Vol.2022-SE-210 No.17
2022/3/11

Pythonにおける機械学習関連ライブラリの自動推薦手法の初期評価

小栁 慶[1,a)]　秋山 楽登[1,b)]　山手 響介[1,c)]　近藤 将成[1,d)]　亀井 靖高[1,e)]　鵜林 尚靖[1,f)]

概要：近年，深層学習や機械学習の研究が盛んに行われ，それに伴って，機械学習に関するライブラリを多数活用可能なプログラミング言語 Python の需要が増大している．しかしながら，ライブラリの数は年々増加し続けているため，必要となるライブラリを発見することが，開発者にとって非常に負担のかかる作業となっている．そこで，ライブラリ推薦に関する研究が盛んに行われているが，これまでのほとんどの研究の対象言語は，Java である．本研究では，従来使用されている推薦手法を参考にし，協調フィルタリングを用いて，Python における機械学習関連ライブラリを推薦し，従来の推薦手法が Python においても有効であるか 5 つの評価指標を用いて調査する．調査の結果，評価指標値の推移は対象言語が Java の場合と類似した傾向を示したため，対象言語を Python に変更しても，協調フィルタリングを用いた推薦手法は有効であることが確認できた．推薦されたライブラリは，一般的に使用されるものが多く，今後推薦システムの改善が必要である．

キーワード：協調フィルタリング，機械学習ライブラリ，Python，レコメンドシステム，Collaborative filtering

1. はじめに

近年，ソフトウェアシステム開発において，大量のデータを分析する手段として，機械学習や深層学習が注目を集めている．機械学習とは，コンピュータが自動でデータを学習し，データの背景にあるパターンを発見する手法のことで，交通量や生産量の予測，画像認識や音声認識など日常生活の様々な場面で応用されている．

このように，機械学習の積極的な使用に伴って，機械学習ライブラリの数が豊富なプログラミング言語である Python の需要が増大している．ライブラリとは，汎用性に富んだプログラムを複数集め，それらを再利用可能な状態でまとめたものである．そのため，システム開発を行う際にオープンソースソフトウェア（以下，OSS）のライブラリを探索して，それを開発者自身のプログラムに組み込むことは，ソフトウェア開発の大幅なコスト削減につながる．

しかし，OSS で提供されている再利用可能なライブラリは年々増加しており，その種類も様々である．ゆえに，開発者にとって必要なライブラリを発見することが負担のかかる作業となっている．この問題を解決するために開発者にライブラリを推薦する研究が盛んに行われているが，現在提案されている手法のほとんどの対象言語は Java である [1] [2] [3] [4] [5] [6][7]．そこで，Java に対して提案されている推薦手法を Python に応用することを考える．得られた結果を対象言語が Java である場合と比較することで，Python においても推薦手法が有効であるか調査し，機械学習システム開発者に対して，開発コストの削減に寄与できることを目指す．

本研究では対象の言語を Python，推薦対象を機械学習ライブラリとして，プログラム中で開発者が使用しているライブラリ情報をもとに，開発途上で開発者が使用する可能性の高い機械学習関連ライブラリを予測，推薦し，5 つの評価指標 $SuccessRate@N$, $Precision@N$, $Recall@N$, $Coverage@N$, $Entropy@N$ を使用して結果を分析する．調査の結果，5 つの評価指標値の推移は Java の場合と類似傾向を示したため，Java における推

[1] 九州大学
Kyushu University
[a)] koyanagi@posl.ait.kyushu-u.ac.jp
[b)] akiyama@posl.ait.kyushu-u.ac.jp
[c)] yamate@posl.ait.kyushu-u.ac.jp
[d)] kamei@ait.kyushu-u.ac.jp
[e)] kamei@ait.kyushu-u.ac.jp
[f)] ubayashi@ait.kyushu-u.ac.jp

薦手法は Python にも適用可能であると考えられる．

以降，第 2 章で関連研究および本研究の目的について紹介し，第 3 章では，本研究の実験設計について説明する．第 4 章で調査結果を示し，第 5 章で，調査結果の考察を行い，第 6 章で妥当性への脅威について述べる．最後に第 7 章で本論文の結論と今後の課題について述べる．

2. 背景と目的

2.1 本研究の目的

従来の手法を Python に応用することができれば，機械学習のシステム開発をより効率的に進めることができると考えられる．本研究では推薦手法として，Nguyen ら [4] の推薦システムを参考にする．そして，プロジェクト間の類似度を算出し協調フィルタリングを用いて，推薦対象を Python の機械学習関連ライブラリとして初期調査を行う．調査内容は以下の通りである．

調査課題： 本研究では，Python において機械学習関連ライブラリを推薦できるのか（RQ1），データ数を変更した際，推薦結果に変化が生じるのか（RQ2），およびライブラリの種類によって推薦頻度に差が生じるのか（RQ3）について調査を行う．

2.2 協調フィルタリング

協調フィルタリングとは，あるユーザーに対して，そのユーザーが興味を持つ可能性の高いアイテムを推薦するシステムを実現する手法の一つである．具体的なシステムの概要を以下に示す．まず，推薦対象のユーザー X（以下，X）は，用意されているアイテムに対して，最低一つ以上評価を行う．次に，膨大なデータセットの中から，X と様々な観点において類似しているユーザー Y（以下，Y）を探索する．そして，X が未評価のアイテムに対して，Y の行動データをもとに X の評価値を予測する．最後に，予測した評価値から X が興味を持つ可能性の高いアイテムを X に対して推薦する．

本研究では，ユーザーをプロジェクト，アイテムをライブラリに見立て，ユーザーが各アイテムに対して行う評価を，プロジェクトが各ライブラリを使用しているかどうかとし，調査を行う．ここで Python におけるライブラリとは，ある目的のために機能をまとめたパッケージのことを指し，パッケージとは複数の関数を集めたモジュールをさらに複数集めたものである．プロジェクトとライブラリの関係を図 1 に示す．

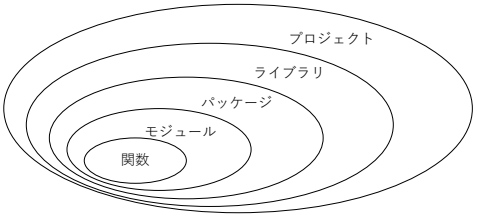


図 1 プロジェクト・ライブラリ関係図

2.3 関連研究

ソフトウェアシステム開発において，開発者を支援するための推薦システムは，Java クラスやライブラリ，ソフトウェアコンポーネントを対象として，様々な研究が行われており，本節では，その一部を紹介する．

Tsunoda ら [1] は，システムが使用している Java クラスを入力として，協調フィルタリングを用いて Java クラスファイルを推薦する手法 Javawock を提案し，性能を評価した．その結果として，協調フィルタリングを用いて類似プロジェクトを発見し，そこから使用されるべきクラスを推薦した場合の方が，単にデータセットからクラスの使用頻度（平均）を算出して，クラスを推薦した場合よりも高精度な予測ができたことを示している．

Ouni ら [2] は，多目的最適化アルゴリズム NSGA-Ⅱ を用いてライブラリを探索し，推薦する LibFinder という手法を提案し，従来のライブラリ推薦システムとして用いられている手法 LibRec[3] との比較を行っている．その結果として，すべての評価指標で LibFinder の方が優れた結果が得られたことを示している．

さらに，Nguyen ら [4] は，プロジェクトが使用しているライブラリ情報を入力とし，協調フィルタリングを用いてライブラリを推薦する CrossRec という手法を提案している．この手法では，各プロジェクト間の類似度を求める際に，使用されていないライブラリほど重みを追加して類似度を算出している．そして，CrossRec と推薦システムとして広く用いられている LibFinder[2]，LibRec[3]，および LibCUP[5] と性能比較を行っている．その結果として，全ての評価指標において，従来の手法と同等かそれ以上の結果が得られたことを示している．

3. 実験設計

本章では，データセットの構築から推薦するライブラリの選出までの流れと推薦システムの評価指標を説明する．図 2 に推薦システムの概要を示す．以降では図 2 の流れに沿って調査手法を説明する．

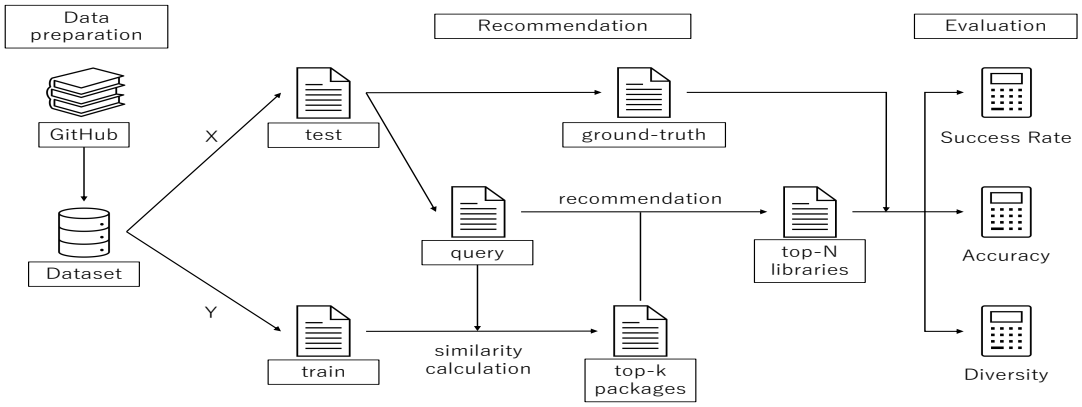


図 2 推薦システムの概要

3.1 データセットの構築

3.1.1 取得プロジェクトの選定

本研究では，多数のライブラリが公開されている GitHub 上のデータを用いて，調査を行う．Python の機械学習ライブラリにおいて，使用頻度の高い上位 3 件のライブラリは，Scikit-learn, Tensorflow, Xgboost であった [8]．この情報をもとに，GitHub 上でこれらのライブラリを最低 1 つ以上使用するプロジェクトのうち，スター数の多い各上位 100 件を重複を含んで取得する．

3.1.2 ライブラリ使用情報の取得

Python において，開発者はプロジェクトを構築する際に，‘requirements.txt’ という名前を含んだファイルに，開発段階で使用しているライブラリ，およびそのバージョンの情報を記すことが一般的である．そこで，本研究では，3.1.1 節で取得したプロジェクトに含まれる ‘requirements.txt’ ファイルのみからライブラリ使用情報を抜粋して使用する．その結果，プロジェクト数 190 件，ライブラリ数 1,384 件が取得できた．ただし，本研究は初期評価であるため，異なるバージョンでもライブラリ名が同じ場合，同一のライブラリと見なす．

3.1.3 評価行列の作成

協調フィルタリングを用いてライブラリを推薦するにあたり，プロジェクトとライブラリの関係を可視化するためのプロジェクト・ライブラリ評価行列（以下，評価行列）を作成する．評価行列において，各行には取得できた全プロジェクトが，各列には取得できた全ライブラリが割り当てられる．評価行列の各セルには，あるプロジェクトが指定されたライブラリを使用していた場合は 1 を，それ以外の場合は 0 を格納する（以下，ライブラリ使用情報）．

表 1 は，プロジェクト数 3 件（p_1, p_2, p_3），ライブラリ数 4 件（$lib_1, lib_2, lib_3, lib_4$）で作成された 3×4 評価行列の例である．各プロジェクトの ‘requirements.txt’ を参照すると，p_1 は lib_1, lib_3, lib_4，p_2 は lib_3, lib_4，p_3 は lib_1 を使用していたため，評価行列の該当するセルが 1 を示し，それ以外のセルは 0 を示している．

本研究では，3.1.2 節で取得したプロジェクト数 190 件，ライブラリ数 1,384 件を用いて，$190 \times 1{,}384$ の評価行列を作成し，データセットとして使用する．

3.2 推薦システム

本研究では，協調フィルタリングを用いた機械学習ライブラリ関連の推薦を行う．その推薦システムとして，2.3 節の Nguyen ら [4] の研究で紹介した CrossRec を参考にし，モデルを構築する．ただし，本研究は，Python の機械学習関連ライブラリ推薦の初期評価であるため，簡易的なシステムを構築している．よって，システム構造を変更することでより高精度な推薦を行うことができる可能性があり，システムの最良化に関しては今後の課題である．以下で，構築した手法について説明する．

3.2.1 データの分割

はじめに，データセットから test データとして X 件のプロジェクトを無作為に選び，残りの Y 件のプロジェクトを train データとする．次に，test データのライブラリ使用情報を無作為に半分欠損させて値を Nan とする．そして，欠損させたライブラリ使用情報を ground-truth データとし，残りの半分のライブラリ使用情報を query データとする．ground-truth データは，推薦対象プロジェクトが推薦されたライブラリを実際に使用しているかの比較，query データは，プロジェクト間の類似度を算出する際に使用する．

表 1　評価行列の例

	lib_1	lib_2	lib_3	lib_4
p_1	1	0	1	1
p_2	0	0	1	1
p_3	1	0	0	0

3.2.2　推薦スコアの算出

3.2.1 節で作成した test データと train データのプロジェクト間の類似度を，query データからコサイン類似度を利用して算出する．コサイン類似度の計算方法を以下に示す．

$$sim(p,q) = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \times \sqrt{\sum_{i=1}^{n} q_i^2}} \quad (1)$$

p は推薦対象プロジェクトのライブラリ使用情報，q は p の類似プロジェクトのライブラリ使用情報，$sim(p,q)$ はコサイン類似度，n は query データの数を表す．

算出された類似度のうち，上位 k 件のプロジェクトを train データから抽出し，抽出されたプロジェクトのライブラリ使用情報から test データのプロジェクトの欠損値を予測する（以下，推薦スコア）．本稿では，推薦スコアを上位 k 件の類似プロジェクトのライブラリ使用情報の平均値とする．表 1 を使用して，以下に推薦スコアの算出例を示す．p_1 を推薦対象プロジェクト，p_2，p_3 を p_1 の類似プロジェクト上位 2 件，枠で囲まれた部分を欠損値とする．lib_2 に対する p_2 および p_3 のライブラリ使用情報は 0 より，p_1 の lib_2 に対する推薦スコアは平均をとって 0 となる．同様にして，lib_3 に対する p_2 および p_3 のライブラリ使用情報は 1 と 0 より，p_1 の lib_3 に対する推薦スコアは 0.5 となる．

3.2.3　推薦と評価

3.2.2 節で算出した推薦スコアのうち，スコアの高い上位 N 件のライブラリを test データの推薦対象プロジェクトが使用する可能性の高いライブラリとして推薦し，ground-truth データを利用して評価指標値を算出し，推薦システムの初期評価を行う．ただし推薦されるライブラリは，モデリング機能を持たないライブラリも含まれる．これは，機械学習システムを開発する流れの中で，モデリング機能を持たないライブラリも同時に使用される可能性を考慮するためである．

3.3　評価指標の区分

推薦システムの成功率や精度，カバー率を評価するために，Nguyen ら [4] が用いた評価指標である，$SuccessRate@N$，$Precision@N$，$Recall@N$，$Coverage@N$，$Entropy@N$ を使用して評価を行う．本節ではこれらの評価指標を 3 つの区分に分けて説明する．これらの指標を用いたのは，推薦対象が Java である場合と比較するためである．

3.3.1　Success Rate（成功率）

推薦システムの成功率を Success Rate と定義し，$SuccessRate@N$ を評価指標として用いる．$SuccessRate@N$ の値が高いほど，システムが有効であることを示す．

$SuccessRate@N$：$SuccessRate@N$（成功率）は，test データのプロジェクトのうち，推薦されたライブラリを少なくとも 1 つ以上使用していたプロジェクトの割合を表す．以下に $Successrate@N$ の算出式を示す．

$$SuccessRate@N = \frac{count(|match_N| > 0)}{|P|} \quad (2)$$

P は test データに含まれるプロジェクト数，$match_N$ は推薦されたライブラリのうち，推薦対象プロジェクトが実際に使用していたライブラリの個数，$count(|match_N| > 0)$ は，推薦されたライブラリを最低 1 つ以上使用しているプロジェクト数を表す．

3.3.2　Accuracy（精度）

推薦システムの精度を Accuracy と定義し，$Precision@N$ および $Recall@N$ の 2 つを評価指標として用いる．各評価指標の値が高いほど，システムがより高精度であることを示す．

$Precision@N$：$Precision@N$（適合率）は，推薦された上位 N 件のライブラリのうち，推薦対象プロジェクトが実際に使用していたライブラリの割合を表す．以下に $Precision@N$ の算出式を示す．

$$Precision@N = \frac{|match_N|}{|N|} \quad (3)$$

N は推薦されたライブラリの個数を表す．

$Recall@N$：$Recall@N$（再現率）は，ground-truth データに含まれるライブラリが実際に推薦された割合を表す．以下に $Recall@N$ の算出式を示す．

$$Recall@N = \frac{|match_N|}{|GT|} \quad (4)$$

GT は，ground-truth データに含まれるライブラリの個数を表す．

3.3.3　Diversity（カバー率）

推薦システムの推薦のカバー率を Diversity と定義し，$Coverage@N$ および $Entropy@N$ の 2 つを評価指標として用いる．各評価指標の値が高いほど，システムがより幅広いライブラリを推薦していることを示す．

$Coverage@N$： $Coverage@N$ は，推薦可能なライブラリのうち，システムによって推薦されたライブラリの種類の割合を表す．以下に $Coverage@N$ の算出式を示す．

$$Coverage@N = \frac{|REC_N|}{|L|} \quad (5)$$

L は，データセットのライブラリ数を，REC_N は，test データの各プロジェクトに対して推薦された上位 N 件のライブラリの集合の要素数を表す．

$Entropy@N$： $Entropy@N$ は，推薦されたライブラリの偏りを評価する．以下に $Entropy@N$ の算出式を示す．

$$Entropy@N = -\sum_{l \in L} \left(\frac{rec(l)}{all}\right) \ln \left(\frac{rec(l)}{all}\right) \quad (6)$$

all は，test データの各プロジェクトに対して推薦された，重複を含んだライブラリの総数を，$rec(l)$ は，ライブラリ l が推薦された回数を表す．

4. 実験結果

本章では，第 3 章で作成したデータセットをもとに，3 つの各 RQ について，目的とアプローチ，そしてその調査結果を示す．ただし，各評価指標値 X の最大値および最小値を X_{max}，X_{min} と表記する．

4.1 RQ1: Python において機械学習関連ライブラリを推薦できるのか

目的：協調フィルタリングを用いた推薦システムが Python の機械学習関連ライブラリにも有効かどうかを明らかにする．

アプローチ： 3.1.2 節で取得したプロジェクト数 190 件について，図 2 の X，Y をそれぞれ 10 件，180 件として，学習を行う．そして，上位 N 件の推薦ライブラリを予測する際に使用する類似プロジェクト数 k を $k = \{5, 10\}$ に固定して，test データの 10 件の各プロジェクトに対して，$N = \{1, 3, 5, 7, 10\}$ と変更し，推薦を行う．類似プロジェクト数 $k = \{5, 10\}$ に設定した理由は，Nguyen ら [4] が $k = \{10, 20\}$ に設定しており，本研究で使用したデータセット数は Nguyen ら [4] の 10%であったためである．得られた結果をもとに $SuccessRate@N$，$Precision@N$，$Recall@N$，$Coverage@N$，$Entropy@N$ を算出し，評価を行う．なお，データによる偏りを防ぐため，各 N に対して test データおよび train データを変更して，10 回推薦を行い，算出された評価指標の値の平均値を使用する．

結果：調査の結果を図 3 に示す．

図 3(a) より，推薦ライブラリの数 N を増やすほど，$SuccessRate@N$ は高い値を示したが，$N \geq 3$ 以降，N の値を増加させてもほとんど値に変化は見られなかった．また，類似プロジェクト k の値を増加させても，変化は 5.0%以内であり，各 N に対して $SuccessRate@N$ の値にほとんど変化は見られなかった．

> ライブラリを 3 つ以上推薦した場合，80%以上の確率で各プロジェクトが使用しているライブラリを 1 つ以上推薦できた．また，類似プロジェクト数を変化させても，ほとんど成功率に影響はなかった．

図 3(b) より，推薦ライブラリの数 N を増やすほど，$Precision@N$ の値は低くなることが見て取れる．また，類似プロジェクト k の数を増加させると，全ての N の値において，$Precision@N$ は高い値を示し，$N = 1$ のとき，最大で 4.7%予測精度が向上した．

> ライブラリの推薦数を減少，または類似プロジェクトの値を増加させると，予測精度は向上した．

図 3(c) より，推薦ライブラリの数 N を増やすほど，$Recall@N$ は高い値を示した．また，類似プロジェクト数 k を増加させると，全ての N の値において，$Recall@N$ は高い値を示し，最大で，0.049%再現率が向上した．

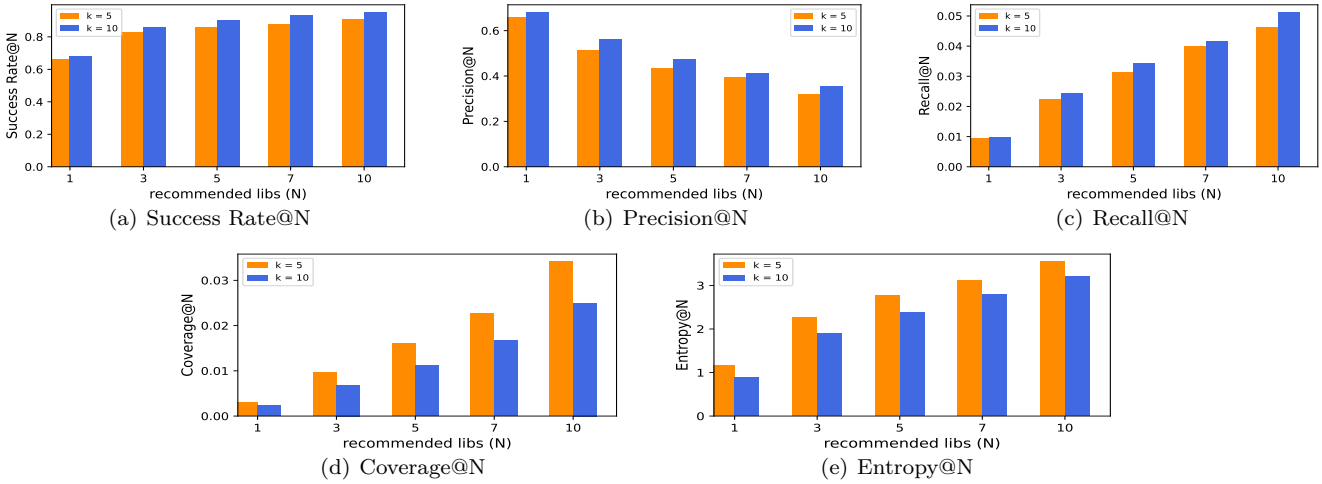
> ライブラリの推薦数または，類似プロジェクト数を増加させると，推薦システムの再現率は向上した．

図 3(d) より，推薦ライブラリの数 N を増やすほど，$Coverage@N$ は高い値を示した．また，類似プロジェクト数 k を増加させると，全ての N において推薦されるライブラリの種類は少なくなり，最大で 0.91%減少した．

> ライブラリの推薦数を増加，または類似プロジェクト数を減少させると，推薦できるライブラリの種類は豊富になる．

図 3(e) より，推薦ライブラリの数 N を増やすほど，$Entropy@N$ は高い値を示した．また，類似プロジェクト数 k を増加させると，全ての N の値において，一定のライブラリに推薦が集中していた．

> 推薦ライブラリの数を増加させると，推薦できるライブラリの種類は豊富になるが，類似プロジェクト数を増加させることで，一定のライブラリに推薦を集中できる．



(a) Success Rate@N (b) Precision@N (c) Recall@N

(d) Coverage@N (e) Entropy@N

図 3 RQ1 で得られた各評価指標値

4.2 RQ2: データ数を変更した際，推薦結果に変化が生じるのか

目的： 本研究で使用したデータセットのプロジェクト数は 180 件，ライブラリ数は 1,384 件であるが，推薦システムを構築する際に参考にした Nguyen ら [4] の研究で使用されたデータセットは，どちらも約 10 倍ほど大きい．そこで，データセットのプロジェクト数，およびライブラリ数を増加させれば，予測精度は向上するのか各評価指標値を用いて調査する．

アプローチ： 図 2 の test データ $X = 10$ と train データ $Y = 180$ に分割後，さらに Y から無作為に Z 件抽出し，それを新たな train データとする．そして，図 2 の要領で学習，および推薦を行う．RQ1 同様，データによる偏りを防ぐため，Z に対して test データおよび train データを変更して，10 回推薦を行い，算出された各評価指標値の平均値を使用する．そして，Z の値を $Z = \{30, 40, \cdots, 170, 180\}$ と 10 件ずつ増加させて同様に実験を行い，評価指標の推移を分析する．なお，各 Z において，test データは全て同一のものを使用し，図 2 の k，N の値は，$N = 1$ の場合を除いて，RQ1 で $Precision@N$ の値が最も高かった $k = 10$，$N = 3$ に固定して，実験を行う．

結果： 調査の結果，最大で $SuccessRate@N$ は 5.0%，$Precision@N$ は 4.0%，$Recall@N$ は 0.17%，$Coverage@N$ は 0.26%，$Entropy@N$ は 0.33 の差が生じたが，大きな精度向上は認められなかった．

> 学習データの数を増加させても，推薦システムの予測精度に大きな変化は見られなかった．

4.3 RQ3: ライブラリの種類によって推薦頻度に差が生じるのか

目的： 推薦システムによって推薦されたライブラリの推薦頻度を調査し，実際にライブラリの種類によって推薦結果に偏りが生じているのか明らかにする．また，偏りが生じていた場合，Python において推薦されやすいライブラリの特徴を明らかにする．

アプローチ： RQ1 において，$k = \{5, 10\}$，$N = \{1, 3, 5, 7, 10\}$ の全ての場合において推薦されたライブラリの回数を調査する．

結果： RQ1 において推薦されたライブラリの内，推薦回数上位 10 件のライブラリと推薦回数，およびライブラリの総推薦数に対する推薦回数の割合を表 2 に示す．

表 2 より推薦回数上位 10 件のライブラリの内，scikit-learn と tensorflow を除く 8 件がモデリング機能を持たないライブラリであった．また推薦された全ライブラリの内，上位 10 件のライブラリが 31%を占めていた．

> ライブラリの種類によって推薦頻度に差が生じており，推薦回数の多かった上位 10 件のライブラリが，推薦された全ライブラリの 31%を占めていた．

5. 考察

本章では，Nguyen らの研究 [4] の Dataset D1 で得られた 5 つの評価指標値と比較することで，実験結果を考察する．Dataset D1 で得られた各評価指標値を使用したのは，本研究の各評価指標と同じだからである．

表 2　RQ1 において推薦されたライブラリ上位 10 件と推薦回数

ライブラリ	scikit-learn	numpy	pandas	scipy	matplotlib	sphinx	tqdm	pytest	tensorflow	pytest-cov
回数	535	430	414	398	226	191	190	181	178	127
割合	0.10	0.083	0.080	0.077	0.043	0.037	0.037	0.035	0.034	0.024

表 3　Java と Python における各評価指標の最大値と最小値 (Java の結果については D1 の結果を抜粋 [4])

	SuccessRate@*N*		*Precision*@*N*		*Recall*@*N*		*Coverage*@*N*		*Entropy*@*N*	
	max	min	max	min	max	min	max	min	max	min
Java	0.96	0.70	0.7 前後	0.2 前後	0.3 前後	0.05 前後	0.059	0.0081	1.2	0.13
Python	0.94	0.60	0.67	0.33	0.051	0.095	0.033	0.0025	3.5	0.97

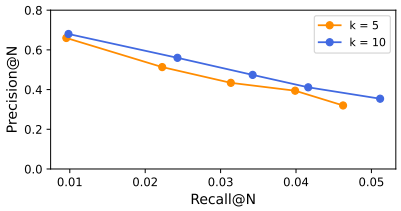


図 4　Precision-Recall 曲線

5.1　先行研究との比較

Success Rate：表 3 より，最大値は 0.02，最小値は 0.10，Java の方が高い値を示した．また先行研究と図 3(a) より，Java および Python における *SuccessRate*@*N* は，推薦ライブラリの数 N を増加させるほど高い値を示しているが，$N \geq 3$ 以降，ほとんど値に変化はなかった．さらに，k の値を増加させても，各 N において，*SuccessRate*@*N* の値にほとんど差は生じなかった．

> N，k の変化に伴う *SuccessRate*@*N* の値の推移は類似した傾向を示した．

Accuracy：RQ1 で得られた結果をもとに，$N = \{1, 3, 5, 7, 10\}$，$k = \{5, 10\}$ の場合の *Precision*@*N*，*Recall*@*N* の値を図 4 に示す．Precision-Recall 曲線（以下，PRC）は値が右上にいくほど，高精度であることを示す [9]．Java においては，*Precision*@*N* と *Recall*@*N* の値はトレードオフの関係を示し，図 4 より，Python においても同様の傾向が見られた．*Recall*@*N* の値が低い理由は，test データの数と推薦ライブラリ数 N の違いに起因すると考えられる．

> Java, Python ともに *Precision*@*N* と *Recall*@*N* はトレードオフの関係を示した．

Diversity：表 3 と先行研究より，*Coverage*@*N* において，最大値は 0.026，最小値は 0.0056，Java の方が高く，より多くの種類のライブラリを推薦できている．また，*Entropy*@*N* において，最大値は 2.3，最小値は 0.84，Java の方が低く，推薦の再現性が高かった．この違いも *Recall*@*N* と同様，test データ数と推薦ライブラリ数 N，さらに推薦対象の全ライブラリ数に起因すると考えられる．Java における *Coverage*@*N* と *Entropy*@*N* は，推薦ライブラリ数 N が増加するほど高い値を示し，全ての N において，類似パッケージ数 k を増やすと値が減少しており，図 3(d)，(e) より，Python においても同様の傾向が見られた．

> N，k の変化に伴う *Coverage*@*N* および *Entropy*@*N* の値の推移は類似傾向を示した．

以上の比較より，CrossRec[4] の手法を参考にした推薦システムは，対象言語を Python，推薦対象を機械学習関連ライブラリとした場合にも有効であると考えられる．

5.2　推薦対象を Python に変更したことによる特徴

5.1 節より，Java と Python において類似パッケージ数 k と推薦ライブラリ数 N の値の変化による各評価指標値の推移は類似した傾向を示したが，Java の方が推薦精度が高かった．この要因としては，データセット数が Java の場合よりも少ないこと，推薦スコアの算出方法，さらに Java と Python におけるライブラリ同士の依存性が挙げられる．

1 つ目のデータセット数が少ないという要因に関して，Nguyen ら [4] の研究では本研究で使用したパッケージ数 190 件，ライブラリ数 1,384 件のデータセットの約 10 倍のデータセットを使用して実験を行っている．そのため，RQ2 でデータ数を変更して調査したものの，本研究の推薦システムにおいては，データセットの数を増加させても評価指標値に変化は認められなかった．こ

のことと，Python においてデータ数が少ないものの，*SuccessRate*@*N* と *Precision*@*N* は高い値を示したことより，どのパッケージも同じライブラリを使用しているため，推薦が容易であった可能性がある．

そこで，RQ3 にて推薦されたライブラリの推薦頻度について調査し，表 2 を得た．結果より，推薦されるライブラリの傾向として機械学習関連ライブラリが多く，推薦された全ライブラリの内，上位 10 件のライブラリが 31%を占めていたので，機械学習システムを開発する上でどの開発者も同じ機械学習関連ライブラリを使用していると考えられる．また，推薦されたライブラリは Python においてよく使用されるライブラリであるため，開発者にとって有用であるとはいえない．推薦スコアを算出する際にライブラリ毎に重みを付与し，本研究の調査で推薦スコアが下位のライブラリを推薦できるよう改善する必要がある．

今後，推薦対象を機械学習ライブラリに絞った調査，query のデータ数を変更した際の各評価指標値の推移の調査，推薦システムの構造変更，データ数を拡張した調査，およびライブラリの依存性を考慮して調査を行う必要がある．

6. 妥当性への脅威

6.1 内的妥当性

本研究では，GitHub 内における Python の機械学習ライブラリ Scikit-learn, Tensorflow, Xgboost を最低 1 つ使用しているプロジェクト内の ‘requirements.txt’ からライブラリ使用情報を取得して調査を行った．その理由としては，Python を利用する多くの開発者が ‘requirements.txt’ ファイルを使用して，開発プロジェクトのライブラリ使用情報を管理するからである．ただし，ファイル名は任意であるため，開発者によっては，ファイル名に ‘requiments.txt’ を含んでいない，あるいは，別の拡張子を使用してライブラリ使用情報を管理している可能性がある．そのため，対象にすべきファイルを取得できていない可能性がある．さらに，‘requiments.txt’ 内におけるコメントの内容は考慮していないため，対象のライブラリ使用情報を取得できていない可能性がある．

6.2 外的妥当性

本研究では，Python の機械学習ライブラリの使用率上位 3 件 [8] のライブラリを最低 1 つ使用しているプロジェクトのうち，GitHub でのスター数各上位 100 件を対象に調査を行なったが，調査結果をより一般化するためには，依存ライブラリ数，および対象のプロジェクト数を増加させて調査を行う必要がある．

7. おわりに

本研究では，協調フィルタリングを用いて Python における機械学習関連ライブラリを推薦し，5 つの評価指標を用いて推薦システムの初期評価を行った．調査の結果，評価指標値の推移は対象言語が Java の場合と類似した傾向を示したため，対象言語を Python に変更した場合も，協調フィルタリングを用いた推薦手法は有効であることを明らかにした．

今後の課題として，ライブラリの依存性とバージョン情報を組み込んだ推薦，train データの query のデータ数を変更した調査，データ数を拡張した調査，および推薦システムの構造変更が挙げられる．

謝辞

本研究の一部は，JSPS 科研費 JP18H04097, JP21H04877，および，JSPS・スイスとの国際共同研究事業（JPJSJRP20191502）の助成を受けた．

参考文献

[1] Masateru Tsunoda, Takeshi Kakimoto, Naoki Ohsugi, Akito Monden, and Ken-ichi Matsumoto. Javawock: A java class recommender system based on collaborative filtering. In *Proc. of the SEKE'2005*, pp. 491–497, 2005.
[2] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, and K. Inoue. Search-based software library recommendation using multi-objective optimization. *IST*, Vol. 83, pp. 55–75, 2017.
[3] F. Thung, D. Lo, and J. Lawall. Automated library recommendation. In *Proc. of the WCRE'2013*, pp. 182–191. IEEE, 2013.
[4] P. T. Nguyen, J. D. Rocco, D. D. Ruscio, and M. D. Penta. CrossRec: Supporting software developers by recommending third-party libraries. *JSS*, Vol. 161, p. 110460, 2020.
[5] M. A. Saied, A. Ouni, H. Sahraoui, R. G. Kula, K. Inoue, and D. Lo. Improving reusability of software libraries through usage pattern mining. *JSS*, Vol. 145, pp. 164–179, 2018.
[6] 桂川大輝, 伊原彰紀, 松本健一ほか. ソフトウェア開発において併用されるライブラリ機能の推薦. 研究報告ソフトウェア工学 (SE), Vol. 2017, No. 5, pp. 1–7, 2017.
[7] 亀井靖高, 角田雅照, 柿元健, 大杉直樹, 門田暁人, 松本健一. 進行中のプロジェクトに有用なソフトウェアコンポーネントの推薦方法. 電子情報通信学会技術研究報告; 信学技報, Vol. 106, No. 16, pp. 25–30, 2006.
[8] Kaggle. State of data science and machine learning 2021.
[9] T. D. Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *Proc. of the I-SEMANTICS '2012*, pp. 1–8, 2012.