GitHubとStack Overflowの開発者の活動記録を併用したリポジトリ推薦

永野 真知[1,a)]　早瀬 康裕[1,b)]　駒水 孝裕[1,c)]　北川 博之[1,d)]

概要：ソフトウェア開発プロジェクトを共有するサービスである GitHub は，プロジェクトに関するソースコードの変更履歴や，変更に関する議論をリポジトリに蓄積している．開発者は，GitHub から関心のあるリポジトリを探して開発に携わったり，開発に関する知見を得るためにリポジトリを閲覧する．そのため，もし開発者に対して関心に合うリポジトリを推薦できれば，リポジトリを発見する手間を省くだけでなく，新たな知識を得る機会を増やすことができると考えられる．しかし，平均的な開発者が GitHub 上でリポジトリは少ないため，推薦に利用できる情報が少ないという課題がある．そこで本研究では，ソフトウェアやプログラミングに関する Q&A サービスである Stack Overflow の質問と回答の履歴を，GitHub での活動記録と組み合わせることで，リポジトリ推薦の精度を向上させる手法を提案する．提案手法を，GitHub のデータのみを用いて推薦する場合と比較した結果，多くの場合において提案手法がより高い精度で推薦を行えることが確認できた．

1. 序論

近年，ソフトウェア開発を支援する Web 上のサービスが活発に利用されるようになっており，それらのサービスは開発に関する膨大な記録を蓄積している．特に，ソフトウェア開発を支援する GitHub [*1]（以下，GH）や，ソフトウェアやプログラミングに関する Q&A の場を提供する Stack Overflow [*2]（以下，SO）は，多数のユーザによって利用されている．これまで行なわれた多くの研究において GitHub や SO の記録が分析され，ソフトウェア開発に役立つ知見がもたらされている．

GH は，ソフトウェア開発における変更記録システム Git リポジトリのホスティングに加えて，リポジトリを通じて複数人で開発を行うことを支援する機能を統合したサービスである．GH は，Git リポジトリへのコミットの記録や，システム上で開発者が行なった議論を蓄積している．近年のオープンソースソフトウェア (OSS) 開発では，GH を用いることが主流となっている．

一方，SO はプログラミングやソフトウェアに関する Q&A の場を提供するサービスであり，開発者は質問や回答を自由に投稿することができる．質問は，技術分野を表すタグを付けて分類するルールになっている．蓄積された質問や回答は Web を通じて公開され，世界中の開発者によって参照されている．

開発者は，GH 上のリポジトリを，様々な目的で検索する．一般のソフトウェア開発者は，自身の関わるソフトウェア開発プロジェクトで必要となる知識や技術の獲得を目的として， GH 上のリポジトリを参照する．また，OSS 開発者は，自分の関心に合致し，かつ力を発揮できるリポジトリを探し，そこで活動を行おうとする．以上のことから，開発者に対して，関心を持つであろうリポジトリを推薦することは，開発者にとっての有益な情報提供となりうることがわかる．

しかし，開発者に対してリポジトリを推薦する際には，開発者が参加するリポジトリが少ないことが障害となる．平均的な GH 上のソフトウェア開発者がコミットやプルリクエストなどで貢献しているリポジトリの数は 3 個程度である [9]．そのため，GH 上の活動記録のみでは，開発者の関心を捉えるための情報が不足しており，そこから有効な推薦を行うことは難しかった．また，GH に参加してから日が浅い開発者は記録がさらに少ないため，関心を捉えることはますます難しい．

そこで本研究では，GH の活動記録に SO の活動記録を加えることで，開発者に対する GH リポジトリ推薦の精度

[1] 筑波大学
University of Tsukuba
[a)] machi@kde.cs.tsukuba.ac.jp
[b)] hayase@cs.tsukuba.ac.jp
[c)] taka-coma@acm.org
[d)] kitagawa@cs.tsukuba.ac.jp
[*1] https://github.com/
[*2] http://stackoverflow.com/

表 1　GH のデータセット

ID	Email	リポジトリ	コミット日時
g1	abc@xxx.com	リポジトリ1	2015/07/31
g1	abc@xxx.com	リポジトリ 3	2015/10/09
g2	efg@xxx.com	リポジトリ 1	2015/03/12
g2	efg@xxx.com	リポジトリ8	2016/09/10
g2	efg@xxx.com	リポジトリ10	2015/05/25
g3	hij@xxx.net	リポジトリ3	2015/12/20
g3	hij@xxx.net	リポジトリ8	2016/02/03
⋮			

表 2　SO のデータセット

ID	Email	タグ	編集した日時
s1	a007be5a61f6aa8	タグ1	2015/06/30
s1	a007be5a61f6aa8	タグ 3	2015/08/09
s1	a007be5a61f6aa8	タグ7	2015/01/12
s2	51d623f33f8b830	タグ8	2016/05/10
s2	51d623f33f8b830	タグ11	2015/03/25
s2	51d623f33f8b830	タグ3	2015/10/20
s3	b437f461b3fd273	タグ1	2016/04/03
⋮			

を高めることを狙う．SO は質問や回答を気軽に投稿することができるため，開発者の知識や技術的興味について多くの情報を得ることができる．この SO 上の活動記録を GH 上の活動記録を併用することで，GH のみでは入手が難しかった開発者の関心を利用した推薦が可能になると期待される．

本研究の貢献は以下の通りである．

- GH と SO における開発者の活動記録を併用することで，リポジトリ推薦の精度を改善する手法を提案する．
- 評価実験を通じて GH と SO の活動記録を併用することの有効性を示す

本論文の構成は以下のようになっている．2 節は本研究の提案手法の説明，3 節は提案手法を用いた評価実験を示す．4 節は関連研究，5 節は結論と今後の課題となっている．

2.　提案手法

本節では，ソフトウェア開発者向けサービスである GH と SO に蓄積された開発者の活動記録を併用することで，ソフトウェア開発者へリポジトリをより高い精度で推薦する手法を提案する．手法の概要を，図 2 に示す．提案手法は，2 つの段階に分かれている．1 つめの段階であるモデル学習では，GH と SO の両方で活動したことのある開発者の，活動記録を入力として受け取り，推薦に必要となるモデルを構築する．2 つめの段階であるリポジトリ推薦では，開発者の要求に応じて，その開発者が参加を望むと考えられるリポジトリを，1 つめの段階で構築されたモデルに基づいて推薦する．モデル構築と学習では，行列分解に基づく手法を用いる．

以下は順に，入力のデータ形式および実際に用いたデータと，手法の詳細な手順について説明する．

2.1　データセット

2.1.1　GH のデータセット

GH のデータセットとして， MSR 2014 Mining Challenge [6] に向けて公開された，GH の 2008 年 7 月 31 日から 2012 年 7 月 31 日の活動記録データを利用する．このデータセットには様々な情報が記録されているが，そのうち本研究で利用する部分の構造と例を，表 1 に示す．この表には，どの開発者が，いつ，どのリポジトリにコミットしたかが記録されている．開発者にはデータセット内で一意な ID が割り振られており，開発者の登録したメールアドレスも含まれている．

2.1.2　SO のデータセット

SO のデータセットとして，MSR 2014 Mining Challenge [3] に向けて公開された，SO の 2008 年 7 月 31 日から 2012 年 7 月 31 日の活動記録データを利用するこのデータセットには様々な情報が記録されているが，本研究は開発者の技術分野への関心を抽出すること目的として，質問や回答を投稿したタグに着目する．表 2 に，本研究で利用する部分の構造と，例を示す．この表には，どの開発者が，いつ，どのタグにおいて質問や回答を行なったかが記録されている．開発者にはデータセット内で一意な ID が割り振られており，同時に，開発者の登録したメールアドレスのハッシュ値も含まれている．

2.1.3　データの前処理

GH と SO のデータから本研究では両方のサービスに登録している開発者の活動記録を利用する．まず，GH と SO の両 方に登録している開発者を Email アドレスから推定する．推定して得られた開発者が関与するリポジトリとタグを本研究で利用する．

GH と SO の両方に登録している同一開発者を [16] で行われてたように Email アドレスに関する情報から推定する．図 2.1.3 は，Email アドレスで同一開発者を推定する様子を表している．GH のデータセットの開発者の情報には Email アドレスが記載されている．一方で SO のデータに含まれる開発者の情報には Email アドレスの MD5 ハッシュ値が記載されている．そこで，GH に登録している開発者の Email アドレスの MD5 ハッシュ値を計算し，同じハッシュ値を持つ SO の開発者を同一開発者として推定する．推定した開発者には新しく ID を振り分ける．

得られた同一開発者に対して，その開発者がコミットしたリポジトリと関与したタグを抽出する．同一開発者の GH での ID を利用し，コミットの情報からこの開発者がコミットしたリポジトリを抽出する．関与したタグは，開発者が投稿や編集またはコメントを行った質問や回答に関連するタグとする．データセットの投稿および編集の履歴とコメントの情報から，同一開発者の ID を用いてこの開発者が関与した投稿の ID を見つける．そして，データセット

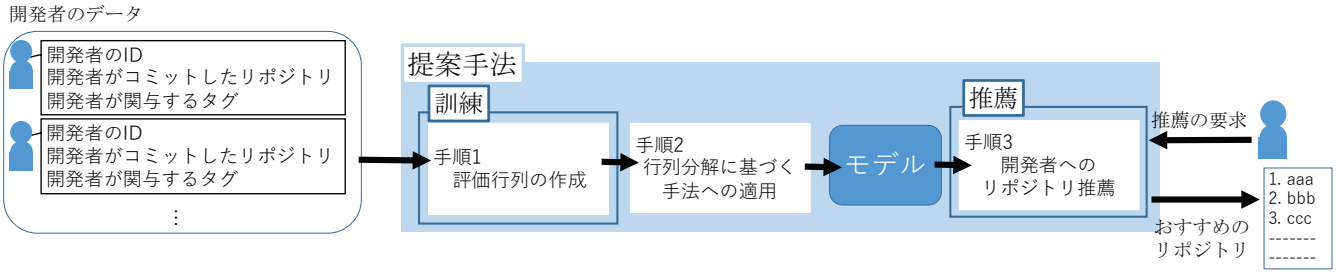

図 1　手法の概要

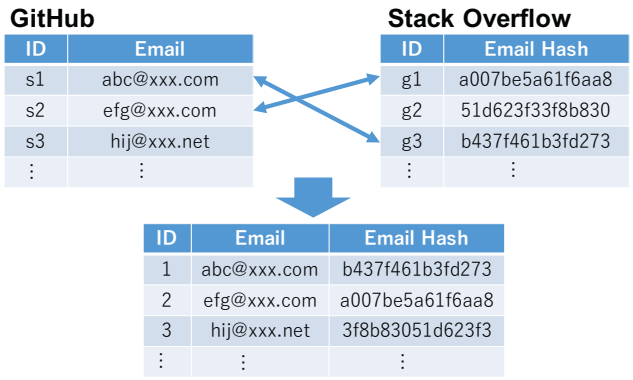

図 2　GH と SO にまたがる開発者の推定

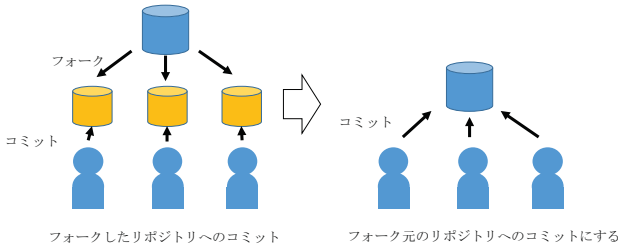

図 3　フォークされたリポジトリへのコミットの処理

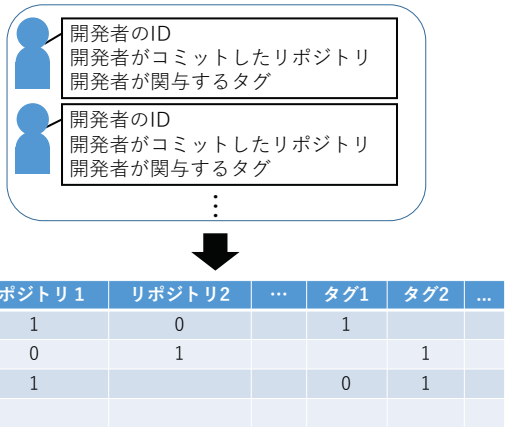

図 4　同一開発者による評価行列の結合

の投稿の情報から，投稿の ID と結びついているタグを見つける．同一開発者には新しく ID を付与し，各開発者に対して ID と開発者がコミットしたリポジトリ，そして開発者が関与したタグの情報を保持するデータを作成する．

こうして得られた同一開発者のデータには，個人用のリポジトリやほとんど使われないタグが存在しているため，それらを調整する．活動記録から得られるリポジトリには個人用として使用するためのフォークしてきたリポジトリが含まれる．フォークしたリポジトリへのコミットは，フォーク元のリポジトリに反映させるために作業したものが多い．そこで，クローン数が少ない不人気のリポジトリへのコミットをフォーク元のリポジトリへのコミットとする．これを図に表したものが図 2.1.3 である．また，タグに関しては，開発者からの利用頻度が低いものは除外する．本節ではクローン数の閾値を 5，タグの利用頻度を 10 とする．

上記の処理をして得られた同一開発者のデータを本節で利用する．同一開発者の各開発者には次のデータが保持されている．

- 開発者の ID
- 開発者がコミットしたリポジトリ
- 開発者が関与したタグ

2.2　手法の手順

2.2.1　手順 1: 評価行列の作成

入力で得られた開発者と開発者がコミットしたリポジトリ，開発者が関与するタグから評価行列を作成する．評価行列の作成は以下のようになる．開発者数を n，リポジトリ数を m，タグ数を l とするとき，$n \times (m+l)$ の行列 R を作成する．開発者 u の評価を表す行列 R の u 行目は $R_u = (r_0, r_1, ..., r_m, t_1, t_2, ..., t_l)$ となる．r_i はリポジトリを表し，t_i はタグを表している．u がリポジトリ r_i にコミットした場合，R_u の r_i には 1 を挿入する．また，u がタグ t_i に関与する場合，R_u の t_i には 1 を挿入する．これを図 2.2 に示す．図 2.2 の ID は開発者の ID を示している．開発者の ID が 1 でリポジトリ 1 の要素には 1 が挿入されている．これは ID が 1 の開発者がリポジトリ 1 にコミットしたことを表している．また，ID が 2 でリポジトリ 1 の

要素にはランダムに挿入した 0 が入っている．しかし，この評価行列にはポジティブな値か欠損値しか含まれていない．そこで，本手法では Pan らの手法 [12] に基づき欠損値にランダムに 0 を代入する．0 を挿入する割合は評価行列に含まれる 1 の割合と同等とする．

2.2.2 手順 2: 行列分解に基づく推薦手法への適用

前節で得られた評価行列 R を行列分解に基づく推薦手法へ適用し，推薦に必要なモデルを得る．具体的には，Python のライブラリである Crab - scikit.recommender [*3]の scikits.crab.recommenders.svd.classes 内にある MatrixFactorBasedRecommender を用いる．当該アルゴリズムは，行列 R を確立的勾配法を用いて $n \times s$ の行列 U と $(m+l) \times s$ の行列 I に分解する．(s は潜在因子数) 2 つの行列 U，I が，推薦に用いるモデルとなる．

2.2.3 手順 3: 開発者へのリポジトリ推薦

入力で得た推薦対象の開発者に対して，手順 2 で求めた行列 U，I を利用してリポジトリの推薦を行う．開発者 u のリポジトリ r_i への予測値 $\hat{R_{u,r_i}}$ は次の式で表せる．

$$\hat{R_{u,r_i}} = \sum_s U_{u,s} \cdot I_{r_i,s} \tag{1}$$

まず，入力で得た推薦対象の開発者 u のすべてのリポジトリに対する予測値 $(\hat{R_{u,r_1}}, \hat{R_{u,r_2}}, ..., \hat{R_{u,r_m}})$ を求める．そして，予測値 $\hat{R_{u,r_i}}$ が高い r_i を開発者に推薦する．推薦するリポジトリの個数は予測値 $\hat{R_{u,r_i}}$ の高い順に任意の個数とする．この推薦する個数は 6 節で調節する．ただし，推薦するリポジトリは今まで開発者がコミットしたことのないリポジトリとする．また，評価行列で欠損値に 0 を代入したリポジトリも推薦するリポジトリの対象となる．

3. 評価実験

提案手法の評価を目的として，GH のデータのみを利用したベースライン手法との比較実験を行う．実験では提案手法とベースライン手法それぞれの，リポジトリの推薦精度を比較し，提案手法がより高い精度で推薦できるかどうかを確認する．評価基準として再現率と，適合率，F 値を用い，2 つの手法によって上位に推薦されたリポジトリが，正解と一致するかを測定する．

以下，ベースライン手法の定義と，評価実験の詳細な手順，評価に用いたデータ，実験結果の順で説明する．

3.1 ベースライン手法

提案手法に対する比較手法として，GH のデータのみを利用した行列分解に基づく推薦手法をベースライン手法として採用する．ベースライン手法は，提案手法の評価行列を差し替え，GH のデータのみで構築した評価行列に変更したものである．

[*3] http://muricoca.github.io/crab/

GH のデータのみをした評価行列の作成方法は，以下の通りである．各開発者の ID と，開発者がコミットしたリポジトリを利用し，開発者とリポジトリからなる評価行列 R を作成する．開発者数を n，リポジトリ数を m とすると，行列 R は $n \times m$ の行列となる．開発者 u の評価を表す行列 R の u 行目は $R_u = (r_0, r_1, ..., r_m)$ となる．r_i はリポジトリを表している．u がリポジトリ r_i にコミットした場合，R_u の r_i に 1 を挿入する．評価行列の欠損値は 1 と同数の 0 をランダムに代入する．

3.2 評価の方法

提案手法とベースライン手法で同一の開発者に対してリポジトリ推薦を行い，その結果を比較する．訓練データと評価データの分割の方法としては，交差検定と，時系列に前半と後半に分け，前半を訓練データ，後半を評価データとして用いる方法の，2 通りを用いる．評価指標としては，再現率と適合率，F 値の 3 種類を用いる．

3.2.1 正解の定義と評価基準

ある開発者に対するリポジトリが正解であるかを判断する基準を，以下のように定義する．

正解リポジトリ その開発者が評価データにコミットしたリポジトリであり，かつ，その開発者が訓練期間でコミットしていないリポジトリ

この正解リポジトリの定義に基づいて，推薦結果のリポジトリのリストを評価する 3 つの指標を，以下のように定義する．$D_u = \{d_1, d_2, ..., d_m\}$ を, 開発者 u に対する全正解リポジトリの集合とする．また，開発者 u に推薦されたリポジトリのリストを $R_u = (r_1, r_2, ..., r_n)$ と表す．R_u は尤もらしさが高い順に整列されている．r_i が正解リポジトリ集合 D_u に含まれていれば，$x_i = 1$ とする．以上の定義に基づき，推薦リポジトリリストの上位 k 位までの，再現率，適合率および F 値を以下のように定義する．

$$\text{再現率@}k = \frac{1}{|D_u|} \sum_{1 \le i \le k} x_i \tag{2}$$

$$\text{適合率@}k = \frac{1}{k} \sum_{1 \le i \le k} x_i \tag{3}$$

$$\text{F 値@}k = \frac{2 \times \text{適合率@}k \times \text{再現率@}k}{\text{適合率@}k + \text{再現率@}k} \tag{4}$$

3.2.2 交差検定の手順

限られたデータセットを有効に活用することを目的として，10 分割交差検定による評価を行う．まず，データセットから得られる GH の開発者とコミットしたリポジトリの全てのペアをランダムに 10 分割にする．分割した GH のデータを $G = (g_1, g_2, ..., g_{10})$ とする．次に，10 分割したうちの 1 つである g_i を評価データ，残りの 9 つのデータを訓練データとする．提案手法とベースライン手法の双方で，訓練データを用いてモデルを構築し，評価データを用いて

各開発者に対する推薦の再現率 @k，適合率 @k，F 値 @k を求める．以上の操作を，評価データを $g_1, g_2, ..., g_{10}$ と順に変化させながら繰り返す．提案手法の再現率 @k とベースライン手法の再現率 @k を評価データと開発者の対で対応付け，ウィルコクソンの符号順位付検定を行う．交差検定で得られた全ての再現率 @k の平均について提案手法とベースライン手法を比較する．これらの結果から，再現率 @k について，提案手法の精度がベースライン手法と比較して改善されるかを確認する．適合率 @k と F 値 @k についても再現率と同様に提案手法とベースライン手法を比較する．

評価する際，各開発者に推薦するリポジトリの件数 k と行列分解の潜在因子数を変化させて各手法の比較を行う．行列分解に基づく推薦手法では行列分解で用いる潜在因子数によって推薦結果に差があるため，潜在因子数を 1-30 と変化させて最適な潜在因子数を見つける．また，再現率@k，適合率@k，F 値@k は推薦件数 k の値によって変化するため，推薦件数を 10，20，40，80 件で変化させ実験を行う．有意水準 1% でウィルコクソンの符号順位検定を行う．

3.2.3 データを時系列順に前後に区切る手順

より現実に近い評価を行うことを目的として，開発者の過去の活動記録から，開発者が将来に参加するリポジトリを予測することを模擬した実験を行う．GH と SO の双方のデータについて，開発者によるリポジトリとタグへの参加を，期間の前後で分割する．分割の境界は 2012 年 5 月 1 日とし，訓練データの期間を 2008 年 08 月 01 日～2012 年 04 月 30 日，評価データの期間を 2012 年 05 月 01 日～2012 年 07 月 31 日とする．訓練データを用いてモデルを構築し，評価データを用いて各開発者に対する推薦の再現率 @k，適合率 @k，F 値 @k を求める．得られた全ての開発者の再現率 @k の平均について提案手法とベースライン手法を比較する．また，同じ開発者の提案手法の再現率 @k とベースライン手法の再現率 @k を対応付け，ウィルコクソンの符号順位付検定を行う．これらの結果から，再現率 @k について，提案手法の精度がベースライン手法と比較して改善されるかを確認する．適合率 @k と F 値 @k についても再現率と同様に提案手法とベースライン手法を比較する．

評価する際，各開発者に推薦するリポジトリの件数 k と行列分解の潜在因子数を変化させて各手法の比較を行う．3.2.2 節と同様に潜在因子数を 1-30 と推薦件数を 10，20，40，80 件と変化させて実験を行う．有意水準 1% でウィルコクソンの符号順位検定を行う．

3.3 データセット

本実験では，2.1.3 節で前処理をして得られた GH と SO を組み合わせたデータを利用する．前処理をして得られたデータセットのリポジトリ数が 88 個，タグ数が 7173 個である．

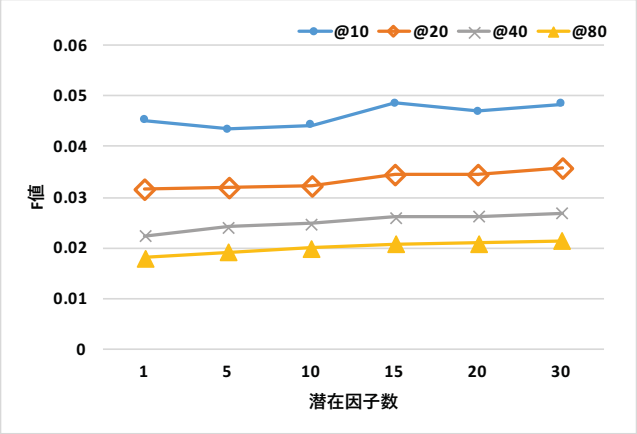

図 5 交差検定での潜在因子数と推薦件数に対する F 値の遷移の比較

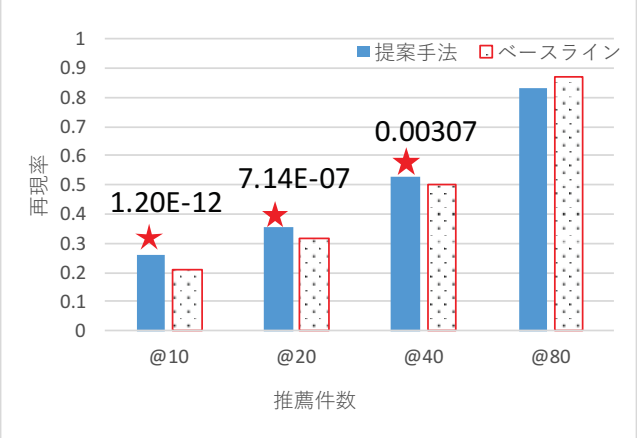

図 6 交差検定での潜在因子数を 15 とした推薦件数に対する再現率の比較

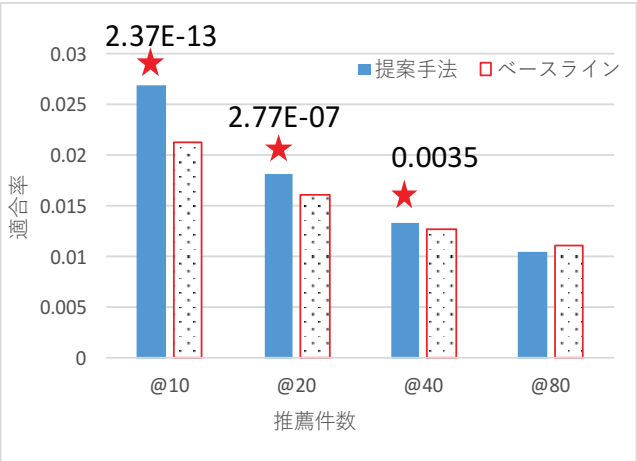

図 7 交差検定での潜在因子数を 15 とした推薦件数に対する適合率の比較

3.4 交差検定における実験結果

交差検定を行った場合の実験結果について示す．まず，潜在因子数についてどのように推薦精度が変化するのか確

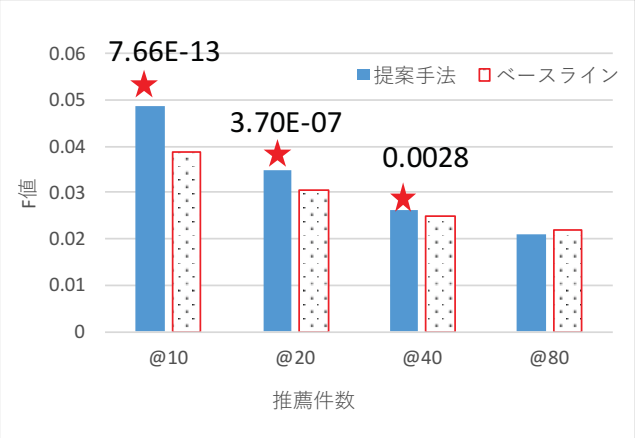

図 8　交差検定での潜在因子数を 15 とした推薦件数に対する F 値の比較

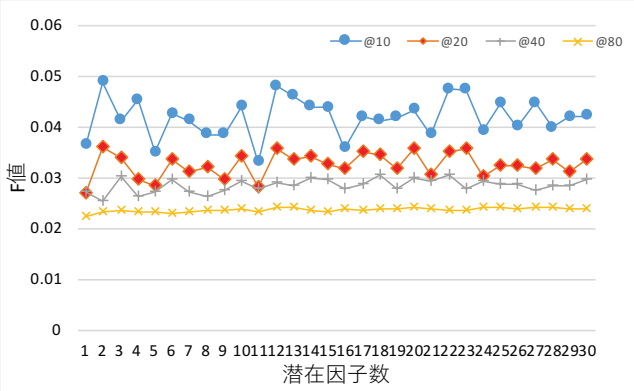

図 9　潜在因子数と推薦件数に対する F 値の遷移の比較

かめる．図 5 は縦軸が F 値，横軸が潜在因子数となっている．F 値 @k の潜在因子数による変化を表している．k は 10，20，40，80 と変化させている．F 値は再現率と適合率の調和平均であるため，潜在因子数の変化を見るために利用した．@10 の場合，潜在因子数 15 でよくなり，その後は値に大きな変化はない．そのため，潜在因子数を 15 として実験を進める．

図 6 は提案手法とベースライン手法の再現率の比較である．図は各推薦件数に対する再現率の平均を表した棒グラフであり，提案手法とベースライン手法を比較している．棒グラフの上にある黒字の数字はウィルコクソンの符号順位付検定での p 値になる．赤い星は提案手法での再現率の平均値がベースラインと比較して良く，検定で有意差がある場合に付いている．この図から，推薦件数が 10，20 の場合に提案手法の方がベースライン手法よりも良い結果が得られることがわかる．推薦件数が 10 の場合，提案手法がベースライン手法の約 1.26 倍改善している．

図 7 は提案手法とベースライン手法の適合率の比較である．図は各推薦件数に対する適合率の平均を表した棒グラフである．図の縦軸は適合率であり，その他の部分は図 6 と同様である．この図から，適合率に関しても推薦件数が 10，20 の場合に提案手法の方が良い結果となり，推薦件数 10 の場合に提案手法の方がベースライン手法の 1.25 倍改善している．

図 8 は提案手法とベースライン手法の F 値の比較である．図は各推薦件数に対する F 値の平均を表した棒グラフである．縦軸は F 値であり，その他の部分は上記と同様である．この図からも再現率適合率と共に推薦件数が 10，20 の場合に提案手法の方がベースライン手法よりも良い結果となり，推薦件数 10 の場合に提案手法の方が約 1.26 倍改善している．

交差検定で手法を比較した結果，推薦件数が上位 10，20 件の時に提案手法の推薦精度がベースライン手法の推薦精度よりも改善することがわかった．また，上位 10 件と上位 20 件を比較すると上位 10 件の方が 2 つの手法の有意差がある結果となり，推薦件数が 10 の場合，提案手法が再現率，適合率，F 値を約 1.25 倍改善することがわかった．

3.5　期間で区切ったデータを用いた実験結果

データを期間で区切って行った実験の結果を示す．

図 9 は，GH と SO の活動記録を併用した提案手法の推薦件数と潜在因子数を変化させた F 値の遷移を表している．パラメタを変化させることで F 値がどのように変化するか確認した．この結果，潜在因子数が 2，推薦件数が 10 のときに F 値が最も高くなっている．これ以降，潜在因子数を 2 として，実験を進める．

図 10 は提案手法とベースライン手法の再現率の比較である．図の横軸や縦軸は 3.4 節の図 6 と同様である．この図は提案手法の方がベースライン手法よりも再現率で良い結果が得られることを示している．推薦件数 10 の時に最も提案手法とベースライン手法との差が大きい．そこで推薦件数 10 としてウィルコクソンの符号順位検定を行った．しかし，p 値は 0.4049 となり，有意差は見られなかった．

図 11 は，提案手法とベースライン手法の適合率の比較である．この図の横軸と縦軸は 3.4 節の図 7 と同様である．この図より，提案手法の方がベースライン手法よりも適合率で良い結果を示しており，推薦件数 10 の時に最も提案手法とベースライン手法との差が大きいため，図 10 と同様に推薦件数 10 でウィルコクソンの符号順位検定を行ったところ p 値は 0.5413 となり有意差は見られなかった．

図 12 は，提案手法とベースライン手法の F 値の比較である．図は，横軸と縦軸が 3.4 節の図 8 と同様である．この図でも上記と同様に推薦件数 10 とし，ウィルコクソンの符号順位検定を行ったところ p 値は 0.4049 となり有意差は見られなかった．

これらの結果から，提案手法とベースライン手法の有意差は見られなかった．

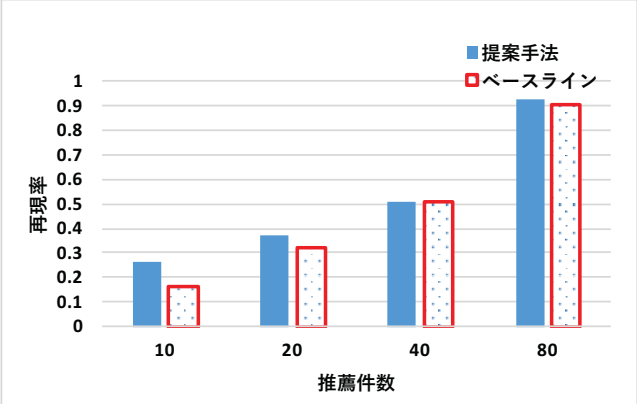

図 10　潜在因子数を 2 とした推薦件数に対する再現率の比較

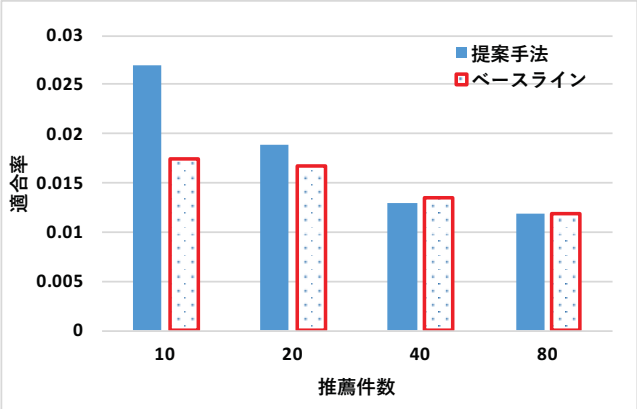

図 11　潜在因子数を 2 とした推薦件数に対する適合率の比較

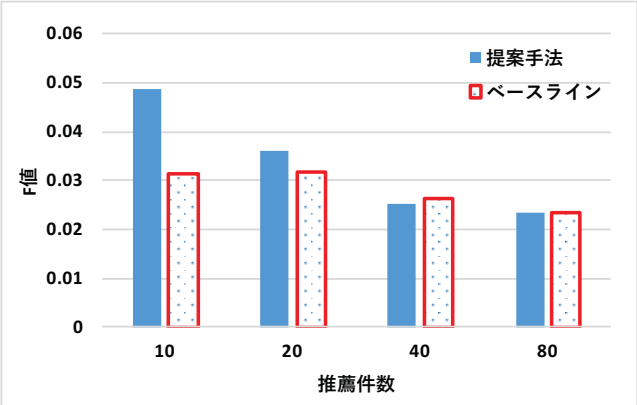

図 12　潜在因子数を 2 とした推薦件数に対する F 値の比較

3.6　考察

交差検定を行った結果では推薦件数が 20 件以下の場合に提案手法による推薦精度改善の有意差が見られた．推薦件数が 10 の場合の方が提案手法とベースライン手法の差が大きく，提案手法の方が再現率，適合率，F 値において約 1.25 倍良い結果となった．この結果から，提案手法ではベースラインよりも推薦するリポジトリの上位に開発者の嗜好に合ったリポジトリを推薦できていると考えられる．

データを期間で区切った場合では提案手法とベースライン手法の有意差は見られなかった．これは，設定した期間で一回しか実行していないためだと考えられる．開発者がコミットするリポジトリは平均 1 ～ 2 個 [9] と少なく，評価データに含まれる開発者と開発者がコミットしたリポジトリのペアが少数なため，評価が十分に行えなかったと考えられる．

4.　関連研究

4.1　単一の開発者向けサービスに関する研究

4.1.1　GH に関する関連研究

GH のネットワーク構造のリンク予測を利用したリポジトリ推薦や開発者の推薦の研究が行われている．Matek ら [9] は，ユーザとリポジトリの 2 部グラフを作成し，さまざまなリンク予測の手法を比較，分析していた．その結果，グラフのノードのまとまりであるコミュニティを決定し，コミュニティのリンク状況からノード同士の類似度を求め，リンクを予測するリポジトリ推薦手法が良いことを明らかにしていた．Surian ら [15] は開発者とプロジェクトの特製，プロジェクトからなる 3 部グラフを作成し，Random Walk with Restart を用いることで開発者の推薦度の計算を行っている．そして，Random Walk with Restart を用いることは開発者の推薦に有効であることを示していた．Palnitkar ら [11] は，ユーザ同士が繋がるグラフに対して，教師付き Random Walk with Restart を用いた手法を提案している．そして，提案手法が教師付きでない Random Walk with Restart と比べ良いことを示している．

協調フィルタリングの手法を用いたリポジトリ推薦の研究も行われていた．Guendouz ら [7] はユーザ同士が共有するリポジトリ数から似ているユーザのグループを求め，そのグループ内の開発者が多くフォークしたリポジトリを推薦するといった協調フィルタリングに基づく手法を提案している．そして，提案手法がリポジトリ推薦に有効であることを示していた．

4.1.2　SO に関する関連研究

SO の開発者の関心のあるトピックを抽出する研究がなされている．Meng ら [10] は，開発者が関心を持つ複数のトピックを質問に付けられるタグから求める手法を提案している．たくさんの質問からタグのグループを分類し，開発者が関心を持つタグのグループを発見している．提案手法を他の手法と比較し，有用であることを示している．

開発者が作り出すコミュニティと開発者の動向に関する研究がなされている．Gimaraes ら [2] はユーザと似ている興味を持つユーザ集合であるコミュニティとの関係を分析し，コミュニティ発展をモデル化する感染症モデルを拡張した CERIS モデルという手法を提案している．コミュニティ間の関係性が大きいほどユーザの動向に影響を与えるということを示している．

4.2 複数の開発者向けサービスを組み合わせた研究

4.2.1 GH と SO の両方を用いた関連研究

GH と SO での同一の開発者の活動の関連性の研究がなされている．Vasilescu ら [16] は，GH と SO のどちらでも活動している開発者を抽出し，開発者の行動を観察することで，開発者の GH でのコミットが SO で質問や回答にどう影響しているのかを調査している．

GH と他のサービスでの同一の開発者を観察し，各サービスの特徴を分析する研究がなされている．Silvestri ら [14] は SO と GH,Twitter の 3 つを合わせた分析を行っている．この 3 つの SNS で活動しているユーザを見つけ出す方法を提案している．各 SNS のユーザのリンク構造の違いから 3 つの SNS の違いを分析し，各 SNS の特徴を分析している．

4.3 本研究と関連研究の違い

本研究は，関連研究と異なり，GH と SO を組み合わせたリポジトリ推薦を提案している．関連研究では，GH のデータのみを利用したリポジトリ推薦や GH と他のサービスを組み合わせた分析などが行われていた．本研究では，GH と SO のデータを組み合わせることで，リポジトリ推薦の向上を図っている．

5. 結論と今後の課題

本研究では，GH と SO の活動記録を併用したリポジトリ推薦手法を提案した．提案手法は，GH と SO にまたがる開発者を推定し，GH と SO の活動記録を組み合わせる．そして組み合わせた活動記録を行列分解に基づく手法に適用し，リポジトリを推薦する．

提案手法と GH の活動記録のみを用いた手法を比較した結果，リポジトリ推薦精度に対して提案手法で有意差が見られた．交差検定を行った場合に上位 20 件以下で推薦したリポジトリに対して提案手法の方が精度が良くなることを確認できた．提案手法はベースライン手法に比べ，上位に開発者の嗜好に適したリポジトリを推薦することを示した．

今後の課題として，GH と SO を組み合わせることでリポジトリの推薦精度をさらに上げるために，手法や推薦に用いるデータの内容の検討が必要である．本研究では，GH のコミットに関する情報と SO の投稿に関する情報のみ利用していたが，得られたデータには開発者の関心の特徴を表す他の情報も含まれていた．GH の Watch，Stars，Issue comment や SO の各質問に付けられている評価値などを利用する手法を検討したい．加えて，有効なデータの数を増やす為に同一ユーザの推定により洗練された手法（[18] など）を適用したい．また，関連研究のグラフのリンク予測を用いた推薦手法や協調フィルタリングの他の手法などを用いて推薦精度が向上するか調査していきたい．

参考文献

[1] GitHub と Stack Overflow におけるユーザ行動の統一的な分析. In 情報処理学会第 79 回全国大会, 2017.

[2] A. P. C. d. S. Anna Guimarae and and J. M. Almeida. On the dynamics of topic-based communities in online knowledge-sharing networks. In *2015 Second European Network Intelligence Conference*, pages 1–8, 2015.

[3] A. Bacchelli. Mining challenge 2013: Stack overflow. In *MSR 2013*, page to appear, 2013.

[4] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *WWW 2017*, pages 271–280, 2007.

[5] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM 2005*, pages 625–628, 2005.

[6] G. Gousios. The ghtorrent dataset and tool suite. In *MSR 2013*, pages 233–236, 2013.

[7] M. Guendouz, A. Amine, and R. M. Hamou. Recommending relevant github repositories: a collaborative-filtering approach. In *Proc. 2nd International Conference on Networking and Advanceed Systems*, pages 34–37, 2015.

[8] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[9] T. Matek and S. T. Zebec. Github open source project recommendation system. *CoRR*, abs/1602.02594, 2016.

[10] Z. Meng, F. Gandon, C. F. Zucker, and G. Song. Empirical study on overlapping community detection in question and answer sites. In *ASONAM 2014*, pages 344–348, 2014.

[11] A. Palnitkar and V. Khanna. Final project report link recommendation in collaborative coding platforms like github using biased random walks group 35. 2013.

[12] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM 2008*, pages 502–511, 2008.

[13] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. 2010.

[14] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli. Linking accounts across social networks: the case of stackoverflow, github and twitter. In *Proc. the 1st International Workshop on Knowledge Discovery on the WEB*, pages 41–52, 2015.

[15] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos. Recommending people in developers' collaboration network. In *Proc. 18th Working Conference on Reverse Engineering*, pages 379–388, 2011.

[16] B. Vasilescu, V. Filkov, and A. Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *Proc. the 2013 ASE/IEEE International Conference on Social Computing*, 2013.

[17] J. H. Ward. Hierarchical grouping to optimize an objective function. In *J.Am.Stat.Assoc.,Vol.58,1963*, pages 236–244, 1963.

[18] T. Komamizu, Y. Hayase, T. Amagasa, H. Kitagawa. Exploring Identical Users on GitHub and Stack Overflow. In *SEKE 2017*, pages 584–589, 2017