言語ドキュメント間を相互に参照できるよう紐づけする手法を提案する．提案手法では，まずデータセット構築のため，Stack Overflow の投稿に付与されるタグと言語ドキュメントの項目を自然言語処理を用いて機械的に紐づけし，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を得る．このデータセットを用いて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する言語ドキュメントの項目を推定することで目的の紐づけを実現する．また，適用実験を行い，事前学習モデルを用いて一部の事項について高い精度で紐づけを実現できることを確認した．

以降, 2 章では本研究の背景について説明し, 3 章で Stack Overflow と言語ドキュメント間を相互参照できるようにする意義と課題を整理する．4 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間の紐づけ手法について説明する．5 章では提案手法の適用実験の設定について説明し，6 章では実験結果と考察について述べる．7 章で妥当性への脅威について述べ，最後に 8 章でまとめと今後の課題を述べる．

図 1　Stack Overflow における投稿の例 (ID: 30770897)*1

Fig. 1　A post of Stack Overflow (ID: 30770897)

2. 背景

2.1 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 [8]．Stack Overflow における投稿の例を**図 1** に示す．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，また，各質問及び回答にはコメントを付与できる．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 [9]．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．これらのタグには，そのタグの説明が記載された “tag wiki” が設定されている場合がある．

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．2024 年の調査 [10] 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 80.3%であり，これは技

b)　manabe-yuki@fukuchiyama.ac.jp

*1　https://stackoverflow.com/questions/30770897/

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．また，「AI を利用する (37%)」の 2 倍以上であり，生成 AI の利用が広まっている現在においても重要なリソースとされている．Parnin ら [3] は，プログラミングに関する検索クエリにおいて，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としている．Robinson[4] 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[11] 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 [5] は, Stack Overflow の教育目的での利用の有効性を評価し，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 [6] 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．

2.2 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．また，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．これらに加え，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るため，プログラミング言語の基本機能を学ぶのに適したリソースであるといえる．

3. Stack Overflow と言語ドキュメント間の相互参照の意義と課題

本研究では，Stack Overflow と言語ドキュメントを紐づけ相互参照を可能にする手法を提案する．対象とする Stack Overflow の質問は，質問内に説明文のほか該当のプログラミング言語で書かれたコードを含む，もしくは実装に関する議論が行われている質問である．コードや実装に関する議論は，その内容が API などを中心としていても，程度の差はあれプログラミング言語の基本機能に関係しているといえるため，提案手法の理解支援の対象となる．言語ドキュメントは学習向けリソースとして公式に提供されており，かつ網羅性があるため，2.2 節でも示したとおり，紐づけ手法を考える際に適したリソースであるといえる．

以下本章では，本研究が想定するシナリオと手法が目指す支援について例示する．その後，Stack Overflow と言語ドキュメントそれぞれの利用時の問題点と，相互参照によってその問題点がどう解決されるかを述べる．また，相互参照の実現に向けた技術的課題を整理する．

3.1 本研究における想定シナリオ

プログラミング経験の少ない初心者が作成したいソフトウェアがあるときに，作成したいソフトウェアに関するキーワードを Web 検索するような場合を考える．検索結果の中には Stack Overflow の質問がある可能性が高い．その質問には，作りたいソフトウェアを作るうえで有益な情報が含まれているかもしれない．

例題として，図 1 でも示した質問 “How to shoot infinite bullets in game coded in Python?” を用いる．これは “How to shoot infinite bullets” と Google で検索したとき発見される質問である．この質問では，質問者によるプログラムと説明文が掲載されている．そこでは，pygame というモジュールが使われていること，ループの中であることであることが示されている．一方，回答の文章では，スプライトが使われていないことが指摘され，そのために Bullet クラスを作り，update メソッドを定義することで毎回座標を更新するように指示している．また，プログラム中では，Bullet クラスの実装例を示している．この投稿を理解するために必要な知識は様々であるが，本研究では特に，ループやクラスといった言語の基本機能の概念への理解が不足している場合を対象にする．これらの事項は，投稿の中心事項ではないため，タグには出現していない．そのため，追加の理解支援技術が必要になる．

ここでは回答の内容を理解しなければならない例を挙げたが，利用者がまず行うのは質問の内容を理解して自分の作りたいものと関係しているかを判断することである．そのため，特に質問内容に関する読解が，言語の基本機能への理解不足により進まなくなることも問題である．

提案手法は，例題のケースにおいて，回答に対するループやクラスという概念に対して，それを説明するドキュメントへの参照を投稿のタイトルや質問中の文章およびソースコードを用いて作成することを目指すものである．この手法により，初学者に対し投稿の理解に必要なリソースを推薦でき，Stack Overflow の投稿の利活用を進めることが期待できる．

3.2 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 の投稿内容に含まれる

[*2] https://docs.python.org/3/tutorial/
[*3] https://learn.microsoft.com/ja-jp/cpp/cpp/
[*4] https://www.php.net/manual/ja/

要素全てに対して説明があるわけではない．そのため，投稿で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，内容が理解できない可能性があり，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．

Stack Overflow の投稿内容の理解を阻害する要因は複数考えられる．一つは，投稿の中で使用されている API やプログラミング言語の基本機能である．初心者が体系的にプログラミング言語の基本機能を理解しているとは言い難い．もう一つは，質問が扱うドメインの知識である．そのドメイン固有の知識が質問の背景知識となっている場合，様々なソフトウェアを作成した経験のない初心者にとっては理解を阻害する要因となる可能性がある．あるいは，Stack Overflow での質問や回答の書き方が，初心者とって不慣れであり理解しづらい部分があるかもしれない．

ここで，前述した理解を阻害する要因として API やプログラミング言語の基本機能に着目する．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 [12]．一方，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我々の知る限りない．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．そのため，プログラミング言語の基本機能によって Stack Overflow の投稿の理解が阻まれている場合は，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く，解決が困難になることが予想される．

3.3 言語ドキュメント利用時の問題点

言語ドキュメントだけでは，言語の基本機能の適用先を見つけることはできない．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 [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る．実装するためには小問題に対して，どの基本機能が適用できるかを考えなければならない．Kember ら [7] 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きる別のリソースを併用して学習する必要がある．

3.4 Stack Overflow と言語ドキュメント間の相互参照

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，3.2 節で述べた Stack Overflow 利用時の問題と，3.3 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間を紐づけて相互参照可能にする．

Stack Overflow から言語ドキュメントへ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．

言語ドキュメントから Stack Overflow へ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 [5]，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 [6]，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．

3.5 紐づけの困難さ

眞鍋 [14] 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た．その結果, if 文の場合は共通して現れる紐づけに適した名詞がなかったことが示されている．また，我々は以前，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 [15]．プログラミング言語の基本機能のうち，Python における iteration, set，exception の 3 つの単語について，タグはそれが付与された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頻出単語であった場合にはタグとして付与されないことがわかった．これらのことから，Stack Overflow と言語ドキュメント間の紐づけは，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とみられる．

3.6 事前学習済みモデルを用いた紐づけに関する調査

前節の結果をうけて，単純なキーワードマッチングではなく，機械学習を用いた紐づけ手法を検討する．ここでは，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事前学習済みモデル CodeBERT[16], [17], [18] を用いて Stack Overflow の投稿の質問と言語ドキュメントの項目の類似度を調査した．類似度は，Stack Overflow の投稿の質問と言語ドキュメント項目の本文をそれぞれ CodeBERT により分散表現に変換して，2 つの分散表現間のコサイン類似度を用いる．Stack Overflow の投稿は 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のうち，各項目と関連した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 4 件ずつ抽出した．また，言語ドキュメント項目は

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（コサイン類似度は小数第三位で四捨五入）.
Table 1 Cosine similarity between Stack Overflow posts and language documentation items.

言語ドキュメントの項目	質問の ID	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	843277	0.88	0.82	0.76
	1483429	0.82	0.76	0.74
	6470428	0.92	0.89	0.83
	16511337	0.89	0.80	0.79

Python Tutorial[19] の 3 つの項目（“4.2. for Statements”, “5.4. Sets”, “8.3. Handling Exceptions”）とした.

調査結果を表 1 に示す．質問の ID はそれぞれが属するはずの言語ドキュメントの項目の行に配置する．しかし，どの質問においても “4.2. for Statements” 項目との類似度が最も高い．この原因の 1 つとして，CodeBERT のファインチューニングが行われていないことがある．特定のタスクで CodeBERT を使用するには，そのタスク用のデータセットでファインチューニングする必要がある [20], [21] ことから，Stack Overflow と言語ドキュメント間の紐づけに適したデータセットが必要となる．

4. 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間の紐づけ手法を提案する．提案手法の概要図を図 2 に示す．提案手法は，Stack Overflow の投稿について，その質問とタイトルから関連する言語ドキュメントの項目を推薦する．以降，「Stack Overflow の投稿の質問とタイトル」をまとめて単に「質問」と表記する．

投稿を構成する要素のうち回答およびコメントは本手法の対象外とした．これらは質問のコンテキストを受け継いで記述されるものであるため，まず質問に対する正確な紐づけ手法を確立することを本研究の目的とした．

提案手法ではまず，質問とその質問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を作成する（STEP1）．作成したデータで事前学習済みモデルをファインチューニングし，そのモデルを用いて質問からその項目に関連する言語ドキュメントの項目を推定する（STEP2）．以降，手法のキーアイデアについて説明したのち，各 STEP について詳細を述べる．

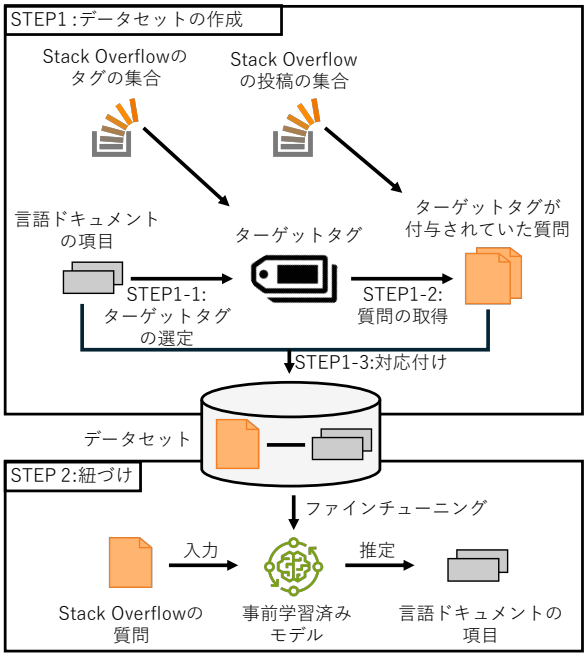


図 2 提案手法の概要図
Fig. 2 An overview of the proposed method.

4.1 キーアイデア

事前学習モデルのファインチューニングのための，質問と関連する言語ドキュメントの項目の対応には，質問に付与されているタグを用いる．先行研究 [15] の考察において，タグが本文の頻出単語であった割合は最大 65.95%であった．一方で，本文の頻出単語がタグとして付与されないケースでは，ライブラリに関するタグが付与されている場合も多く見られた．このことから，現在付与されているタグ自体はある程度正しく，他方正しいタグが欠けている可能性がある．この性質をふまえて，正しいタグが付与されている投稿で紐づけを学習し，タグが不十分な投稿に対処できるモデルを作成することを考えた．

本手法の技術的な新規性としては，以下の 2 点が挙げられる．1) 訓練データ構築に使用する言語ドキュメントとタグの紐づけを，自然言語処理を用いて機械的に行うようにした．2) 既存研究のタグ推薦フレームワークを応用し，事前学習モデルのファインチューニングによって言語ドキュメントの項目を推薦できるようにした

4.2 STEP1: データセットの作成

質問とその質問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 のタグの集合からターゲットタグを選定する．

まず，言語ドキュメントのタイトルから単語列を生成する．これは，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ため (例: sql-server)，それとの対応をとるためである．タイトルを各単語に分割，レンマ化し，ストップワードを除去したリストを作成する．その後，リスト中の各単語から 1~3 単語を組み合わせた単語列を生成する．この際, 単語の前後関係はもとのタイトル中の出現順によらない．また, 動詞であればその動詞に対応する名詞で置き換えた場合についても考慮する．例として，得られた単語が “define” と “function” であった場合，生成される単語列は “define-function”，“definition-function”，“function-define”，“function-definition” である．

次に，Stack Overflow のタグの集合から，1) 出現頻度が低いタグ（今回は 50 以下に設定），2) タグに対する説明が記載できる tag wiki が設定されていないタグ，3) 対象とするプログラミング言語のタグと一緒に質問に付与されたことないタグを除去する．除去されずに残ったタグに対してはレンマ化を行う．

最後に，Stack Overflow におけるタグの同義語 (tag synonyms[22]) を考慮したうえで，単語列の集合と Stack Overflow のタグ集合の間で一致する語句をターゲットタグの候補として選出する．ここで，選出したタグの候補が複数ある場合に，以下の基準に従って候補を選択する．

- 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．重要度の指標として，項目の本文に対する tf-idf[23] を用いる．ただし，tf には対数で計算した $1 + log(tf)$ を用いる．これは，ドキュメントの意味的アノテーションタスクにおいて本文を考慮せずタイトルだけを用いても有効である [24] ように，言語ドキュメントにおいても，タイトルに出現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ためである．なお，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．

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質問を取得する．

STEP1-3: Stack Overflow の質問と言語ドキュメントの項目の対応付け

STEP1-3 では，STEP1-2 で取得した各質問と，その質問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，質問とその質問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．

4.3 STEP2: Stack Overflow の質問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質問と言語ドキュメントの項目の紐づけは，質問と言語ドキュメントの各項目との関連度を計算し，投稿と最も高い関連度を持つ項目の間に紐づけることによっておこなう．提案手法では，He ら [25] 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ベースとし，入力が質問を「タイトル・質問内容の本文・質問内容のコード」に分割したもの，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した．このために，1) 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，2) 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 で作成した質問とその質問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，それぞれ変更を行った．この結果，質問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，その中で関連度の高い項目を入力した投稿と紐づけることが可能になる．

5. 実験設定

提案手法に基づく Stack Overflow と言語ドキュメント間の紐づけの精度を評価するために適用実験を実施した．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．以降では実験項目，対象としたプログラミング言語と言語ドキュメント，それらを用いた提案手法の実行手順，および実験のための評価データと評価指標について説明する．

5.1 実験項目

最初に，本手法による推薦の可能性を評価するため，ターゲットタグが付与されている質問に対して手法を適用する．タグが付与されているということは，質問内容が基本機能に関連していると考えられる．つまり，推薦の難易度は 3 章で例に挙げたような，基本機能が話題の中心ではない場合に比べて低いため，ここで得られる精度が，今回ファインチューニングするモデルで実現可能な精度の上限とみなすことができる．実験では，関連する言語ドキュメントの項目が 1 つである質問を用いて，項目を正しく見つけることができるかを調査する．また，関連する言語ドキュメントの項目が複数ある質問を対象にして，項目を網羅できるかをそれぞれ評価する．その後，3 章の例題で示したような実際の利用に近い状態，つまりターゲットタグが付与さ

表 2 Python Tutorial の各項目に対するターゲットタグの選定結果および収集したデータのサイズ.

Table 2 Results and size of collected data for each item in the Python Tutorial.

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	タグ付き投稿数	(単一)	NO_TAG
4.1. if Statements	if-statement	○	6,377	(4,729)	14
4.2. for Statements	for-loop	○	11,662	(9,631)	28
4.3. The range() Function	range & function	○	57	(46)	15
4.4. break and continue Statements, and else Clauses on Loops	break, continue, if-statement & loops	○	991	(379)	9
4.5. pass Statements	statements	×(−)	-	(-)	-
4.6. match Statements	match	○	-	(-)	-
4.7. Defining Functions	function-definition	○	6	(6)	21
4.8.1. Default Argument Values	default-arguments	○	52	(48)	11
4.8.2. Keyword Arguments	keyword-argument	○	616	(540)	11
4.8.3. Special parameters	parameters	×(−)	-	(-)	-
4.8.4. Arbitrary Argument Lists	arguments	×(−)	-	(-)	-
4.8.5. Unpacking Argument Lists	argument-unpacking	○	84	(61)	9
4.8.6. Lambda Expressions	lambda	○	3,553	(3,158)	11
4.8.7. Documentation Strings	documentation	× (docstring)	-	(-)	-
4.8.8. Function Annotations	function & annotations	○	13	(13)	5
5.1.1. Using Lists as Stacks	stack	○	457	(409)	2
5.1.2. Using Lists as Queues	queue	○	1,442	(1,381)	3
5.1.3. List Comprehensions	list-comprehension	○	4,445	(3,477)	16
5.1.4. Nested List Comprehensions	list-comprehension & nested-lists	○	51	(0)	4
5.2. The del statement	del	○	162	(133)	4
5.3. Tuples and Sequences	tuples, sequence	○	6,251	(5,069)	10
5.4. Sets	set	○	2,465	(1,974)	9
5.5. Dictionaries	dictionary	○	34,580	(31,866)	10
5.7. More on Conditions	conditional-statements	○	1,225	(928)	5

れていない質問に対しても推薦が可能であるかを評価する．よって，実験項目を以下の 3 点とした．

実験 1: タグ付与あり/単一項目 ターゲットタグが付与された質問のうち，質問に関連する言語ドキュメントの項目が 1 つである質問での評価

実験 2: タグ付与あり/複数項目 ターゲットタグが付与された質問のうち，質問に関連する言語ドキュメントの項目が複数ある質問での評価

実験 3: タグ付与なし ターゲットタグが付与されていない質問での評価

5.2 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 Python を選択した．Python の言語ドキュメントは Python Tutorial[19] が該当する．このうち，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．

5.3 STEP1: データセットの作成

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** の 1 列目から 3 列目に示す．「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を取得する．「著者の評価」列は，選定されたタグが適切であるかを，第一著者が自身の経験とタグの tag wiki をもとに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，適切でない場合は「×」に加え，より適切なタグが見つかった場合はそのタグを，見つからなかった場合はハイフンを括弧内に記載した．

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 19 項目を対象として，そのタグを持つ質問を取得した．除外した項目 “4.6. match Statements” は，後述する本実験で用いるデータセットのリリース時には Python の機能として含まれていないものであった．

STEP1-3 では，STEP1-2 で取得した質問と，その質問

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質問数を，表 2 の 4 列目から 5 列目に示す．「(単一)」列は，複数の言語ドキュメントの項目と関連する質問を除いて，単一の言語ドキュメントの項目のみに関連する質問の数を示す．この単一の言語ドキュメントの項目のみに関連する質問を，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する質問については，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 つの項目と関連する質問の数が 4934，3 つと関連する質問のが 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[26] 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．これは Stack Overflow のタグ推薦の既存研究 [25], [27], [28] 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[29](Version 2020-12-31) を利用した．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．

5.4 STEP2 の準備: ファインチューニング

質問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，CodeBERT[21] を用いた．これは，タグ推薦フレームワークにおいて比較された事前学習済みモデルの中で CodeBERT が最高性能を達成しているからである [25]．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 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．CodeBERT が処理できる最大トークン数は 512 であるため，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 [30] 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．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．

5.5 評価データ

評価データとして，質問から 3 種類の実験にそれぞれ対応する評価データを構築する．

- 1 種類目: ターゲットタグが付与された質問のうち，投稿に関連する言語ドキュメントの項目が 1 つである質問 (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質問のうち，投稿に関連する言語ドキュメントの項目が複数ある質問 (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質問 (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質問であり，5.3 節で説明した訓練データに使用しなかった質問である．評価データに含まれる質問の数は，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 つの項目と関連する質問の数が 4886，3 つと関連する質問の数が 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質問に対する紐づけの精度を評価するためのものである．5.3 節で収集したデータとは別に，ターゲットタグが付与されていない質問を収集し，質問と質問に関連する言語ドキュメントの項目が対応づいた NO_TAG 評価データを作成した．具体的には，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質問の中から，ターゲットタグが付与されていない質問を抽出して内容を目視で確認し，その質問と関連する言語ドキュメントの項目を対応付けて，134 件のデータを得た．NO_TAG 評価データに含まれる言語ドキュメントの項目ごとの質問の数を，表 2 の 6 列目に示す．1 つの質問が複数の項目と関連する場合もあるため，合計は 134 にはならない．NO_TAG 評価データのうち，質問に関連する言語ドキュメントの項目が 1 つである質問の数は 81，2 つの項目と関連する質問の数が 45，3 つと関連する質問の数が 6，4 つと関連する質問の数が 2 である．

5.6 紐づけの精度の評価指標

事前学習済みモデルを用いて質問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，タグ推薦 [25], [27], [28] において一般的に使用されている指標である $Precision@k$，$Recall@k$，$F1\text{-}score@k$ の 3 つを用いた．これらは質問 $x_i(1 \leq i \leq N, N$ は評価データの総数) が与えられたときの，推薦された項目の上位 k 件に対する各指標 $Precision@k_i$，$Recall@k_i$，$F1\text{-}score@k_i$ の平均であり，以下のように計算される．

$Precision@k_i$ は，モデルが推定した関連度の高い上位 k 個の項目の中に正解の項目が含まれていた割合である．質問 x_i に対する正解項目の集合を GT_i，質問 x_i からモデルが推定した関連度の高い項目上位 k 個の集合を $Item_i^k$ とすると $Precision@k_i$ は式 1 で計算される．この平均をとったものが $Precision@k$ である．

$$Precision@k_i = \frac{|GT_i \cap Item_i^k|}{k} \tag{1}$$

$Recall@k_i$ は，正解の項目のうちモデルが推定した関連度の高い上位 k 個の項目の中に含まれていた割合である．質問 x_i に対して $Recall@k_i$ は式 2 で計算される．ただし，

表 3 TAG_SINGLE 評価データでの評価結果
Table 3 Evaluation results for TAG_SINGLE dataset.

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

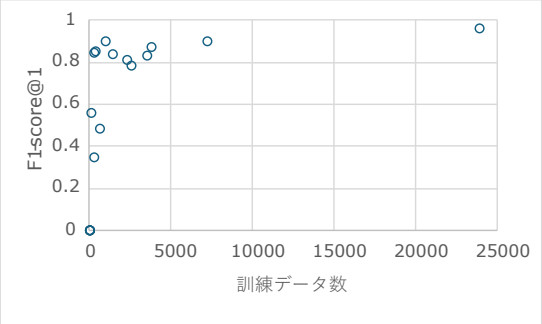


図 3 訓練データの数と $F1$-$score$ の関係
Fig. 3 Relationship between training data and $F1$-$score$.

k の値が正解の項目数未満の場合は，正解すべてを推薦することが不可能なため，k を基準として計算する．この平均をとったものが $Recall@k$ である．

$$Recall@k_i = \begin{cases} \frac{|GT_i \cap Item_i^k|}{k} & (|GT_i| > k) \\ \frac{|GT_i \cap Item_i^k|}{|GT_i|} & (|GT_i| \leq k) \end{cases} \quad (2)$$

$F1$-$score@k_i$ は $Precision@k_i$ と $Recall@k_i$ の調和平均であり，式 3 で計算する．この平均をとったものが $F1$-$score@k$ である．

$$F1\text{-}score@k_i = 2 \times \frac{Precision@k_i \times Recall@k_i}{Precision@k_i + Recall@k_i} \quad (3)$$

6. 実験結果と考察

本章では，5 章で説明した，3 種類の評価データに対して紐づけの精度を評価した実験の結果と，その結果にもとづく考察を述べる．以降，各評価指標の値は小数第 4 位以下を切り捨てたものである．

6.1 実験 1: タグ付与あり/単一項目

6.1.1 結果

各 epoch における $F1$-$score$@1 の結果を**表 3** に示す．$k = 1$ の場合は $Precision$@1，$Recall$@1 及び $F1$-$score$@1 が一致するため $F1$-$score$@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．なお，紐づけに要した時間は約 10 分であった．

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．その結果を**表 4** に示す．ただし，“5.1.4. Nested List Comprehensions” はその項目だけに関連している質問が存在しなかったため結果がない．結果から，“4.2. for Statements”，“5.5. Dictionaries’，“5.1.2. Using Lists as Queues” のように精度が高い項目がある一方，“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” が 0 であるなど，精度が低い項目も存在していることがわかる．

表 4 TAG_SINGLE 評価データでの項目ごとの評価結果
Table 4 Evaluation results for TAG_SINGLE dataset for each item in the Python Tutorial.

言語ドキュメントの項目	$F1$-$score$@1
4.1. if Statements	0.829
4.2. for Statements	0.897
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	0.810
4.8.8. Function Annotations	0
5.1.1. Using Lists as Stacks	0.844
5.1.2. Using Lists as Queues	0.895
5.1.3. List Comprehensions	0.785
5.1.4. Nested List Comprehensions	-
5.2. The del statement	0.558
5.3. Tuples and Sequences	0.873
5.4. Sets	0.836
5.5. Dictionaries	0.962
5.7. More on Conditions	0.482

6.1.2 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，それが精度に影響している可能性がある．そこで，訓練データの数と $F1$-$score$@1 に関係があるのかを調査した．訓練データの数と $F1$-$score$@1 の関係を表した散布図を**図 3** に示す．散布図の各点は，各項目の訓練データの数と $F1$-$score$@1 に対応する．散布図から，訓練データの数が少ない場合には $F1$-$score$@1 が極端に低い場合があることがわかる．特に，訓練データの数が 50 以下になると $F1$-$score$@1 が 0 となっていた．このことから，訓練データの数は少なくとも 100 程度は必要といえる．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質問を組み合わせて新たな質問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．

また，“4.4. break and continue Statements, and else Clauses on Loops” の精度が低い．ターゲットタグのうち if-statement & loops については，この 2 つのタグが付与された質問は “4.1. if Statements” の該当条件も満たす．このため 2 つのドキュメントに対応する質問として扱わ

表 5 TAG_MULTI 評価データでの評価結果
Table 5 Evaluation results for TAG_MULTI dataset.

関連する	$F1\text{-}score@k$				$Precision@k$		$Recall@k$	
項目の数	$F@1$	$F@2$	$F@3$	$F@4$	$P@3$	$P@4$	$R@3$	$R@4$
2	0.918	0.766	0.684	0.609	0.570	0.456	0.855	0.913
3	0.946	0.869	0.729	0.694	0.729	0.607	0.729	0.810

れ，TAG_SINGLE 評価データには含まれない．それ以外のターゲットタグ break や continue が付与された質問が 4.4. に対応する質問として扱われる．4.4. に対する推薦結果は，“4.2. for Statements” 約 49%，4.4.（正解）約 35%，“4.1. if Statements” 約 12%であった．ターゲットタグが包含関係になっていることと，内容が他の章の発展的内容であることから，期待した推薦結果にならなかったと考えられる．言語ドキュメントの項目とタグの対応が必ずしも 1:1 でないことから，ドキュメントの構成によってはこのような状況が起こり得る．ターゲットタグの包含関係は “5.1.4 Nested List Comprehensions” でも発生している．list-comprehension & nested-lists が付与された質問は同時に “5.1.3 List Comprehensions” の該当条件を満たすため，2 つのドキュメントに対応する質問として扱われる．こちらは他にターゲットタグがないため，TAG_SINGLE データセットには出現しない．内容が他の章の発展的内容であり精度が低い現象は “5.7 More on Conditions” でも発生しており，推薦結果は 5.7.（正解）約 48%，“4.2. for Statements” 約 19%，“5.5. Dictionaries” 約 3%であった．

6.2 実験 2: タグ付与あり/複数項目

6.2.1 結果

ファインチューニング済みのモデルには，6.1 節の評価で最高性能を達成した epoch1 のモデルを用いた．紐づけに要した時間は質問に関連する言語ドキュメントの項目が 2 つの質問で約 3 分，3 つの質問で約 10 秒であった．評価結果を**表 5** 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，関連する項目の数が 2 つの時は 0.917，関連する項目の数が 3 つの時は 0.951 を達成した．関連する項目すべてと紐づける精度については，関連する項目の数が 2 つの時の $F1\text{-}score@2$ は 0.766，関連する項目の数が 3 つの時の $F1\text{-}score@3$ は 0.729 を達成した．

6.2.2 考察

F1-score@1 が高精度になる要因として，TAG_SINGLE で高精度を達成した項目がほとんどの質問に含まれていたことが挙げられる．実験 1 において精度低下の原因となっていた，訓練時にデータ数 50 以下の項目が使われているケースは 61 件（全体の約 1%）だった．

一方で関連する項目すべてと紐づける精度はこれよりもやや低い．関連する項目の数が 2 の質問について組み合わせ別に分析を行ったところ，精度が高い (0.7〜0.9) 項目はほとんどが学習時にデータ数の多かった 4.2 もしくは 5.5 を含んでいた．これら以外の項目については，他の項目と複合した際の推薦精度が単体時に比べて悪く (0.5〜0.7) なっている．

また，複数項目との関連については，質問に付与されているタグの品質が良くないケースも見られた．例えば，ループ内で辞書を呼び出す質問[*5]においては，タグは `for-loop` と `lambda` が付与されているが，手法は “4.8.6. Lambda Expressions” と “5.5. Dictionaries” を提案している．この質問に対する回答を見ると，この質問の課題は辞書に対する参照方法の問題であったため，タグ付けされた `for-loop` に関連する “4.2. for Statements” よりも “5.5. Dictionaries” を上位に推薦することが問題の理解と解決に役立つと考えられる．つまり，ターゲットタグが付与されている質問であっても，本手法の推薦結果を併用することでより質問の理解が促進されるケースもあるといえる．

訓練データの数を十分に確保できなかったことから，質問に関連する言語ドキュメントの項目が複数ある質問は訓練データに含めていなかった．それにもかかわらず，質問に関連する言語ドキュメントの項目が 1 つである質問での学習が，複数の項目に関連する質問における紐づけにも有効であるという結果が得られた．一方で，複数の項目に関連する質問のデータセットを拡張して，それを訓練データに含めることで精度向上を目指すことも考えられる．例えば，複数の質問を組み合わせて 1 つの質問として複数の項目に関連していると扱う方法が考えられる．

6.3 実験 3: タグ付与なし

6.3.1 結果

ファインチューニング済みのモデルには，6.1 節の評価で最高性能を達成した epoch1 のモデルを用いた．比較のために，ターゲットタグが付与された質問の評価データとして，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) を用いた．評価結果を**表 6** 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．

[*5] https://stackoverflow.com/questions/51830722/

表 6 NO_TAG 評価データと TAG 評価データでの評価結果
Table 6 Evaluation results for NO_TAG dataset along with TAG dataset.

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	0.496	0.399	0.731	0.429	0.303	0.731	0.630	0.645
TAG	0.902	0.670	0.535	0.902	0.550	0.386	0.902	0.911	0.942

表 7 NO_TAG 評価データと TAG 評価データでの評価結果（TAG_SINGLE での低精度項目を除く）
Table 7 Evaluation results for NO_TAG dataset along with TAG dataset (omitting low-precision items on TAG_SINGLE).

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

ここで，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ことに注目する．特に，6.1 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．つまり，モデルが低精度で紐づけを行う項目に関連する評価データが多いため，ターゲットタグが付与されていない質問での $F1\text{-}score@1$ を引き下げている可能性がある．

そこで，低精度項目のみに関連している質問を除去した各評価データを用いて，紐づけの精度を改めて評価した．除去により 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．**表 7** に示す評価結果では，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，低精度項目の除去前 (表 6) 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．

6.3.2 考察

ターゲットタグが付与された質問に対する紐づけの精度は，低精度項目を除去する前後でほとんど変化していなかった．また，低精度項目を除去した場合，ターゲットタグが付与されていない質問での紐づけの精度とターゲットタグが付与された質問での紐づけの精度が同程度であった．以上の結果から，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質問に対しても，ターゲットタグが付与された質問と同程度の性能で適用可能であるといえる．

6.4 データセット構築のための紐づけに対する考察

実験 1，2 では現在付与されているタグが正しいものとして実験を行った．いくつかの質問を確認したところ，言語ドキュメントとして別の項目が優先して紐づけられるべきパターンはあったものの，今回の紐づけ対象外となるタグが付与されている例は見当たらなかった．Stack Overflow のタグは新規ユーザーは作成できないなど，乱立に制限がかけられている．また，言語ドキュメントの見出しはその内容となる文法事項を正確な用語を用いて示しており，今回のキーワードを用いたターゲットタグ選定手法によってカバーできていないタグが多く残っている可能性は低いと考える．このことから，今回利用したターゲットタグは，対応付けのファインチューニングを行うために十分正しく選択できていたと考えられる．

7. 妥当性への脅威

7.1 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は主張できない．ただし，他の主要なプログラミング言語においても言語ドキュメントは整備されており，Stack overflow においても多くの質問が行われている．このことから，主要なプログラミング言語においては同様の結果が得られると考えている．

7.2 内部妥当性への脅威

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す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，質問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．しかし，質問につけられるタグは他者からレビューされ，修正されることもあるため，タグ付けが誤っている可能性は低いと考えられる．

8. おわりに

本研究では，Stack Overflow と言語ドキュメント間を相互参照可能にするために，事前学習済みモデルを用いた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 の質問と Python Tutorial の項目との間の紐づけの精度を調査した．その結果，事前学習モデルを用いて一部の事項について高い精度で紐づけを実現できることを確認した．

今後の課題として，紐づけが低精度であった項目に対する改善があげられる．訓練データの数が少ない項目に対して，複数の質問を組み合わせて新たな質問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が考えられる．また，提案手法は StackOverflow のタグや単語の組み合わせに依存しており，ターゲットタグが生成できない場合があった．このような場合でもタグが生成できるように手法を拡張することも重要な課題である．

提案手法の応用先として，Stack Overflow の回答やコメントに対する理解支援が考えられる．例題のケースでは回答の中で内容が完結しているため，今回作成したモデルをそのまま適用しやすいと考えられるが，一般的に回答やコメントは質問のコンテキストを受け継いで記述されるものであるため，質問を対象とする場合に比べて推薦精度を保つための工夫が必要である．また，対象とするドキュメントを基本機能から広げることで，参照先が増えてより効率的な理解支援を行うことができる可能性がある．他には，API 利用についても，本文の内容を解析することで関係する項目を探し出すという面で応用できる可能性がある．具体的な API のシグネチャが記述されていれば，それをもとにドキュメントを推薦する手法は存在するが，そうではない場合，例えば抽象的な質問内容から関連するライブラリを推薦するといった用途に応用できる可能性もある．また，言語ドキュメントの中でも今回利用しなかった項目や，言語ドキュメント以外の文書，例えば仕様書やマニュアル，チュートリアルなどへの紐づけを行い，利用可能なリソースを増やすことも考えられる．

謝辞 本研究は，JSPS 科研費 JP21H04877, JP21K02862, JP21K18302, JP22K11985, JP23K24823, JP23K28065, JP24H00692, JP24K14895 の助成を受けたものです．

鬼塚 仙太郎

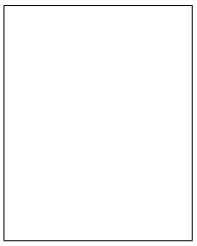
2022 年大阪大学基礎工学部情報科学科卒業．2024 年同大学大学院情報科学研究科コンピュータサイエンス専攻博士前期課程修了．在学中，プログラミング学習の支援に関する研究に従事．

神田 哲也 （正会員）

2011 年大阪大学基礎工学部情報科学科卒業．2016 年同大学大学院情報科学研究科博士後期課程修了．同年奈良先端科学技術大学院大学情報科学研究科博士研究員．2017 年大阪大学大学院情報科学研究科特任助教．2018 年同助教．2024 年ノートルダム清心女子大学情報デザイン学部准教授．博士（情報科学）．ソフトウェア進化，ソースコード解析に関する研究に従事．情報処理学会，IEEE 各会員．

眞鍋 雄貴 （正会員）

2006 年大阪大学基礎工学部情報科学科退学．2011 年同大学大学院情報科学研究科博士課程修了．2011 年同大学院特任助教．2013 年熊本大学自然科学研究科助教．2016 年熊本大学大学院先端科学研究部助教．2020 年福知山公立大学情報学部講師．現在に至る．ソフトウェア工学，特にソフトウェアの再利用，ソフトウェアライセンス，リポジトリマイニングの研究に従事．博士（情報科学）．電子情報通信学会，日本データベース学会，ACM，IEEECS 各会員．



肥後 芳樹 （正会員）

2002 年大阪大学基礎工学部情報科学科中退．2006 年同大学大学院博士後期課程了．2007 年同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助教．2015 年同准教授．2022 年同教授．博士（情報科学）．ソースコード分析，特にコードクローン分析，リファクタリング支援，ソフトウェアリポジトリマイニングおよび自動プログラム修正に関する研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE 各会員．本会シニア会員．

参考文献

[1] Stack Overflow: Stack Overflow Developer Survey 2022, (online), available from ⟨https://survey.stackoverflow.co/2022⟩ (accessed 2024-01-13).
[2] Stack Overflow: Stack Overflow, (online), available from ⟨https://stackoverflow.com/⟩ (accessed 2024-01-13).
[3] Parnin, C. and Treude, C.: Measuring API Documentation on the Web, *Proc. of Web2SE 2011*, pp. 25–30 (2011).
[4] Robinson, D.: How Do Students Use Stack Overflow?, Stack Overflow Blog (online), available from ⟨https://stackoverflow.blog/2017/02/15/how-do-students-use-stack-overflow/⟩ (accessed 2024-01-13).
[5] Dondio, P. and Shaheen, S.: Is StackOverflow an Effective Complement to Gaining Practical Knowledge Compared to Traditional Computer Science Learning?, *Proc. of ICETC 2019*, pp. 132–138 (2020).
[6] Singh, G. K., Kumar, V., Bhat, S. and Pedanekar, N.: Automatically augmenting learning material with practical questions to increase its relevance, *Proc. of FIE 2015*, pp. 1–7 (2015).
[7] Kember, D., Ho, A. and Hong, C.: The importance of establishing relevance in motivating student learning, *Active Learning in Higher Education*, Vol. 9, pp. 249–263 (2008).
[8] Stack Exchange: All Sites, (online), available from ⟨https://stackexchange.com/sites⟩ (accessed 2024-01-13).
[9] Stack Overflow: What are tags, and how should I use them?, (online), available from ⟨https://stackoverflow.com/help/tagging⟩ (accessed 2024-01-17).
[10] Stack Overflow: Stack Overflow Developer Survey 2024, (online), available from ⟨https://survey.stackoverflow.co/2024⟩ (accessed 2025-01-20).
[11] Staton, M.: *Stretching the Higher Education Dollar*, chapter Disaggregating the Components of a College Degree, Harvard Education Press (2012).
[12] Subramanian, S., Inozemtseva, L. and Holmes, R.: Live API Documentation, *Proc. of ICSE 2014*, pp. 643–652 (2014).
[13] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B. and Thomas, L.: A multi-national study of reading and tracing skills in novice programmers, *ACM SIGCSE Bulletin*, Vol. 36, No. 4, p. 119–150 (2004).
[14] 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，Vol. 122, No. 138, pp. 43–48 (2022).
[15] 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，Vol. 123, No. 123, pp. 98–1032 (2023).
[16] Huang, J., Tang, D., Shou, L., Gong, M., Xu, K., Jiang, D., Zhou, M. and Duan, N.: CoSQA: 20,000+ Web Queries for Code Search and Question Answering, *Proc. of ACL-IJCNLP 2021*, pp. 5690–5700 (2021).
[17] Mashhadi, E. and Hemmati, H.: Applying CodeBERT for Automated Program Repair of Java Simple Bugs, *Proc. of MSR 2021*, pp. 505–509 (2021).
[18] Zhou, X., Han, D. and Lo, D.: Assessing Generalizability of CodeBERT, *Proc. of ICSME 2021*, pp. 425–436 (2021).
[19] Python Software Foundation: The Python Tutorial, (online), available from ⟨https://docs.python.org/3/tutorial/⟩ (accessed 2024-01-28).
[20] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *Proc. of NAACL 2019* (2019).
[21] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D. and Zhou, M.: CodeBERT: A Pre-Trained Model for Programming and Natural Languages, *Proc. of EMNLP 2020*, pp. 1536–1547 (2020).
[22] Stack Overflow: Tag Synonyms, (online), available from ⟨https://stackoverflow.com/tags/synonyms⟩ (accessed 2024-01-24).
[23] Salton, G., Fox, E. A. and Wu, H.: Extended Boolean information retrieval, *Communications of the ACM*, Vol. 26, No. 11, pp. 1022–1036 (1983).
[24] Galke, L., Mai, F., Schelten, A., Brunsch, D. and Scherp, A.: Using Titles vs. Full-text as Source for Automated Semantic Document Annotation, *Proc. of K-CAP 2017* (2017).
[25] He, J., Xu, B., Yang, Z., Han, D., Yang, C. and Lo, D.: PTM4Tag: Sharpening Tag Recommendation of Stack Overflow Posts with Pre-Trained Models, *Proc. of ICPC 2022*, pp. 1–11 (2022).
[26] Bird, S., Klein, E. and Loper, E.: *Natural language processing with Python: analyzing text with the natural language toolkit*, O'Reilly Media Inc (2009).
[27] Xu, B., Hoang, T., Sharma, A., Yang, C., Xia, X. and Lo, D.: Post2Vec: Learning Distributed Representations of Stack Overflow Posts, *IEEE Trans. Softw.*, Vol. 48, No. 9, pp. 3423–3441 (2022).
[28] Li, C., Xu, L., Yan, M. and Lei, Y.: TagDC: A tag recommendation method for software information sites with a combination of deep learning and collaborative filtering, *J. Syst. Softw.*, Vol. 170, p. 110783 (2020).
[29] Baltes, S., Dumani, L., Treude, C. and Diehl, S.: SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts, *Proc. of MSR 2018*, pp. 319–330 (2018).
[30] Sun, C., Qiu, X., Xu, Y. and Huang, X.: How to Fine-Tune BERT for Text Classification?, *Chinese Computational Linguistics*, pp. 194–206 (2019).