卒　業　研　究　報　告　書

平成23年度

研究題目

OSSプロジェクトのコミュニケーション支援を目的とした情報推薦システム

指導教員　　上　野　秀　剛　　助教

氏　　名　　河　居　　寛　樹

平成24年2月29日　提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

OSSプロジェクトの
コミュニケーション支援を目的とした情報推薦システム

上野研究室　　河居　寛樹

Firefox や Linux に代表されるオープンソースソフトウェアの発展や，外部組織に開発作業の一部を委託する外注の増加に伴い，複数の国や地域に点在した開発者による分散開発が増えている．複数人でのソフトウェア開発においては，開発者間のコミュニケーションが重要とされているが，分散開発では開発者が世界中にいるためコミュニケーションを取るのが難しいとされている．開発者間の情報共有にはメーリングリストやバグ管理システム，ソースコードのバージョン管理システムといった開発支援システムが用いられる．複数の開発支援システムを併用している場合，1つのバグを修正するためには複数の情報源を見る必要があり，収集に時間や手間がかかる，情報を見落とす可能性があるなどの問題があり，結果として開発者間の円滑なコミュニケーションの妨げになっている可能性がある．Tani らの先行研究では，既に利用している開発支援システムを変更することなく，複数のシステムに記録された一連の脈絡から関連した情報を予測し推薦するシステムを提案している．しかし，Tani らの研究ではシステムの作成が目的だったため，推薦する関連情報の精度が低いという問題があり，改善の必要がある．そこで，本研究では先行研究で作成されたシステムを改良し，開発者に提示する関連情報の推薦精度を向上することを目的とする．推薦精度を向上させるために，形態素解析，TF-IDF 法，クラスタリングを用いて推薦を行う手法を提案する．提案手法では，開発支援システムから取得した文書データ群に対して形態素解析を行う．文書データ群から形態素解析によりすべての単語を抽出し単語リストを作成する．各文書と文書データ群の単語の出現回数を求め，各文書の各単語ごとに TF-IDF 値を求め，文書に含まれている単語の TF-IDF 値を示した表を作成する．この表の TF-IDF 値を元に各種クラスタリングを行い，類似度を元に 20 個のクラスタに分割する．分割された後の同一クラスタに分類された文書が関連した情報として推薦される．提案手法を実際の開発プロジェクトのデータベースに対して適用する実験を行い，その結果から提案手法の有用性を評価する．提案手法の有用性を評価するために行った実験では，オープンソースソフトウェアの開発プロジェクトで記録された文書データベースに対して提案手法を適用し，その結果から推薦精度を求めた．実験，評価の結果，先行研究で作成されたシステムに比べ，開発者に提示する関連情報の推薦精度が向上させることができた．推薦精度が向上したことにより，開発者が開発に充てられる時間が増えることになり開発が進み，開発コンテキストを理解するのが容易になり開発者の新規参入のハードルが下がり OSS 開発全体が活発化し，必要な情報を短時間で探すことができ，バグの特定や修正にかかる時間が短くなると考えられる．

目 次

1 はじめに

Firefox や Linux に代表されるオープンソースソフトウェア (Open Source Software:OSS) の発展や，外部組織に開発作業の一部を委託する外注の増加に伴い，複数の国や地域に点在した開発者による分散開発が増えている．複数人によるソフトウェア開発では，開発者間のコミュニケーションが重要とされているが，分散開発ではコミュニケーションを取るのが難しい [1].

分散開発において開発者間でコミュニケーションを取るために，開発者が直接顔を合わせて行うオフラインミーティングや電話会議が用いられる．しかし，これらの手段は多大な移動時間・費用が必要なことに加え，会議参加者全員のスケジュールを調整し，同じ時間を共有する必要がある．そのため，開発者間で時差が発生するオフショア開発[1]においてはオンラインによる非同期な情報交換がよく用いられている [2][3].

ソフトウェア開発における代表的なオンライン非同期コミュニケーションツールとして，メーリングリスト (Mailing List:ML) やバグ管理システム (Bug Tracking System:BTS)，ソースコードのバージョン管理システム (Version Control System:VCS) がある．本論文では以降，これらのシステムを開発支援システムと呼ぶ.

複数の開発支援システムを併用している場合，あるバグが発生したとき，そのバグについて BTS で症状や再現手順，重要度などについて報告し，修正方法や修正を担当する開発者の決定について ML で議論を行う．その後，バグの修正が行われたら，修正後のソースコードを VCS に登録しバグの修正プロセスが完了する.

バグ修正以外の，機能追加やリファクタリング[2]などにおいても同様の手順で履歴の記録，および議論が行われる．テスターが発見したバグの修正を担当する開発者は，どのような症状のバグなのかを調べる作業を行う．その際，BTS に記録された症状の説明や，ML で行われた議論を参考にバグの症状を理解し，VCS に記録されたソースコードの変更履歴を元にバグの発生箇所を特定する．また，過去に同様のバグが発生していた場合，過去の修正方法が新しいバグの修正の助けになることがあるため，BTS で同様のバグが過去に発生していたかどうかを調べることがある.

このように 1 つのバグを修正するためには複数の情報源を見る必要がある．そのため，収集に時間や手間がかかる，必要な情報を見落とす可能性があるなどの問題がある．必要な情報を見落とした場合，開発者間でのコミュニケーションに齟齬が発生し，円滑なコミュニケーションの妨げになる可能性がある.

Tani らの先行研究では，複数のシステムに横断して記録された情報の閲覧を支援することを目的に，複数のシステムに記録された一連の脈絡から関連した情報を予測し推薦するシステムを提案している [4]．しかし，Tani らの研究ではシステムの作成が目的だったため，関連情報の推薦に簡易な方法を用いており，精度が余り高くない.

[1] システム開発・運用管理などを海外の事業者や海外子会社に委託すること.

[2] 設計変更やバグの修正などによりプログラムが冗長で汚くなった場合に，将来の仕様変更に柔軟に対応できるようソースコードの手直しを行うこと.

本研究では先行研究で作成されたシステムを改良し，開発者に提示する関連情報の推薦精度を向上することを目的とする．

以下，2 章ではソフトウェア開発に用いられる開発支援システムの概要と問題点，先行研究について説明する．3 章では提案手法について説明する．4 章では提案手法を用いて実験を行い，その結果を評価し考察する．5 章では先行研究との比較を行い本研究の位置づけを明らかにする．最後に 6 章ではまとめと今後の課題を述べる．

2　開発支援システム

2.1　概要

開発支援システムとは，開発に関する様々な情報を共有するためのシステムである．具体的なシステムにはメーリングリストやバグ管理システムがあり，開発に関する議論や情報共有，バグや不具合の検出から除去までの履歴などの情報を共有するために利用される．開発支援システムの多くはサーバ上で CGI やサーバソフトウェアとして動作し，Web ブラウザなどから利用される．以下で主要な開発支援システムについて説明する．

- メーリングリスト：ML
  開発者間のメールのやり取りを容易にし，メーリングリストに流れたメールの保管，閲覧機能を提供する．

- バグ管理システム：BTS
  発生したバグの報告・登録を行い，バグが起こった状況の再現やバグの修正方法，修正履歴といったバグの修正までの流れを管理する．

- バージョン管理システム：VCS
  ソフトウェア開発における様々な成果物を保存，プロジェクト内のファイルの共有や管理の支援，変更履歴の保存をする．

開発支援システムの利用者は，これらのシステムをそれぞれ参照し，開発やサポートに必要な情報を収集することができる．利便性向上のために，複数の開発支援システムを統合したシステムも存在する [5][6].

ソフトウェア開発を行なっている企業では．独自に開発支援システムを開発し，社内のネットワークで運用・管理が行われている．一方で，小規模な OSS プロジェクトでは開発のための予算を持たず，サーバの用意・運用にかかる費用を負担できないことも多い．また，OSS プロジェクトに参加する開発者はボランティアであることが一般的であり，参加者は流動的に変化するため，サーバを適切に管理できる人員を安定的に確保できないことがある．そのため，googlecode[7] や Source Forge[8], Lanchpad[9] のような無償のレンタルサービスが利用されている．OSS 開発プロジェクトではこれらの開発支援システムを組み合わせて利用することが多い．1 つのプロジェクトで複数の開発支援システムを併用している場合，バグを発見すると BTS に報告を行い，ML で修正方法を議論し，ソースコードの変更を VCS で記録するといった使い方をする．このように 1 つのバグを修正するためには複数の開発支援システムにアクセスする必要がある．

2.2 問題点

前節で述べた個別の開発支援システムや統合されたシステムを用いることで，開発者間の情報共有が容易になるが，これらのシステムには 1) 複数の開発支援システムを併用した際の問題，2) 統合システムを用いる際の問題が存在する．本節ではこれらの問題について説明し，本研究の意義を明確にする．

2.2.1 複数の開発支援システムの併用による問題

複数の開発支援システムを用いて，1 つのプロジェクトに関する情報を管理している場合，開発者はある 1 つの事柄を調査するために，手動で複数のシステムから情報を収集する必要がある．このとき，調査している事柄（例えば，ある不具合がどのように議論・修正されたか）を示す開発における一連の流れ（開発コンテキスト）を理解していなければならない．

図 1 に複数の開発支援システムを用いた開発におけるコンテキストの様子を示す．3 つの長方形は VCS・BTS・ML の各開発支援システム，楕円はそれぞれのシステムに保存された情報，楕円を繋ぐ点線は同一コンテキストに属する情報のつながりを表している．それぞれのコンテキストに含まれる情報は，その種類ごとに異なるシステムに分散して保存されている．ある情報に関係する情報を探したいとき，点線で示されるコンテキストを元に情報をたどることになるが，コンテキストは開発支援システムに存在せず，開発者の知識に頼っているのが現状である．

異なるシステムに分散したある情報に関係する情報を検索するような状況において，開発者が調査を行う際には以下のような問題点がある．

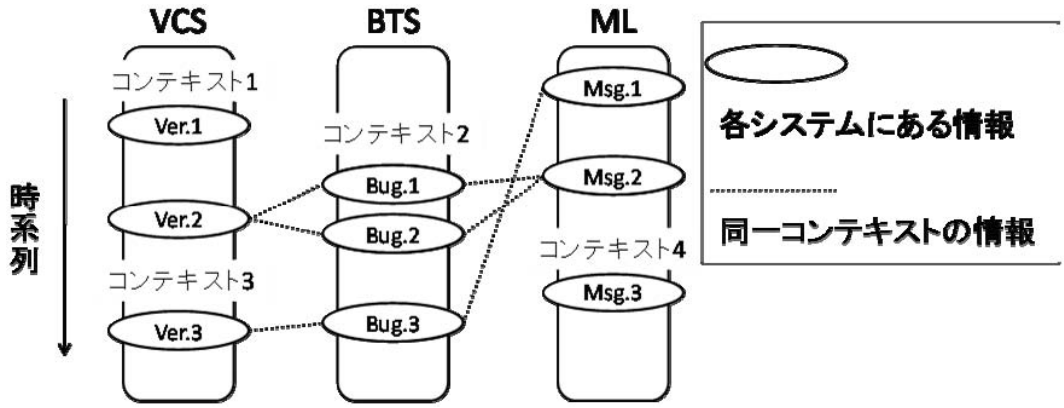

図 1: 開発におけるコンテキストの様子

問題 1：複数のシステムにまたがった検索が必要

複数の開発支援システムを併用している場合，1 つの開発コンテキストに関する情報を収集するために，開発者が手動で複数のシステムを調査する必要がある．このとき，各システムが持つ，同じ開発コンテキストに関する情報は互いにリンクされておらず，必要な情報を見落としてしまう可能性がある．BTS や VCS に記録される不具合情報・履歴情報には，内容の理解や検索に用いるためのコメントや不具合 ID などが入力されるが，システム間が統合されていない場合，それらを基に手動で検索を行う必要がある．

問題 2：開発コンテキストの不足

複数の開発支援システムから，必要な一連の情報を探すためには，開発時のコンテキストを理解していなければならない．例えば BTS に記録されたある不具合の情報を収集する時，不具合についてどのような議論や修正が行われてきたのかを知らなければ，その開発プロジェクトでの開発支援システムの使われ方（BTS に登録する情報は要望やソフトウェアテストの結果報告なども含めるのか，ML で取り扱う話題はどうするのか，個別のバグに関する話題も含めるのかなど）を理解できていないために，VCS や ML から効率よく情報収集することが難しい．このような開発コンテキストは開発支援システムには保存されておらず，新規にプロジェクトに参加した開発者はコンテキストの把握ができない．また，以前から参加している開発者であっても，過去のコンテキストを完全に覚えているわけではなく，記憶に頼っているのが現状である．

2.2.2 統合システムを用いる際の問題

複数のシステムの併用による問題を解消するために，複数のシステムを統合したシステムが提案・実装されている [5][6]．これらの統合システムでは BTS や VCS，ML の他に Wiki やテスト管理システムなどを統合することで，プロジェクトの管理に必要な情報を 1 つのシステム内で管理できる．従来の統合システムには，既存の開発支援システムを拡張したものと，統合システムとしてあらかじめ用意されたシステムを利用するものがある．しかし，これらの統合システムをプロジェクトに導入するためには以下に示す問題 3，4，5 がある．

問題 3：既存開発支援システムのデータ利用が困難

プロジェクトが既に開発支援システムを導入している場合，これまでに開発支援システムに蓄積された情報の新規に導入する統合システムでの使用が難しい．一部の統合システムでは他の開発支援システムからのデータインポートをサポートしているが，全ての情報が適切に変換されるわけではない．

問題 4：外部システムを統合できない

既に運用している開発支援システムが存在する場合，複数のシステムを接続することで蓄積された情報をそのまま使用しながら統合システムとして動作させることができる．しかし，レンタルサービスのような外部で提供された開発支援システムを用いている場合，システムの変更ができないため，統合することができない．

問題 5：使用できる開発支援システムに制限がある

個別の開発支援システムを用いる場合，それぞれの情報の種類に応じて，プロジェクトで必要な機能を持ったシステムを選択する事ができる．しかし，統合システムを用いる場合，各統合システムが提供する機能しか利用できず，個別の開発支援システムを組み合わせた場合に比べて制限がある．

2.3 先行研究

問題点に対応するために，既に利用している外部の開発支援システムを変更することなく，開発情報を統合して表示する研究がある [4]．このシステムでは，開発者の開発支援システムに対するアクセスを監視し，システムに保存されている開発情報を取得し，テキスト解析による類似情報の推定と提示を行う．

システムが収集する情報は，BTS では投稿ごとに振られる固有番号，投稿者名，投稿時刻，バグの症状や再現手順の説明文など，VCS ではリビジョン番号[3]，変更者名，変更内容の説明文，ファイル名などである．収集した情報は，システム内部のキャッシュデータベースに保存される．情報収集のあと，収集した情報に対して同一コンテキスト情報の推定が行われる．収集した情報はキャッシュデータベースに保存された他システムの情報と関連性が推定される．

推定処理の結果はそれぞれの開発支援システムの参照結果の中に組み入れ，開発支援システムを閲覧しようとした開発者に提示される．先行研究のシステムによる出力例を図 2 に示す．

問題 3，4 については既存のシステムが予め持っているインタフェースを介して通信を行いシステムを変更することなく蓄積された情報をそのまま利用することで解決している．大抵の開発支援システムでは HTTP プロトコルを用いており，先行研究では HTTP などのインターフェースに対応している．プロジェクトで利用したい開発支援システムと提案システムの結合を容易にすることで，問題 5 を解決している．これらの 2.2.2 節の問題については解決している．

[3] 一度の編集ごとに振られる番号

2.4 既存システムの問題点

先行研究のシステムを用いて，単体で動作している開発支援システムにアクセスすると，自動的に他の開発支援システムに保存されている関連する情報を抽出し提示する．これにより問題 1 に対応している．この時複数のシステムに保存された情報間の関連性を推定し，開発コンテキストを知らない開発者に一連の情報を示す事で，問題 2 に対応している．これらの 2.2.1 節の問題については対応しているが，先行研究の段階では問題 1 の関連する情報を提示する精度が低く，問題 2 の関連性の推定には簡易な方法を用いており，開発コンテキストを知っている開発者が選定した単語 61 個を推定に用いている．そのため，開発者が選定した 61 個の単語を含まない情報を必要としている場合，その情報を推薦できないという問題があった．そこで，本研究では複数の開発支援システムの併用による問題を解決する．問題 1，2 を解決するために開発者に一連の情報を推薦する手法を提案する．

システム間で登録されている情報のリンクを管理するという手法ではなく，一連の情報を推薦するというアプローチを選んだ理由は以下の 3 点である．

１つ目は，システム間で登録されている情報のリンク管理を行う場合，人の手で管理を行う必要があり，開発者が開発以外の作業に時間を取られることになり，開発の遅れにつながるからである．

２つ目は，人の手で管理を行うことができた場合，管理を行う者が全ての開発コンテキストを把握しておく必要があり，開発者が入れ替わる OSS 開発プロジェクト

図 2: システムによる出力例

では全てのコンテキストを把握することが困難になる．

3つ目は，長期に渡って開発を行なっているOSS開発プロジェクトが蓄積する情報は膨大なものとなり，たとえリンクが管理されていたとしても，リンクを辿って必要な情報を探すには時間がかかり，見落としが発生しないとはいえないからである．

推薦というアプローチを取ることで，1つ目の理由により開発者が開発に充てられる時間が増えることになり開発が進み，2つ目の理由により開発コンテキストを理解するのが容易になり開発者の新規参入のハードルが下がりOSS開発全体が活発化し，3つ目の理由により必要な情報を短時間で探すことができ，バグの特定や修正にかかる時間が短くなる．

3　提案手法

先行研究のシステムの推定精度を高めるための手法を提案する．開発支援システムから取得した文書データに対して形態素解析を行い単語を抽出し，各単語の特徴ベクトルを TF-IDF 法で求める．特徴ベクトルの類似度から文書データをクラスタリングし，同一クラスタの文書を関連情報として扱い推薦する．

3.1　日本語の形態素解析

形態素解析とは，自然言語で書かれた文書を言語の中で意味のある最小単位に分割し，辞書を用いて品詞を特定する手法である．TF-IDF 法を用いるためには単語の出現回数をカウントする必要があり，そのためには文書から単語を抽出する必要があるため，形態素解析を用いる．

本研究では，オープンソース形態素解析エンジンの MeCab[10] を用いて形態素解析を行う．例として「吾輩は猫である。名前はまだない。」という文を形態素解析した結果を図 3 に示す．

3.2　TF-IDF 法

TF-IDF 法とは，ある文書の集合（文書セット）の中に含まれる一つの文書に注目したとき，その文書が文書セットの中でどういった単語 (term) で特徴づけられるかを調べる手法である [11]．TF-IDF は，ある単語が注目している文書から出現した回数を，文書セット中に出現した回数で割って求めた出現頻度を表す TF(Term Frequency) と，その単語が文書セット中のいくつの文書に含まれているかを表す

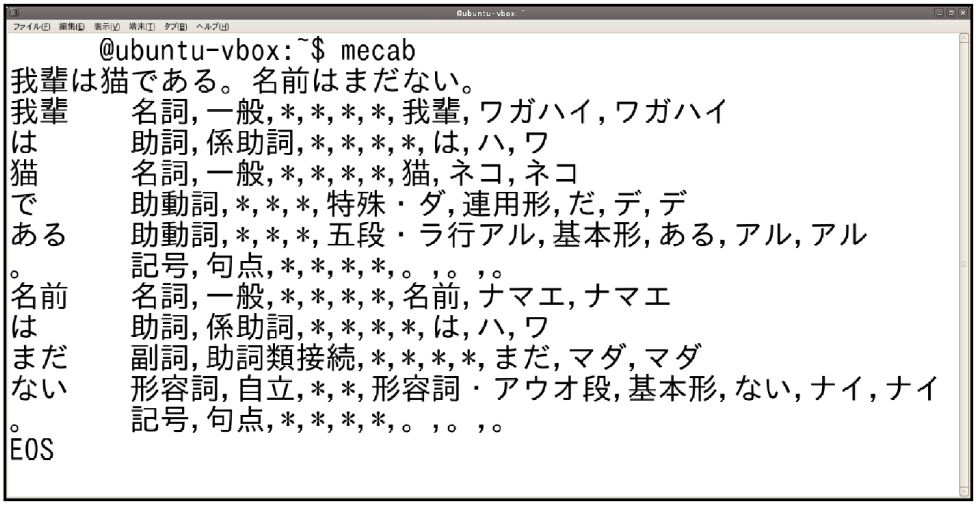

図 3: 日本語形態素解析の実行例

DF(Document Frequency) の逆数の対数をとった IDF(Inversed DF) のふたつの指標に基づいて計算される．TF 値が高ければある文書中にその単語が多く出現していることを表し，IDF 値が高ければ特定の文書にしか出現しないことを表す．式 (3) により単語ごとの特徴度を求める．

$$tf_{c,n} = \frac{d_{c,n}}{\sum_{i=1}^{C} d_{i,n}} \tag{1}$$

$$idf_c = log_e \frac{|D|}{|\{d : t_c \in d\}|} \tag{2}$$

$$TF - IDF_{c,n} = tf_{c,n} \cdot idf_c \tag{3}$$

$d_{c,n}$ は単語 c の文書 n における出現回数，$|D|$ は全文書数，$|\{d : t_c \in d\}|$：単語 c を含むドキュメント数を表す．

式 (3) で求められる TF-IDF 値は文書におけるその単語の特徴度の高さを表している．

本研究では，この TF-IDF 法を用いて BTS，VCS の複数の文書間の特徴度を測定する．

形態素解析で抽出した単語のうち，全文書中 1 回しか出現しない単語はその単語を含む文書以外に類似する文書が存在しないため，推薦できない．推薦するためには最低でも 2 回出現する必要があるため，全体で 2 回以上出現した単語を用いて TF-IDF 値を求める．

3.3 クラスタリング

与えられた文書から類似した文書を外的基準なしにクラスタと呼ばれるグループに分類する，クラスタリングという手法を用いる．クラスタリングの手順は以下の通りである．

1. 与えられた文書のひとつをクラスタとする（文書の数だけクラスタが生成される）．

2. 各クラスタ間の類似度（距離）を求める．

3. 最も類似しているクラスタをひとつのクラスタとしてまとめる．

4. クラスタがひとつになるまで 2. 3. の処理を繰り返す．

同一のクラスタに分類されると共通の特徴を持つことを意味し，別のクラスタに分類されたものは共通の特徴を持たないことを意味する．類似度を求める指標として TF-IDF 法によって求めた値を元にクラスタリングを行う．

手順 2 のクラスタ間距離を求める手法として，完全連結法，単連結法，群平均法，重心法，ウォード法，メディアン法がある．

3.3.1 完全連結法

完全連結法は，2 つのクラスタの中からそれぞれひとつずつ文書を選び，文書間の距離を求め，それらの中で最も距離の短い文書間の距離を 2 つのクラスタ間の距離とする方法である．

3.3.2 単連結法

単連結法は，2 つのクラスタの中からそれぞれひとつずつ文書を選び，文書間の距離を求め，それらの中で最も距離の遠い文書間の距離を 2 つのクラスタ間の距離とする方法である．

3.3.3 群平均法

群平均法は，完全連結法と単連結法を折衷した方法で，2 つのクラスタの中からそれぞれひとつずつ文書を選び，文書間の距離を求め，それらの距離の平均値を 2 つのクラスタ間の距離とする方法である．

3.3.4 重心法

重心法は，2 つのクラスタのそれぞれの重心を求め，その重心間の距離をクラスタ間の距離とする方法である．重心を求める際にはクラスタに含まれる文書数が反映されるように文書数を重みとして用いる．

3.3.5 ウォード法

ウォード法は，2 つのクラスタをまとめた際にクラスタ内の分散とクラスタ間の分散の比が最大化するようにクラスタを形成していく方法である．

3.3.6 メディアン法

メディアン法は重心法の変形で，2 つのクラスタの重心間の重み付きの距離を求める際，重みを等しくして求めた距離の値を 2 つのクラスタ間の距離とする方法である．

3.4　提案手法の手順

提案手法の手順の流れを図 4 に示す．図の下部の 3 つの四角は処理を表し，矢印はデータの流れを表し，上部は処理ごとのデータの変化を表している．重みづけの点は TF-IDF 法により重み付けされた文書を表し，クラスタリングによりクラスタ分けされている．開発支援システムから取得した文書データ群に対して形態素解析を行う．文書データ群から形態素解析によりすべての単語を抽出し単語リストを作成する．各文書と文書データ群の単語の出現回数を求め，各文書の各単語ごとに TF-IDF 値を求める．文書を縦方向に，形態素解析により抽出した単語を横方向に羅列し，文書に含まれている単語の TF-IDF 値を示した表を作成する．この表の TF-IDF 値を元に各種クラスタリングを行い，類似度を元に 20 個のクラスタに分割する．分割された後の同一クラスタに分類された文書が関連した情報として推薦される．

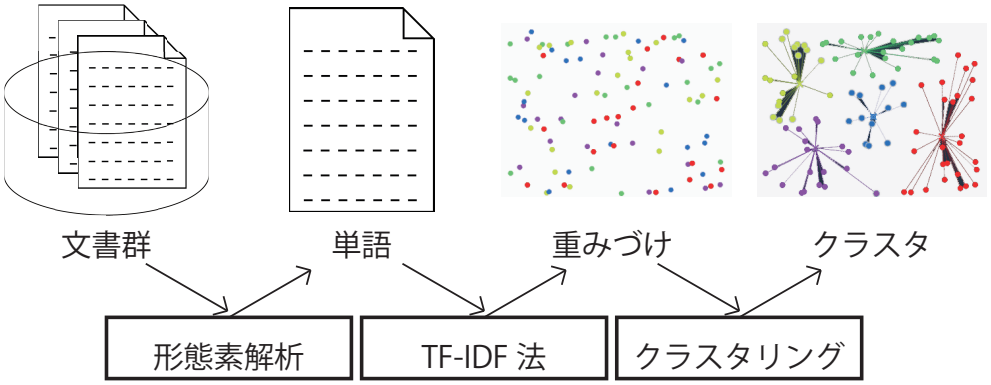

図 4: 提案手法のフロー

4 実験

4.1 実験方法

提案手法の有用性を確認するための評価実験を行った．実験では，オープンソースソフトウェアの開発プロジェクトで記録された文書データに対して提案手法を適用し，推薦精度を算出する．クラスタリングのクラスタ間距離を求める方法として完全連結法，単連結法，群平均法，重心法，ウォード法，メディアン法の 6 種類を用いる．

4.2 評価方法

本研究の目的は推薦精度の向上である．そのため提案手法による推薦の精度が向上したかどうかを評価する必要がある．評価には，システムが関連があるとして推薦したものと，開発者が実際に関連があるとして手動で設定しているリンクを比較し，求めた精度を用いる．

精度を求めるために，Precision と Recall を求めて評価を行う．システムが推薦した解答のうち何割が正しい答えかをあらわす値 Precision は (4) 式によって求められる．Precision は 0〜1 の範囲をとり，値が高いほどシステムが開発者の意見に反する結果を推薦していないことを示す.

開発者が有用と判断した文書のうち何割をシステムが推薦できているかをあらわす値 Recall は (5) 式によって求められる．Recall は 0〜1 の範囲をとり，値が高いほどシステムは開発者が有用と判断した文書を推薦していることを示す.

Precision のみを向上させようとすると，正しい答えである可能性が低い文書を推薦しなければいいが，答えである可能性が低いと判断された文書に答えが含まれている場合，推薦漏れが発生し，Recall が低下する．Recall のみを向上させようとすると，正しい答えである可能性が少しでもある文書を推薦すればいいが，答えでない文書が増え，ノイズとなり Precision が低下する．このように，Precision と Recall はトレードオフの関係にあるため，仮に Precision が良い値でも Recall が極端に低い場合や，Recall が良い値でも Precision が極端に低い場合などが考えられる．このため，システムの推薦性能について評価する指標として F1-value を用いる．F1-value は (6) 式によって求められる．

$$Precision = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{4}$$

$$Recall = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{5}$$

$$F1 - value = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{6}$$

true positive ：システムが推薦すべき文書を推薦した数
false negative：システムが推薦すべき文書を推薦しなかった数
false positive：システムが推薦すべきでない文書を誤って推薦した数
true negative ：システムが推薦すべきでない文書を推薦しなかった数

4.3 対象プロジェクト

本研究では，日本語で記述可能なプログラミング言語「なでしこ」の開発プロジェクトを対象に実験分析を行う．開発プロジェクトで用いられている 2008 年 10 月 12 日から 2010 年 9 月 28 日までのバージョン管理システムのデータ 235 件と，2008 年 8 月 22 日から 2011 年 9 月 22 日までのバグ管理システムのデータ 1842 件を実験に用いる．VCS のデータの例を図 5 に示す．

4.4 結果と考察

VCS では 235（縦）× 266（横）の行列ができ，BTS では，1842（縦）× 2493（横）の行列ができた．この行列に 6 種類のクラスタリングを適用し，Precision, Recall,

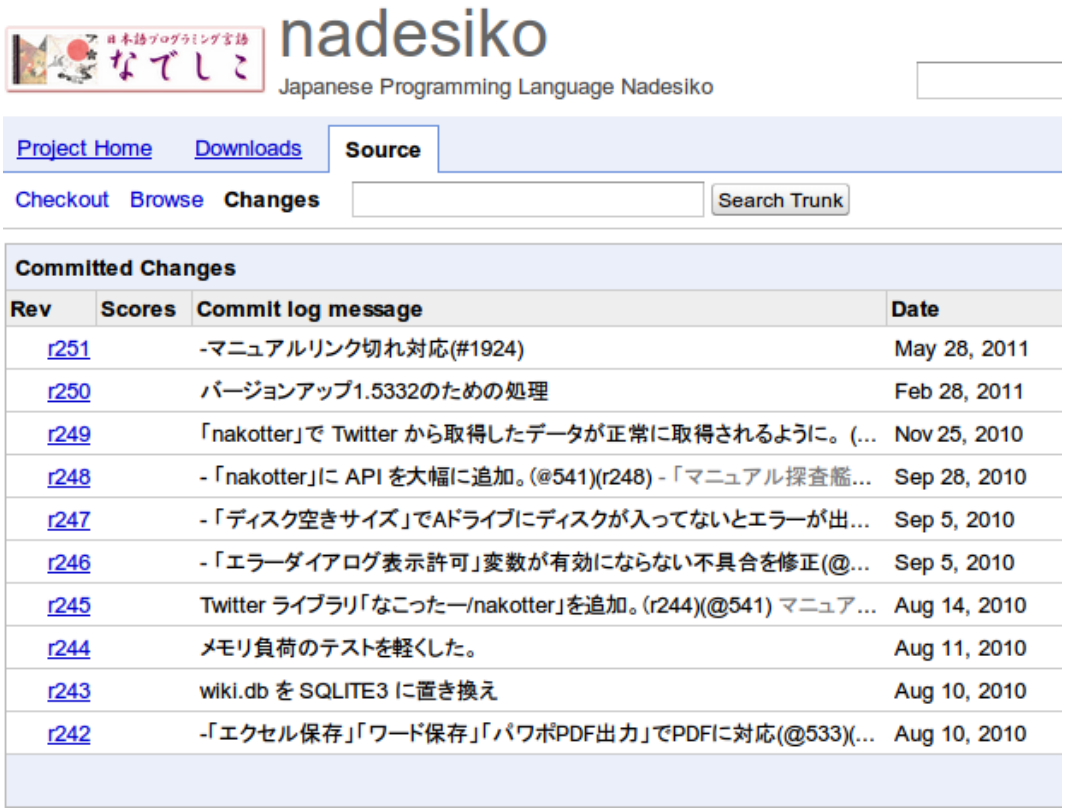

図 5: VCS のデータの例

表 1: VCS と BTS に対する推薦精度

	VCS（235 件）			BTS（1842 件）		
	Precision	Recall	F1-value	Precision	Recall	F1-value
完全連結法	0.090	0.798	0.113	0.035	0.999	0.065
単連結法	0.074	0.833	0.061	0.072	0.986	0.018
群平均法	0.073	0.821	0.105	0.011	0.996	0.022
重心法	0.091	0.798	0.095	0.035	0.996	0.065
ウォード法	0.089	0.845	0.134	0.035	0.999	0.066
メディアン法	0.095	0.798	0.104	0.049	0.989	0.061
平均	0.086	0.816	0.102	0.040	0.994	0.050

F1-value を求めた結果を表 1 に示す.

VCS の Recall に注目すると，平均で 0.816 と高い数値が出ているが，Precision に注目すると，平均で 0.086 と低い数値になっている．このため，F1-value が平均で 0.102 という数値になっている.

BTS の Recall に注目すると，平均で 0.994 と高い数値が出ているが，Precision に注目すると，平均で 0.040 と低い数値になっている．このため，F1-value が平均で 0.0496 という数値になっている.

つまり，Recall の値は高いが，Precision の値が低くなっており，推薦される文書の数が多く，その中に少数の答えが含まれていることを表している．4.2 節で述べたように，Precision と Recall はトレードオフの関係にあるため，どちらかの値が低ければ F1-value の値も低くなる．提案手法ではいずれのクラスタリング手法を用いた場合においても Precision が低く，結果として F1-value が低くなったと考えられる.

今回用いた VCS のデータは 235 件の文書を含み，それを 20 個のクラスタに分割しているため，1 つのクラスタあたり平均 11.75 件の文書が存在することになる．VCS の開発者が作成したリンクは，1 つの文書に 1 つのリンクしかないパターンが大半を占めているため true positive = 1 となる．これより，Precision = 1/11.75 = 0.085106 となるパターンが多くなり，Precision の値が小さくなっていると考えられる.

BTS のデータでも同様に 1842 件の文書を 20 個のクラスタに分割しているため，1 つのクラスタあたり平均 92.1 件の文書が存在し，Precision を求める分母が大きくなるため，Precision の値が小さくなっていると考えられる.

VCS と BTS の 1 つのクラスタが含む文書の最小値と最大値を表 2 に示す．表 2 より，VCS のようにデータ数がそれほど多くない場合には各クラスタリング手法の差は見られないが，BTS のようにデータ数が多くなるとクラスタリング手法によって差が見られた.

6 手法の中で最も特徴的な結果が出た単連結法に注目する．BTS の単連結法の場合，最も多くの文書を含むクラスタには 1818 件の文書が含まれている．これは，1 つの文書を見た際に，1817 件の文書が推薦されることを表しており，VCS の単連結

表 2: VCS と BTS の 1 つのクラスタが含む文書の最小値と最大値

	VCS		BTS	
	最小値	最大値	最小値	最大値
完全連結法	2	39	6	142
単連結法	1	70	1	1818
群平均法	4	39	6	145
重心法	1	39	6	145
ウォード法	4	39	56	152
メディアン法	1	39	1	196

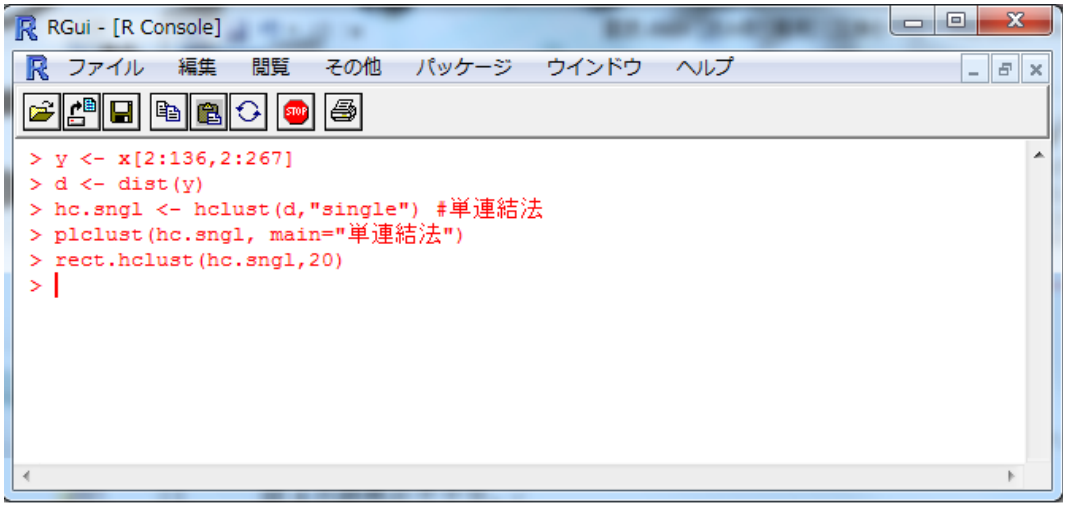

図 6: 木を出力するのに用いた R コマンドライン

法の場合も最も多くの文書を含むクラスタには 70 件の文書が含まれており，すべての文書を読むことは現実的ではないと言える．

VCS のデータに対してクラスタリングを行った際に木の形で表示することができる木の作図には統計解析向けプログラミング言語の R を用いた．用いたコマンドを図 6 に，生成される木を図 7 に示す．図 6 の一行目のコマンドで読み取るデータの範囲を求め，二行目では一行目で読み取ったデータの距離行列を求めている．三行目で単連結法を用いたクラスタリングを行い，四行目でクラスタリング結果から求まる木を描画し，五行目で 20 個に分割したことがわかりやすいように赤い枠を描画している．

図 7 の左側，木の葉に並んでいる数字は文書（クラスタ）を表し，黒い縦線は複数のクラスタを一つのクラスタにまとめていることを表す．黒い横線はクラスタ間の距離を表し，木を根から 20 個に分割したものが赤い線であり，ほとんどが同じクラスタに分類されていることがわかる．このように，少数の文書しか含まないクラスタと，多数の文書を含むクラスタが出来てしまい，推薦される文書の数が多くな

るため，なでしこの開発プロジェクトと単連結法は相性が悪いと考えられる．

F1-value の値を向上させる，つまり Precision の値を向上させる手段として，分割するクラスタの数を増やすことが挙げられる．しかし，分割するクラスタの数を増やしていくと，Recall の値が下がっていくと考えられる．Precision の値が高く，Recall の値が低い場合，推薦される文書の数は少ないが，答えが含まれているとは限らないことを表している．F1-value が最大値を取るような数にクラスタを分割すると，推薦される文書の質と推薦される文書の数のバランスが取れると考えられる．

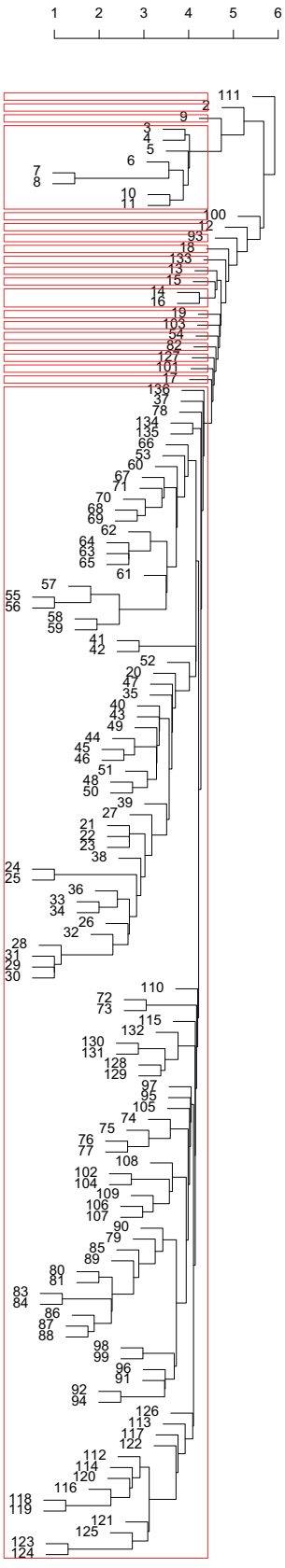

図 7: 単連結法により生成された木

5　先行研究のシステムとの比較

5.1　推薦の精度

本研究と先行研究では推薦方法が根本的に違うため評価方法を統一することはできない．先行研究の推薦システムでは，ある文書と類似度が高いと判定された文書のうち上位 5 件の文書を推薦し，推薦した 5 件の文書のうち開発者が設定したリンクが 1 つでも含まれていれば一致とみなし，開発者が設定したリンクの数に対して一致している割合を求めシステムの評価を行なっている．これに対し，本研究の提案手法ではある文書と同一クラスタに分類された文書をすべて推薦しており，順位付けはしていないため，先行研究と同じ評価をすることはできないが，先行研究で使用された評価方法と本研究で用いた評価方法のうち Recall の性質が似ているため，これらの比較を行う．ちなみに Recall は開発者が設定したリンクの数に対してリンク先の文書を推薦している割合を表している．

本研究の提案手法の Recall は，VCS では 0.816 であり，BTS では 0.994 であるのに対し，先行研究のシステムの推薦精度は，VCS では 0.533 であり，BTS では 0.579 となっている．VCS の場合は約 1.5 倍の精度の向上が見られ，BTS の場合は約 1.7 倍の精度の向上が見られた．これより，先行研究に比べ提案手法では精度が向上していることがわかる．

5.2　適用可能な範囲

先行研究では，推薦を行うための分類方法として，なでしこプロジェクトにおける不具合や要望などの特徴を示すと考えられる単語 61 個を用いていた．この単語はプロジェクトに参加している開発者の主観によって決定されているため，手法を他のプロジェクトに適用することが困難であった．また，この単語が含まれない文書を推薦することができなかった．

本研究では，全ての文書に対して TF-IDF 法を適用したため，先行研究が用いた 61 個の単語を含まない文書の推薦を行えるようになった．さらに，動的に単語を抽出することができるため，なでしこ以外のプロジェクトでも推薦を行えるようになった．

6　おわりに

本稿では先行研究で作成された，複数の開発支援システム間にまたがった検索における，同一コンテキスト情報にスムーズにアクセスできないという問題を解決するために，開発支援システムを変更することなく統合するシステムを改良し，開発者に提示する関連情報の精度を向上する手法を提案した．提案手法は，開発支援システムから取得した文書データに対して形態素解析を行い単語を抽出し，各単語の特徴ベクトルを TF-IDF 法で求める．特徴ベクトルの類似度から文書データをクラスタリングし，同一クラスタの文書が関連情報として扱い推薦する．提案手法の有用性を確認するためにオープンソースソフトウェアの開発プロジェクトの文書データに対して評価実験を行った．評価方法は Precision, Recall, F1-value を用いた．その結果，Recall の値は非常に高く，先行研究の精度と比較すると約 1.5〜1.7 倍精度が向上しており，提案手法に一定の効果があったことを確認した．

今後の課題としては，Precision と F1-value を実用できるレベルまで向上させることが挙げられる．これにはクラスタリングを行った後，分割するクラスタの数を増やすことで Precision の値が上昇すると考えられる．本研究では分割するクラスタの数を固定していたが，動的に分割する数を変更し，F1-value の値が最大値を取るようなクラスタの数を明らかにすることで，さらなる精度の向上が見込めると考えられる．

本研究ではなでしこの開発プロジェクトに対してのみ実験を行ったため，実験結果になでしこの開発プロジェクトが持つなんらかの特徴が出ている可能性がある．そのため，他の開発プロジェクトに対して提案手法を適用し検証する必要がある．また，提案手法では標準的な手法を組み合わせているため，ソフトウェア開発特有の性質を持ったデータに対して考慮できていない．データからソフトウェア開発特有の性質があるかどうかを調べ，特有の性質があった場合はソフトウェア開発に特化した推薦方法を考慮する必要がある．

参考文献

[1] Cleidson R. B. De Souza , Santhoshi D. Basaveswara , David F. Redmiles: “Supporting Global Software Development with Event Notification Servers ”, Proc. the ICSE 2002 International Workshop on Global Software Development (2002).

[2] Bikram Sengupta, Satish Chandra, Vibha Sinha: “A Research Agenda for Distributed Software Development”, In Proc. The 28th International Conference on Software Engineering (ICSE), pp.731-740 (2006).

[3] Carl Gutwin, Reagan Penner, Kevin Schneider: “Group Awareness in Distributed Software Development”, In Proc. The 2004 ACM Conference on Computer Supported Cooperative Work, pp.72-81 (2004).

[4] Soichiro Tani, Akinori Ihara, Masao Ohira, Hidetake Uwano, and Ken-ichi Matsumoto: “A System for Information Integration between Development Support Systems”, In 3rd International Workshop on Empirical Software Engineering in Practice (IWESEP), November 2011.

[5] 大平雅雄, 横森励士, 阪井 誠, 岩村 聡, 小野英治, 新海 平, 横川智教: “ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム”, 電子情報通信学会論文誌 D-I,Vol.J88-D-I, No.2, pp.228-239 (2005).

[6] 石川武志, 山本哲男, 松下 誠, 井上克郎: “ソフトウェア開発時における版管理システムを利用したコミュニケーション支援システムの提案, 情報処理学会研究報告”, 2001-SE-133,Vol.2001, No.92, pp.23-30 (2001).

[7] Google Project Hosting: “Project Hosting on Google Code”, http://code.google.com/hosting/ (2011).

[8] Geeknet, Inc.: SourceForge.net: “Find and Develop Open Source Software”, http://sourceforge.net/ (2011).

[9] Canonical Ltd.: “Launchpad”, https://launchpad.net/ (2011).

[10] MeCab: “MeCab: Yet Another Part-of-Speech and Morphological Analyzer”, http://mecab.sourceforge.net/ (2011).

[11] G. Salton, M.J.McGill: “introduction to modern information retrieval,” McGraw - Hill, NewTork (1983)

謝辞

本論文の執筆および研究を進めるにあたって，多くの方に協力していただきました．この場を借りてお礼を申し上げます．ありがとうございました．

指導教員である上野秀剛助教にはお忙しい中，数回に渡る論文のチェックを丁寧に行なっていただき，的確なご指摘をいただきました．ありがとうございます．

中間発表会では，山口智浩教授に的確なご指摘をしていただき，その後の資料の作成について非常に参考になる意見をいただきました．ありがとうございます．

査読教員である山口賢一准教授，岡村真吾講師には査読コメントで的確で鋭い質問をしていただき，本論文の修正の上で気付かされる点がいくつもあり，大変参考になりました．ありがとうございます．

また，同研究室の先輩，同輩，クラスメイトの皆様には研究を進めるにあたって様々な助言や励ましを頂き，相談もさせてもらいました．ありがとうございます．