信州大学審査学位論文

情報検索手法に基づくトレーサビリティリンク回復のための手法オプションについてのマイニングの提案と評価

2014年3月

上田健之

目次

第 1 章
まえがき

1. まえがき

ソフトウェア保守はそのソフトウェアの価値を維持し，さらに高めるための工学的に重要な工程であるのだが，生産性が低く不具合の混入しやすい困難な工程であることも知られている．この困難さの原因の一つが，ソフトウェア進化の過程で本論文が対象とするトレーサビリティリンクが失われていくことである．

トレーサビリティリンクはソフトウェアを構成する各種ドキュメント（要求文書，設計文書，テストケース，ソースコード等）の単位(要求文，設計文等)間の関連付けである．これらは開発の初期においては明確に意識され，各種ドキュメントの章立てやファイル構造として表現されたり，キーワードとして識別子やドキュメント中の単語として明記されたり ,または専用の管理ツールによって別途維持管理されるなど ,明確に存在が意識され，設計に利用される．しかし，ソフトウェアのメインテナンス工程においてその変更管理の維持がおろそかにされることが多く，次第に失われていく事が多い．

そもそも，トレーサビリティリンクの情報はソフトウェア保守工程においてこそ重要であり，ソフトウェアの保守作業においてあるドキュメント単位と何らかの関係のある別のドキュメント単位を見つける事はしばしば必要となる．代表的な用途はコード理解と変更影響範囲解析である．このような関係は理想的な開発プロセスにおいては開発の各段階でつけられ，維持管理されているといえるが，実際には各ドキュメントに対して独立に編集等が行われ，関連付けが放置され，必要となった時にゼロからリンク回復を行うという事がしばしば行われる．これをできるだけ自動的に精度よく行おうというのがトレーサビリティリンク回復研究のテーマである．

トレーサビリティリンク回復[3,4,9,22]には種々の手法が提案されているが，本論文で対象とするのは自動化手法の代表格である情報検索手法である．情報検索手法にも単語の処理方法，単語ベクトルの作成方法，リンクの判断の方法等，種々の選択肢があり，対象のドキュメントやプロジェクトの特性および回復するリンクの意味等によって選択肢の選択によるトレーサビリティリンクの回復の精度が変わってくる[15,20]．しかし ,それら選択肢の適否を識別して適切な手法を適用することは大変難しく，そのような研究はない．そもそも，個々の手法の適否，つまりその手法を適用することによって識別の精度が向上するのか否かは対象となるドキュメントの特徴に依存する物であるのであれば，少なくとも定式化されたドキュメントの特徴量をもとに手法の適否のマイニングを行う事が可能なのではないかというアイデアを発端として，実際のソフトウェア開発で使う事のできるマイニングの手順を提案するとともに，そのようなマイニングが実際のソフトウェア開発で必要となる精度で予測が可能であることを実証実験で示す．

本論文では対象ドキュメントの特性およびリンクの意味等に基づいて適切なトレーサビリティリンクの判定法を選択する方法の提案を行うと共に ,CoEST [1]で提供されている参照データ集合に基づいて，その有効性を仮説検証により実証する．

本論文の貢献は以下の点である．

トレーサビリティリンク回復を良い精度で実施するための手法を多くの手法の選択肢の中から識別する手法オプションマイニングの提案を行った

提案手法により識別した手法ではトレーサビリティリンク回復の精度の保障が可能であることを示した

以下，本論文では，2章で情報検索手法に基づくトレーサビリティリンク回復について俯瞰し，3章でトレーサビリティリンク回復のための手法オプションマイニングを提案する．4章で提案手法についての実証実験及びその評価について述べ，5章で妥当性への脅威について考察する．6章で関連研究について述べ，7章をむすびとする．

第 2 章
研究の背景

2. 研究の背景

2.1 緒言

本章では本論文の背景を総括的に述べる．トレーサビリティリンクが失われることでソフトウェア保守のどのような活動が阻害されるのかを述べるとともに，失われたトレーサビリティリンクの自動的な回復手段のなかで実行効率がよく精度も満足できるものであることで近年注目されている情報検索手法を用いたトレーサビリティリンク手法を紹介する

さらに，情報検索手法には数々の手法オプションが存在すること，それらはドキュメントとの相性がありオプションの選択がリンク回復の精度に影響を与えることを紹介する．

では，どのように正しいオプションを選択すればよいのかというのが本論文の主たるテーマである

以上の背景を本章で統括的に述べる．

2.2 ソフトウェア保守とトレーサビリティ

プログラミング言語によって記述されたソースコードは，そのソフトウェアシステムのロジックを正確に表現する目的には長けているが，使用方法や作成意図を表現する目的には向かないため，一般に実用的なソフトウェアシステムはソフトウェアのソースコードの他に数々のドキュメントから構成される．それらドキュメントは大きく分けてソフトウェアシステムのエンドユーザに当該ソフトウェアシステムの使用方法を説明することを目的とするドキュメントと，開発及び保守担当者に当該ソフトウェアシステムを開発する際の元になった要求記述，設計，テスト仕様等の記録を伝える目的のドキュメントに大別することができる．前者の目的のドキュメントが不十分であった場合，一般にそのソフトウェアシステムは使い方を理解されないためにユーザを獲得することなく，あまり使われる事なくその使命を終えるであろう．前者が充実している場合，多くのユーザを獲得し，開発者が予期し得なかった使い方による不具合の報告 ,予期し得なかったあらたな要求の発生，当初想定した利用環境を超えた新たな環境の基での動作の要求など所謂”保守”の要求が発生する．このように，有益なソフトウェアシステムがソフトウェア保守(software maintenance) [35] を受け続け，変化を受けながら使い続けられていくことをソフトウェア進化 (software evolution) と呼ぶ．ソフトウェア進化に対する主な困難の要因として，ソースコードの保守性の観点での品質の悪さ，及び後者の目的のドキュメントの貧弱さがよく知られている．前者についてはコードの良形と関連があるとされるメトリクスと進化過程で発生する不具合の予測について相関の存在を示す多くの研究が行われており，ソフトウェア工学の一つの大きな分野になっているのだが，後者についても現実の開発では多くの問題を抱えている．そもそも後者のドキュメントが質，量ともに十分でないことに基因する困難が考えられるが，質と量を充実させていくに伴い，文書の量が膨大になり，全

体を俯瞰することが不可能になる所謂「量の暴力」として現場で知られる現象が発生し，以下のような困難が発生する．

・ 要求の変更に対して影響を受ける箇所がわからない
・ 個々の要求に対してソースコードやテストの実装の存在確認が困難
・ 実装やテストに対して元の要求が何なのかわからない

後に詳しく述べるが，これらの困難が発生する根本原因は自然言語もしくは UML のような緩い基準で構造化されたドキュメント群が本来持っている意味的なつながりを追跡することが困難であることに基因する．このような，ソフトウェアのソースコードやテストコードを含む各種ドキュメント間で，ドキュメントを利用する際に必要となる関係が保たれている状態を追跡可能性があると呼び，そのような関係を追跡可能性リンク，またはトレーサビリティリンクと呼ぶ．

トレーサビリティリンクにはそれが発生するドキュメントの種類や利用方法によってさまざまな種類や意味がある．例えば”○○は××すること”という自然言語によるフレーズの集合として記述された要求記述が存在する場合 ,それらの要求は独立ではなく関係が存在し，ある要求が別の要求の前提条件になっていたり，準拠のような下位の構成要素になっていて幾つかの要求がみたされることで自動的に別のある要求が満たされているような関係が存在することが多い．このように，要求記述間という同種同列なドキュメント間での前提条件や下位構造という関係もトレーサビリティリンクである．このトレーサビリティは要求に矛盾がないことを確認する目的，特に要求への変更を考察する際のその影響の分析に必要となる．このような要求記述のトレーサビリティを管理することが可能な要求管理ツールとして商用では IBM 社 (元 Rational 社) の Doors [33]がある．Doors では要求記述間のトレーサビリティを設計時に個々「準拠する」,「満足する」等の意味付けとともに手作業で登録しておくことで後に必要となるトレーサビリティリンクをマトリックスやリンクとして参照することが可能となる．

ドキュメントの階層を超えたトレーサビリティももちろん存在する．例えば要求仕様書とテスト仕様書で記述される要求とテスト項目には「テストする」というトレーサビリティリンクが存在する．前述の Doors ではこのような関係も記録し，後に表示することが可能である．その他，要求記述とクラス図，シーケンス図のような設計文書との間，または設計文書とソースコードの間にも「実装する」というリンクがある．このようにドキュメントの階層を超えて各種リポジトリに別々に保存されるドキュメントのトレーサビリティの管理こそツールによるサポートの存在が重要になる．そのような商用ツールに Reqtify [34] がある．これは各種リポジトリに分散する既存ドキュメントに Reqtify で処理するためのタグを埋め込み，これを管理・参照することでトレーサビリティリンクを実現する．

もちろん，トレーサビリティリンクの維持・管理にツールの存在は本質的でなく，ツールサポートの期待できない環境でもトレーサビリティリンクの維持管理は手作業で行われ

ており，代表的な手段としては表計算ツールを利用しマトリックスや 2 項関係として記述され，利用されているが，作成や参照に大きな負担が生じる．

トレーサビリティリンクはこのように，保守作業担当者が必要とするドキュメント間の意味，意図(intention)の関係の集合であり，元のドキュメントには明示されていないものの意味，意図を理解する人間によって明示的に抽出し，これ自身がドキュメントとして維持･管理する対象である．

トレーサビリティリンクにはソフトウェア開発のスタイル，ステークホルダーの視点，及び開発フェーズなどの違いにより多く分類が考えられる．代表的なものとして Lindval [37] の水平トレーサビリティリンク(vertical traceability link)と垂直トレーサビリティリンク(horizontal traceability link)がある．図 1 のように種類の異なる階層間でのトレーサビリティリンクを水平，同じ階層の文書間でのトレーサビリティリンクを垂直として区別した．

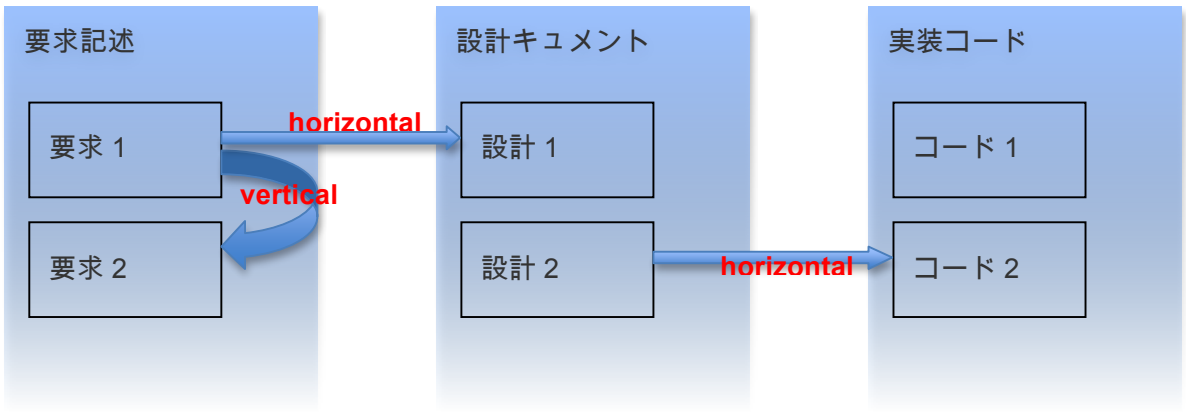

図 1 水平トレーサビリティリンクと垂直トレーサビリティリンク

Figure 1 The horizontal traceability link and the vertical traceability link.

また，Spanoudakis [38]や Ramesh [39]では以下の 7 種類の分類を行っている．

- 依存関係(Dependency relations)： このタイプのリンクにおいて，要素 e1 が要素 e2 に依存する場合は以下のどちらかである
  - 要素 e2 の存在が要素 e1 の前提条件になる
  - 要素 e2 に変更があった場合の影響が e1 に及ぶ
- 一般化/改善関係(Generalization/Refinement relations)：このタイプのリンクは以下の関係を示す
  - 複雑な要素がどのようにコンポーネント内の単純な要素にブレークダウンされるか
  - システムの構成要素がどのように結合することによって他の要素を形作るか
  - システムの構成要素がどのように他のシステムを改善するか
- 進化関係(Evolution relations)：このタイプのリンクはソフトウェアアーティファクト

間の進化関係を示す．例えば要素 e1 が要素 e2 に進化したという場合，e1 は開発，メンテナンス，またはシステムの進化を通じて e2 に改善されて置き換えられたことを示す．

- 満足関係(Satisfiability relations)：このタイプのリンクにおいて e1 が e2 を満足する場合は以下のどちらかである
  - e1 が e2 を必要とする，または要求する
  - e1 は e2 で表現される条件を満たす
- 重複関係(Overlap relations):このタイプのリンクにおいて e1 と e2 が重複する場合，両者は共通の要素を参照している．
- 競合関係(Conflict relations)：このタイプのリンクは要素間の競合を表す
- 根拠関係(Rationalisation relations)：このタイプのリンクは要素の生成や進化について背後に存在する合理的な根拠 ,または異なる詳細化レベルにおけるシステムに対する決定を表現するために使われる
- 寄与関係(Contribution relations)：このタイプのリンクは要求の要素と ,その要素の作成に寄与のあったステークホルダーの関係を示すのに使われる

トレーサビリティリンクの種類や意味が明確になったところで，これが保守で必要になる理由を再考する．トレーサビリティリンクが喪失した際に発生する問題とて前述した内容は

1. 要求の変更に対して影響を受ける箇所がわからない

これは，ある要求の変更について影響を受ける別の要求との関係で要求全体に矛盾が発生しないかという要求の整合性確認の意味と，ある要求が変更を受ける事で修正が必要になる設計や実装，テストといった下位構造への影響波及分析という二つの意味があり，あわせて”影響分析(impact analysis)” [36]とよばれるプロセスに対して発生する困難である .影響分析の場合 ,要求間の「準拠」,「満足」といった関係に加え ,階層を超えた文書間の「実装する」などありとあらゆる種類のトレーサビリティを追跡して，一つ一つ確認していくことが必要となるため，トレーサビリティリンクに高い精度が求められる．正確には精度には 2 つの観点があり，リンクであると判断した物が実際にリンクである精度と，実際にリンクである物をどれだけリンクとして拾うことができたかの精度の二つがあり，前者を適合率(precision)，後者を再現率(recall)と呼ぶ．影響分析の目的の場合，再現率が低いと見落としが発生してしまいリグレッションと呼ばれる不具合を生じさせる原因になる．適合率が低いと本来リンクでないものをリンクとして無駄な確認をすることになるため作業のコストを高くさせる事になる．再現率も適合率も高いことが理想なのだが，リグレッションを一切許容しない開発，例えばリグレッションの発生そのものを重要インシデントと捉える傾向のある公共システムの開発・保守のように不具合の重要度に関わらずリグレッションの発生を全く許容しない開発においては再現率が重要になり，少々の不具合は許容できるがコストには厳しい制約のあるシステムでは適合率が重要になる．尚，参考のため

に再現率及び適合率の定義を以下に明記する．

	予測が真	予測が偽
正解が真	True positive (tp)	False negative (fn)
正解が偽	False positive (fp)	True negative (tn)

Precision (適合率) = tp / (tp + fp)
Recall (再現率) = tp / (tp + fn)

適合率と再現率には背反的な部分があり，一般にチェックを厳しく行うと適合率は上がるが再現率は下がり，逆にチェックを甘くすると適合率は下がるが再現率はあがる．極端な例としてチェックをなくして全て T と予測すると，再現率は 1 になるが適合率は最低になる．このため適合率と再現率のバランスを評価する目的で両者の調和平均である F 値（F-measure）を確認することが多い．F 値の定義は以下になる

F 値(F-measure) = 2 * Precision * Recall /(Precision + Recall)

2. 個々の要求に対してソースコードやテストの実装の存在確認が困難

これはカバレジ分析と呼ばれ，上記の Doors, Reqtify ともに備える機能であるが，要求分析からリンクの連鎖をたどり ,その先に実装やテストが存在するかしないかで確認する．この目的でトレーサビリティリンクを利用する場合，再現率が低く存在するリンクを見落とすことによって「対応する実装やテストが存在しない」という誤った結論が導かれたとしても，その誤解はその旨の問い合わせを受けた実装者へのインタビューによって解決する．むしろ，少々の誤りを含むものであったとしてもリンクが全く利用できなかった場合のインタビューと比べてインタビュアー及びインタビュイーへの負担は大きく改善されているであろう．そのことより，再現率はあまり問題にならない．むしろ，実際には存在しないリンクをリンクと判断してしまうことにより未実装が見落とされることが問題になる．

3. 実装やテストに対して元の要求や設計が何なのかわからない

これは前述の 2 とは逆方向へのリンクの利用になり ,リンクを逆方向にたどることで元の要求や設計を探す活動になり，おもにコード理解(code comprehension)やフレームワーク理解(framework comprehension) と呼ばれる，コードリーディングの際に有益な文書を探す活動で発生する．コードリーディングは保守において本質的に重要な活動である．なぜなら保守におけるあらゆる変更はソフトウェアの元の意図(intention)を変える事なく実施しなければならない．言い換えると，保守を通じて元の意図が変更されるようなことがあると，保守を通じて新たな不具合を作り込む事になってしまう．このように，ソフトウェア保守を通じて作りこまれる不具合をレベルダウンと呼ぶことが多い．しかし，ソフトウェアのロジックはコードから読み取ることができるが，コードとして実装されたエンジ

ニアの意図はコードから完全に読み取る事はできない．例えば，3 桁の数字，ハイフン，４桁の数字という文字列をパースするコードを考えると，コードのロジックがそのようなパースを行っていることは明確であろうが，その処理の意図が郵便番号を読み取る事であるのか，電話番号を読み取る事であるのか，またはその共用であるのかをコードだけから理解することは不可能である．エンジニアによる意図は局所的にはコメントから読み取る事ができるが，その意図の元となる要求や使用，及び意図通りの実装となっているかを確認しるテストコードなどは一般にはソースコードとは別のドキュメントとして管理されるからである．大規模ソフトウェアの保守作業の困難さはこのコード理解の困難さに直接基因することが多く，各種ソフトウェアシステムの大規模保守の直接の失敗原因になっている事が多い．

この時，当該実装に対する全ての要求や設計がわからなくてもその一部でも見つかれば理解の助けになるため，再現率はあまり問題にならない．適合率が低いと無駄な文書を読むことが増えるという意味ではコストを増加させる原因になるため適合率が高いことが好ましいが，たとえ 1 割，2 割という適合率であっても，文書全体を参照しなくても意味のある関連をもつ文書をみつけることができればコード理解には有益であるため，その程度の適合率であってもこの目的には有益である．

2.3 トレーサビリティリンクの喪失

トレーサビリティリンクの対象となるドキュメントの多様性について再度，考察する．ソフトウェアを構成するドキュメントには代表的なものとして以下の様な物がある

- 要求記述
- 設計文書
- ソースコード
- テスト仕様
- テストコード

もちろん，これらに限られる訳ではなく，種々のドキュメントが付随して作成されることがある．標準フレームのようにソフトウェア開発に伴い作成すべきドキュメントを標準化しようとする動きもあるが，そもそもソフトウェア開発の規模や目的によって各ドキュメントの位置づけや重要性も変わるため，普及していない．

これらの文書はさらに記述形式にも多様である．要求記述についても単に自然言語の shall 書きによるものもあれば Use Case 記述による物もある．設計文書についてもクラス図，シーケンス図，ER 図等種々のドキュメントが存在する．

ソースコードについても，記述言語による違いの他，コメントの付け方，識別子の命名規約など多様な形式が存在する．

トレーサビリティリンクの対象となるソフトウェアドキュメントはこの様に，本質的に多様であり統一的なトレーサビリティを管理することは一般に困難である．個々のドキュ

メントの対象箇所に前述の Reqtify のようにタグを埋め込むか，個々のドキュメントの対象箇所に対応する章番号間のリンクを別ドキュメントとして管理しなければいけないのであるが，そもそも管理の単位をどのように決めることが妥当であるのかも多様である．要求記述についてもあるときは個々の要求文や Use Case を単位とすることが妥当である場合もあれば，それらの集まりの章，文書ファイルを単位とすることが妥当である場合もある．また，開発の初期においてはソースコードも含めた各ドキュメントは高い凝集性を有していることが多く，その際はある程度のまとまり，例えばファイル毎を単位とすることが妥当であるであろうが，後述するソフトウェア進化を通じて正しく管理されなかった結果として凝集性が失われていった結果としてファイルより細かい単位でのトレーサビリティリンクの管理があらたに必要になる場合もあるであろう．このようにトレーサビリティリンクを考える場合，そもそもそのドキュメントそのものの多様性，ドキュメントの執筆，改訂の自由度よりなにを単位とすることが妥当なのかといったことから問題になる．しかし，実際のソフトゥエア開発の現場においては，その現場の特徴にあわせて必要な単位を適時選択してトレーサビリティリンクの保守が行われている様である．一例として携帯電話の開発における事例を紹介する．携帯電話は膨大な数のソフトウェアから構成される情報システムであり，それらはデバイスドライバ，通信（呼）制御，TCP-IP，ディスプレイ，GPS 等，互いに互いの状態に依存する複雑なシステムである．携帯電話の開発では通信キャリアから提示される膨大な量の各種要求文書は開発元のトレーサビリティの維持管理チームによって個々の要求の単位に分解され，固有の番号をつけて管理される．番号はファイル名，章番号，節番号，節内の個々の要求の番号という形である程度，元の文書の構造を反映する形でつけられる事がある. 分解され, 固有の番号が与えられた要求は，適切な粒度でのトレーサビリティリンクの作成が行われる．例えば，機能間での状態遷移の干渉の確認の目的であれば「干渉」というリンクがドキュメントの単位で作成され，また機能内での要求と実装コードとフィールドテストの関係の追跡であれば個々の要求記述, 実装コードの関数名，テスト仕様書の章番という単位で「実装」というリンクの管理が行われる．これは数十万件にも及ぶ受け入れ試験を効率的に同時並行して行うためのスケジューリングの優先順位付けの目的で重要であり，実装のすんだ部分からテストしていくとともに多くのテスト項目をブロックする不具合から優先的に修正されていく．

一般に，ソフトウェアはその開発の初期ではトレーサビリティが明確である事が多い．トレーサビリティリンクを管理するツールを使わなかったとしても，時系列的に要求記述が纏められ，要求にあわせて設計が行われ，テスト仕様が作成されコードの実装が行われる時系列の中で，各種文書名，章立て，クラスやメソッド名などを通じてトレーサビリティリンクが明確に体系づけられて用意されることが多い.

しかし，そのように明確にされた開発初期のトレーサビリティリンクもソフトウェア保守を通じてアドホックな変更を繰り返すうちに次第に失われていくことが多い．また，トレーサビリティリンクの管理を行うツールを使っていたとしても，ソフトウェア進化に伴

うリンクの変化を自動的に抽出できるものではなく，また前章で説明したとおりそもそもソフトウェア進化を通じて適切なリンクの単位が変わってしまう事もあり，コードやドキュメントの保守にあわせてトレーサビリティリンクの記載についても適切に保守を行わなければ，トレーサビリティリンクは失われていく．

2.4 トレーサビリティリンクの回復と情報検索手法

トレーサビリティリンクが喪失している状況においてもソフトウェアの保守作業においてあるドキュメント単位と何らかの関係のある別のドキュメント単位を見つける事はしばしば必要となる．例えばコード理解や影響範囲分析のようにソフトウェアプロダクト全体の理解が求められる状況において，このような関係が利用できる場合とされていない場合では作業に要する時間が全く異なり，この関係の喪失が大きなソフトウェアシステムの保守を困難にする大きな原因の一つであると考えられている．前で述べたようにこのような関係は理想的な開発プロセスにおいては開発の各段階でつけられ，維持管理されているといえるが，実際には各ドキュメントが独立に編集等が行われ，関連付けが放置され，結果としてトレーサビリティリンクを喪失する結果につながる．必要となった時に回復を行うという事がしばしば行われる．これをできるだけ自動的に精度よく行おうというのがトレーサビリティリンク回復研究のテーマである．

トレーサビリティリンク回復は最初に述べたように対象も様々であり，また利用の仕方も様々である．そのため，これまでに各種の手法が提案されているが，手法は大きく分けて遡求的（Retrospective）なトレーサビリティリンク回復とプロスペクティブ（Prospective）なトレーサビリティリンク回復の２つに分ける事ができる[40]．遡求的なトレーサビリティリンク回復手法は自動的な手法であり，IR 手法が代表的な手法である．一方プロスペクティブなリンク回復は半自動手法であり，ルールやキーワードを決めておく等の方法を用いる．遡求的なトレーサビリティ回復は精度が落ちるが手間は少なく，対象も限定しない利点があり，組織的にプロスペクティブなリンク回復が行われていない開発プロジェクトで必要に応じて広く使われている．

情報検索手法は文書を構成する個々の単語に着目し，文書間での単語の出現の性質によりリンクを判断する．最も素朴な方法は所謂 grep によるキーワード検索で，ソースコード中のクラス名やコメント中のキーワード，各種ドキュメント中のキーワードなどで検索を行い，該当する単語が存在するドキュメントをリンクと判断する方法である．トレーサビリティリンクと意識することもなく現場で一般に素朴に使われている方法である．単純なキーワード検索ではなく正規表現を利用する等の工夫が行われる事もある．

キーワードの grep によるリンク回復では，キーワードに関連したトレーサビリティリンクの回復が主眼となるのだが，そのようなキーワードによらず，もっと一般的に文書を構成する全単語について単語の出現確率や単語ベクトルを作成し，それらに対して類似性を定量化し，その類似性よりトレーサビリティリンクの回復を行おうとする方法もある．

類似性が高いドキュメント間にはなんらかのリンクがあるのだろうとの仮定による方法である．この共起性をどのように定量化するかによって

1. 確率モデル
2. ベクトル空間モデル

がある．また単語をどのように選ぶか，単語を使うか，それともそれらをまとめたトピックをつかうかというような種類が存在する．更にはリンクを考えるドキュメントの種別によっては同じ意味であっても異なる単語（同義語，類義語）が使われるであろうということでオントロジーを考えるアプローチ[41]もある．

以上，トレーサビリティにはいろいろな種類があり，いろいろな対象ドキュメントがあり，利用目的もいろいろであり，人手による維持管理が基本であるがそれがおこなわれなかった場合は自動回復が行われ，自動回復の代表的な手法として情報検索手法があることを述べた．本論文ではその汎用性より単語ベクトル間の類似性を利用する情報検索手法を対象として，いろいろなドキュメントについてその精度を確保するための方法について考察する．この手法ではトレーサビリティの対象たるドキュメント要素間の出現単語の共起性をベースにしている．単語のドキュメントでの出現状況を表す単語・ドキュメント行列を作り，ドキュメント間の類似度を計算する．リンクの識別は，リンクを張るべき対はそうでない対よりも単語の共起性が高いであろう，という仮説に基づいている．

2.5 情報検索手法の手法オプション

情報検索手法におけるリンクの有無の判断は以下の手順となる

1. 文書をドキュメント単位に分解する
   入力：各種ソフトウェアドキュメント
   出力：ドキュメント単位
2. ドキュメント単位を単語に分解する
   入力：ドキュメント単位
   出力：単語列
3. 各単語の出現頻度より文書を単語ベクトルで表現する
   入力：単語列
   出力：単語ベクトル
4. ２つの文書の距離を単語ベクトル間の距離で計算する
   入力：複数の単語列の結合による単語文書行列
   出力：類似度
5. 上記の距離を何らかの基準で判断することでリンクの有無を判断する
   入力：類似度
   出力：リンクの有無（真偽値）

これらのステップには以下に例示するような種々のオプションが存在する

1. 文書の処理

   このステップでソフトウェアドキュメントをトレーサビリティリンクの有無を判断するための単位に分割するいくつかのオプションがある．具体的にはファイルをそのまま使う，ドキュメントの章立てやソースコードのクラス，メソッドなどの構造の構成単位に分ける，文書の個々のフレーズやコードの個々の行を単位とするなどである．文書の章立て構成は必ずしもソフトウェアを構成するドキュメント全体で統一されているとは限らないため同じ分割方法を統一的に使用することができるという保証はない．また，このように分割した際に各章節に含まれる記載内容について平均的なメトリクスが揃っているという保証はなく，かつ記載内容の粒度が揃っているという保証もない．コードについてもコメントを含めるのか，定義のスケルトン（予約語と型情報）のみを使うのか定義全体を使うのかなどの選択肢が存在する．以上より，精度のよいリンクの識別を行うためには文書の構造を理解した上で適切な単位への分割が必要になる．

2. 単語の処理

   このステップでは，単語をそのまま扱うのか，単語に何らかの加工をおこなうかによっていくつかのオプションがある．具体的にはステミング(stemming)の有無，ストップワード(stop word)除去の有無，キャメルケース(camel case)展開の有無，オントロジーの利用などがある．詳細は後の章で述べる．

3. 単語ベクトルの処理

   このステップではドキュメント中の単語数をどのように扱うかについていくつかのオプションがある．具体的には単語の出現頻度をそのままベクトルにする，単語の有無のみを考える，文書全体での重みを考慮して TF-IDF でベクトルにするなどである．詳細は後の章で述べる．

4. 単語ベクトルからトピックの抽出

   作成した単語ベクトルのドキュメントについての集まりである単語-文書行列(word document matrix)について行列としての次元縮小やトピックの抽出などを目的とした加工を行うオプションがある．行わないという選択もある．複数の単語の共起性より潜在的なトピックを抽出するとともに次元縮小を行う LSA(Latent Semantic Analysis)，トピックについての単語の確率分布を想定しギブスサンプリングなどの方法でトピックを求める LDA(Latent Dirichlet Allocation)などがある

5. 類似度計算の処理

   このステップでは二つの単語ベクトルからどのように類似度計算を行うかでいくつかのオプションがある．離散コサイン類似度が一般的だが本論文ではドキュメントの非対称性に着目した類似度計算法として非対称コサイン類似度を提案し，その有用性について考察する．他に各次元毎の相関から距離を定義するピアソンの相関係数などがある．

6. リンク有無の判断の処理

このステップでは類似度がある閾値より大きいか小さいかでリンクであるかないかを判断する事になるのだが，その閾値の決め方にいくつかのオプションがある．詳細は後に述べる．

2 の“単語の処理”では同じ意味で表現の異なる単語を同じ物と認識する方法としてステミングやオントロジーが，意味を持たない単語を省く方法としてストップワード処理が，及び合成語を分解する方法としてキャメルケース展開等がある．

ステミングとは文法要素としての法，時制，格などによる単語の変化によって同じ単語の出現数が変化形毎に分散しないようにするため，単語を語幹のみに変換する技法である．本来は文の文法構造を正しく認識した上で変形を行う必要があるのだが，正確な語幹抽出を行うかわりに簡易な方法で変換することにより近似的に正しい語幹抽出をおこなうことができ，そのようなステミング方法の一つとしてよく使われるものにポーターのアルゴリズム [31]がある．このアルゴリズムは構文解析をすることなく単に以下のような文字列のパターンマッチングと変換の規則の集まりとして定義される

- 語尾の ed の削除
- 語尾の ate の削除
- 語尾の ational の削除

これによって例えば kicked が kick に変換されることにより，kick も kicked も同じ kick として出現頻度が数えられる事になる．しかし hundred が hundr に変換される等，多くの単語が意味をなさない単語に変換されてしまうことになるのであるが，元々の目的が二つの文書間での同一の単語の出現頻度の比較であるので，二つの文書で等しく変換される以上，これは問題とはならない．ポーターのアルゴリズムの実装としてスノーボールステマーがよく使われている．

ストップワード処理とは，ドキュメントからのあまり文書を特徴付けないない単語の除去である．文書を特徴付けない単語の出現頻度を比較することは意味がないどころか情報のノイズにもなりうるため，冠詞，前置詞，代名詞等，その共起性を議論することにあまり意味のない単語をストップワードと定義し，単語ベクトリから省くことが一般的である．これをストップワード処理と呼ぶ．本論文では IR linguistic utilities [32] で用意されているストップワードリストを使用した．リストの全体は http://ir.dcs.gla.ac.uk/resources/linguistic_utils/ で参照できる．参考のため，一部を以下に例示する

- a
- about
- above
- across

- after
- afterwards
- again
- against
- all

キャメルケース展開とは，合成語の分解である．プログラムコードではクラス名，メソッド名等の識別子に空白類を使えないことからこれらをキャメルケースで表現することがよくある．キャメルケースとは例えば “CamelCase”のように，単語の先頭のみ大文字，それ以外は小文字で表記して複数の単語を空白なしでつなげる事で合成語をつくる事をいい，ケース（大文字，小文字の区別）がらくだのこぶのように見えることからエンジニアの間で慣用的に呼ばれている．単語が ”CamelCase” の様なキャメルケースであった場合，“Camel” と”Case”の様に複数の単語に展開することによって合成語を構成する個々の単語間での共起性による類似度の確認が行えるため，合成語そのものに意味がない場合に有効である反面，合成語そのものがキーワードであるような場合にその共起性が類似度の評価から消えてしまう面もある．

オントロジーとは本来は存在論を意味する哲学用語なのだが，情報工学では類義語辞典の意味でこの言葉が使われる．つまり，類義語辞典を用意して複数の類義語を同一の単語として処理することで精度をあげることを目的とするオプションである．要求分析のドメインで一般的に使われる言葉と設計のドメインで一般的に使われる言葉が，同じ実態を表すにも関わらず異なる場合，例えば通信関連の要求分析で現れる”オクテッド”と情報処理の世界でよく使われる”バイト”は，どちらも８ビットのビット列を表す同じ対象の異なる表現であり，通信の世界での”呼(call)”と情報処理の世界での”セッション”，通信の世界での”終端”と情報処理の世界での”処理”など，同一の概念がそのドメインによって別の単語で表現されることは一般的である．これらを同一のものであるとする辞書をもちいることによって同一の単語として扱う．

しかし，文書の目的ドメインによって有効な類義語辞典は異なるために一般的に有効な類義語辞典をつくることは難しい．また類義語はコンテキストによっても変わってくる．例えば時間割設計システムにおいて，”授業”と”クラス”は同義語と考えられるのだが，ソースコードのコンテキストに現れる ”Class” は授業とは無関係な，単なるオブジェクトの静的な定義であろう．このような難しさがあるため，オントロジーの利用はあまり一般的でない．

続いて 2.で述べた，認識した単語をカウントして単語ベクトルを作る際のオプションについて解説する．

最も単純な方法は単語の出現回数をカウントするのではなく，その単語がそのドキュメントに現れるか現れないかを単に１か 0 で表現する方法である．たとえば強調表現などで

同じ単語が繰り返し出現する文書のように頻度にはその文書を特徴付ける意味がなく，単に有無のみに意味がある場合に有益な方法と考えられる．また，単語の出現頻度を単純にカウントしてそのままベクトルの値とする方法もある．TF-IDF は出現文書数の重みをつけて処理する方法で，どのようなドキュメントにたいしても頻出する単語による共起性を考えることはあまり意味がないとの考え方から，あるドキュメントへの出現頻度に対してドキュメント群全体に対する出現の逆重み付けを付加する方法であり，TF(term frequency：単語頻度)に IDF(inversed Document Frequency：逆文書頻度)を掛けて計算することからこの名前がある．具体的な計算方法は TF を通常の単語出現頻度，総文書数を N，そのうち対象の単語が現れる文書数を D とした時に

tf-idf = TF * IDF

TF　= 当該単語の当該文書内での出現回数

IDF = log (N/D)

で与えられる．つまり，当該単語が現れる文書が少ないほど，その単語はその文書に特徴的な単語であることになり，IDF での重みが高くなるわけである．一つの文書にしかあらわれない単語（そのような単語の文書間での共起性を云々する意味はないが）の IDF は log(N)倍に拡大され，全ての文書に現れる単語は IDF が log(1) = 0 になる．

　例として，文書 D を単語 T_i(i=1,2,…,N)とその出現頻度 $F(T_i)$の組 (T_i, $F(T_i)$)で

D = {(T_1, $F(T_1)$),…,(T_n, $F(T_n)$)}

の様に表すとして以下の二つの文書があったとする

文書 1 = {(私,1),(ソフトウェア,2),(好き,1)}

文書 2 = {(私,1),(ドキュメント,1),(嫌い,1)}

このとき，単語ベクトル W(文書)の基底を"私","ソフトウェア","ドキュメント","好き","嫌い"とすると単語ベクトルは以下の様になる

単語の有無での表現：

W(文書 1) = (1,1,0,1,0)

W(文書 2) = (1,0,1,0,1)

単語の頻度での表現：

W(文書 1) = (1,2,0,1,0)

W(文書 2) = (1,0,1,0,1)

TF-IDF での表現（log の底を 2 とした場合）

W(文書 1) = (0,2,0,1,0)

W(文書 2) = (0,0,1,0,1)

一般の文書の場合これは自然な考え方であるが，ソフトウェアに関する文書の場合は例えば”ログ”や”入出力”などの横断的関心事（cross cutting concern） にも意味がある事から tf-idf が常に有効であるのかは不明である．

次に，3.の“単語ベクトルからトピックの抽出”について解説する．これらは作成した単語ベクトルのドキュメントについての集まりである単語-文書行列(word document matrix)について行列として追加の処理を行うオプションがある．行わないという選択もある．複数の単語の共起性より潜在的なトピックを抽出するとともに次元縮小を行うLSI[42](Latent Semantic Indexing)，トピックについての単語の確率分布を想定しギブスサンプリングなどの方法でトピックを求めるLDA[43](Latent Dirichlet Allocation)などがある．

本論文では LSI 利用の有無をひとつのオプションとして用意した．LSI の基本は，図２の上段のように任意 n 行 m 列の行列 X_{nm} を，U_{nm}，V_{mm} と対角行列 S_{mm} の積として $U_{nl}*S_{ll}*V_{lm}{}^{t}$ と表現する特異値分解である．ここで V^{t} は行列 V の転置行列を表す．

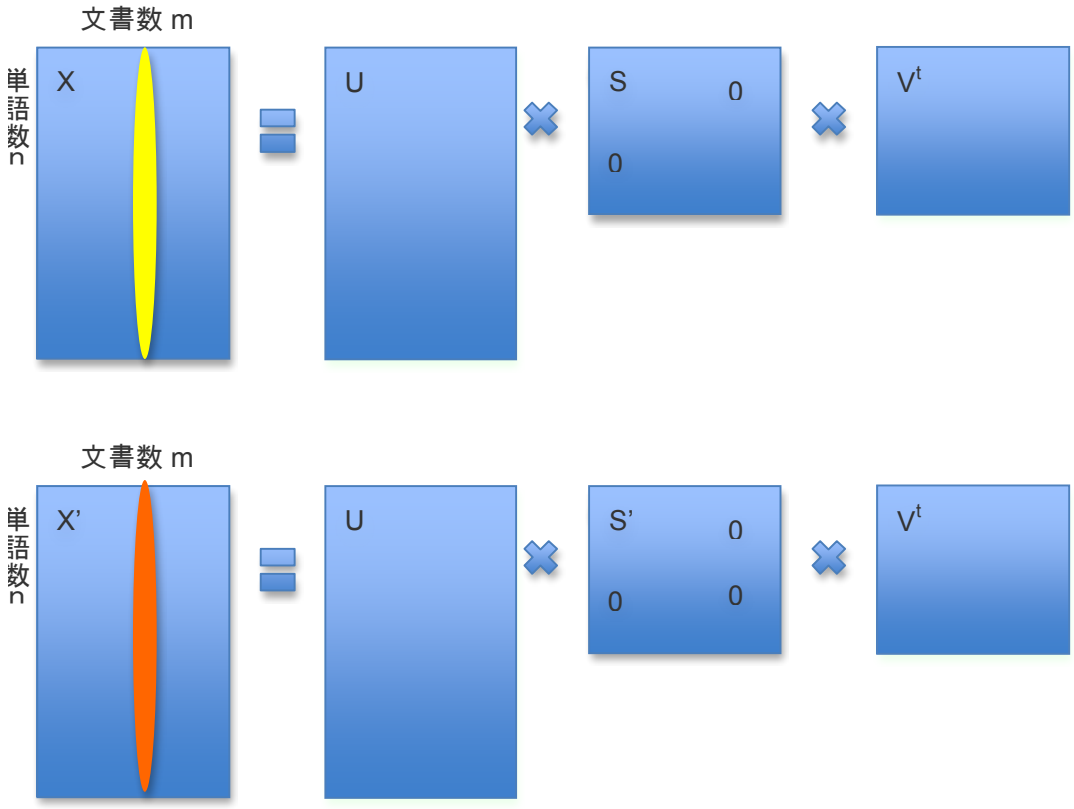

図２　行列 X の特異値分解と次元縮小

Figure 2 The singular value decomposition of matrix X and dimention reduction.

図２の上段で、単語-文書行列 X 黄色で表した列ベクトルは一つの文書に含まれる単語

の頻度ベクトルである．ここで対角行列 S の対角成分のうち、値の小さい下位群を 0 にした行列 S′を考える.m 個の対角成分のうち下位 l 個を 0 にする場合、l/n を圧縮率と呼ぶ.このように対角成分の下位群を 0 にした S′ U_{nl} と V_{lm} をそれぞれ左右から掛けて $X'_{nm}=U_{nl}*S_{ll}'*V_{lm}^{t}$ となる X′を作成し,この X′を単語-文書行列として類似度計算を行う.図 2 下段のオレンジ色で表した列ベクトルは一つの文書に含まれるトピックの頻度ベクトルと考えることができる.

次にリンク候補の判断と手法オプションについて述べる．情報検索手法ではリンク候補の判断はドキュメントの類似度が高いものをリンクと判断し，低いものをリンクでないと判断する．類似度でクラスタリングを行った際に明確に高い群と低い群の２群に分かれ，それが実際のリンクの有無と対応しているのであればそのようなクラスタリングが有効であるが実際はそのように明確でなく，なんらかの閾値を用意してそれより大きい物をリンク，小さいものを非リンクと判断する近似的な手法が使われる．この閾値の設計はリンク認識の精度に直接的な影響を与える．例えば，ある要求文書に対して設計文書が１０ある場合を考える．ここで実際のリンクが１つしかない事がわかっている場合，たとえばリンクの有無の判断として類似度の上位３位までをリンクと判断するように設計したとすると,理想的な場合でさえ 2/3 のリンク判断は誤りとなってしまう.このように,閾値の選択は精度に影響する重要なオプションであるのだが，その閾値を上位何位というランクにするオプション，上位何%というパーセンテージにするオプションなどがある.

以上，情報検索手法においては単語の認識，単語ベクトルの作成，リンクと判断する方法などに種々のオプションが存在することを述べた．それらは対象となるドキュメントの特性と相性があり，オプションの選択によって識別の精度に影響を与えるものと考えられる．さらに，次説では，ドキュメントのリンクにもいくつかの種類が存在し，その違いによっても適切な手法オプション選択が変わってくる可能性について述べる.

2.6 リンクの種別

一方トレーサビリティリンク（以下リンクと略す）といっても，その利用目的，対象によって様々なリンクがある．Ramesh[24]，Jirapanthong[14]などではリンクの参照モデルが提案されている．これらのリンクの種別には単語の共起性から考えれば，必ずしも単語の対称的な共起性（ドキュメント A と B の間の共起性は B と A の間の共起性と同じ）ではなく，包含関係であったり，概念階層の上からオントロジーを仲介として類似度を計算する方が適しているものもある．オントロジーの考慮は考えられているが[29]，類似度の計算でその非対称性が考慮されることはなかった．本論文ではそのために非対称の類似度を提案した．要求文書と設計文書のような対では有効ではないかと考えている.

さらにリンク回復といっても，その利用目的，対象，プロジェクトやプロセスによって様々な特徴を持つ．例えば，要求文書，設計文書，テスト仕様書，実装コードでは単語の共起の仕方が異なるため，適切な手法がそれぞれ異なっていることがいくつかの研究で紹

介されている[15,20]．また，リンク先が広範囲に渡る場合，例えば“横断的関心事についての要求記述とその実装コード”とその逆“特定の機能の記述とそれを対象とするテスト仕様”とでは，文書間の類似度をリンクと判断すべき妥当な閾値も異なる．すなわち対象ドキュメント対の様々な特徴に手法の適否は依存すると考えられる．

従来はこのような特徴に関わりなく同一の手法を同一の基準で適用して回復を行っていたが，それでは精度のよい回復はできない．そこでこの様な特徴を取り入れるために，特徴を定量化し，この特徴量と手法の適否の知識ベースを作り，知識ベースを参照する事により，回復精度の保障を可能とする，リンク識別手法を選択する方法を考案した．

さらに，トレーサビリティリンクにも，その利用目的や対象によって様々なリンクがあり，例えば次のようなリンクがある

1. 要求項目と，その設計もしくは実装
2. 異なった stakeholder からの同じ要求項目（異なった要求として書かれている）
3. 要求項目に対するテストケース
4. 一方の変更により充足度が影響を受ける要求項目対

これらは一方が他方に含まれる，共通の上位概念に対する別々の下位概念として独立する，共通部分を持つなどの構造が想像される．構造的に考えれば図3の5つのパターンがあると考えられる．但し実際にはこれらのパターンの複合したものが実際のパターンになる，ここで Link pattern4 では共通の上位概念を含む文書は存在しない仮想文書である．従ってどのパターンが多く含まれているかという事になる．

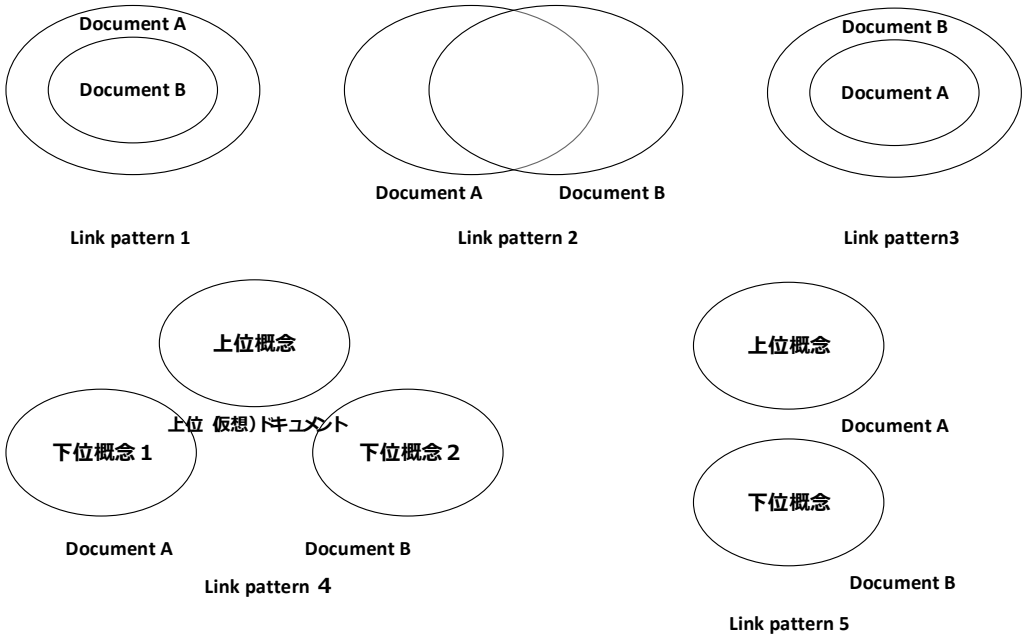

図3　リンクパターン

Figure 3　Link Pattern

- パターン１：２つが非対称なケースで，ドキュメントAの内容を膨らませているような場合で，上記１の要求項目と，その設計もしくは実装がこれに相当するであろう
- パターン２：ドキュメントA,Bが対等で，共通部分を持つというパターン．上記２の異なったステークホルダーからの同じ要求項目がこれに相当するであろう
- パターン３：パターン１の反対のパターンである．
- パターン４：この場合は直接には共起性はない．共通的な上位概念を持つことでA,B間の関係が生まれている．
- パターン５：AとBで同じ概念であっても上流，下流（もしくは視点の異なる）の用語がつかわれている場合である．

パターン２の場合と，パターン１，３の場合で共起性をどの様に定量化するかは異なるであろう．パターン２では双方に含まれない単語があるとみるが，パターン１，３では一方が他方に完全に含まれているとみる．従ってその残りの部分の多寡はリンクの評価にほとんど影響しない．一方パターン２では含まれない部分との割合で双方に含まれる単語の割合を評価するべきであろう．

2.7 非対称類似度

前章で述べた通り，文書対に含まれる単語の共起性に基づくリンクの有無の判断の目安の考え方にはそのパターンによっては非対称性があるが，従来の類似度の計算方法にはそのような考慮がなかった．そこで，我々は従来のコサイン類似度を拡張して以下のような“非対称コサイン類似度”を導入する．まず $X=(X_1,,,,X_n)$，$Y=(Y_1,,,,Y_n)$ をそれぞれドキュメントX，Yの単語ベクトルとする時，従来のコサイン類似度は以下であった

$Symmetrical\ Similarity(X,Y) = \Sigma(X_i * Y_i)/(|X| * |Y|)$

そこで同様に非対称コサイン類似度を

$Non\ Symmetrical\ Similarity(X,Y) = \Sigma(X_i * Y_i)/(|X| * |Y|)$

但し，$X_i==0$ の場合 Y_i を考慮しない

と定義する．つまり，Yのみに現れる単語については，文書対に予想される非対称性よりXとの共起関係はないと考え，類似度の計算に含めない．例として以下の二つの単語ベクトルを考える

X=(0,0,1,1,0)

Y=(1,0,1,1,0)

このとき，類似度の計算結果は各々以下のようになる

対称コサイン類似度：

$Symmetrical\ Similarity\ (X,Y) = 2/(sqrt(2)*sqrt(3))=0.816$

非対称コサイン類似度：

Non Symmetrical Similarity(X,Y) = 2/(sqrt(2)*sqrt(2))=1.0

このように定義した非対称コサイン類似度は，非対称性をもつ文書対について，より精度のよい類似度を与えるものと考える.

2.8 結言

２章では本論文の背景として本章では本論文の背景を総括的に述べた.
ソフトウェア保守の必要性とその困難性を述べ，困難性の原因の一つがとしてソフトウェアが進化(evolution)の過程を通じてトレーサビリティリンクを失っていくことにあることを述べ，その解決手段としてトレーサビリティリンクの自動回復手法，特に近年その有効性が認められている情報検索手法によるトレーサビリティリンクの自動回復について述べた.

さらに情報検索手法には種々の手法オプションが存在することを紹介し，また対象となるドキュメントのリンクにもその対称性の観点よりいくつかの種類があることを紹介し，本質的に非対称なリンクについてリンク回復の精度を挙げる手段となりうる方法として非対称類似度を定義した
では，実際の開発現場でソフトウェア開発に関するドキュメントがあり，それに対してリンク回復を実施する場合，どのオプションを選択することが適切なのか．適切なオプションを正しく選択する方法はあるのか．これが本論文が解決しようと取り組む課題である.
以下，３章で手法の提案を行い，４章でその手法が有効であるという仮説を実験により検証する.

第３章
トレーサビリティリンク回復のための手法オプションマイニング

3.　トレーサビリティリンク回復のための手法オプションマイニング

3.1 緒言

前章では本論文の背景及び基礎知識としてソフトウェア保守においてトレーサビリティリンクが必要でありながら，ソフトウェア保守を通じてトレーサビリティリンクは喪失しがちであり，そのような場合にトレーサビリティリンクを自動回復する手段として情報検索手法が有効であることを述べた．情報検索手法はドキュメント間の単語の共通性や共起性に基づくリンク回復手法なのであるが，その対象となるドキュメント間のリンクの対称性について考察し，共起性が非対称であることが予想されるドキュメントについて有効であることが期待される非対称類似度について述べた．

本章では前章で述べた情報検索手法について，精度の向上のための数々の手法オプションが存在することを解説し，そして本論文が解決しようとする課題である手法オプションのマイニング手法について提案を行う．

3.2 本論文が解決しようとする課題

２章で述べたようにリンク回復の手法には多くの選択肢があり，またその対象，用途にも多くの選択肢がある．この選択肢の手法の候補をやみくもに適用してもトレーサビリティリンク回復の精度が常に向上するわけではなく，かえって精度を低下させることもある．つまり，ある手法を適用することがどのようなドキュメントに対しても常に精度の向上をもたらす，ということはない．従来言われていたこの事実は後に実験でも示す．つまり，手法にはドキュメントとの相性があり，対象，用途とそれに対する適切な手法には何らかの関係があると考えられる．しかしながらこの関係をどのように利用するかについての研究はなく，また対象，用途に対して適切な手法を見つける方法も提案されていない．そのため，実際の開発現場でリンク回復を実践する際，手法の選択に指針がなく一般的に良いと言われている選択が行われる結果として時として良い精度をもたらさない手法が選択され，結果としてリンク回復手法自体の効果が誤解されている恐れがある．理想的な状況は，ドキュメントやそれを保守するプロジェクトの特徴量から正しい手法が選択されることだが，そのような研究はまだない．そこで，現場でのリンク回復の実践において正しい手法を選択するシステマティックな方法を提案する．

尚，本論文では特徴量の有効性の検証のみを目的としており，特徴量の最適性の判断は今後の課題としている．そのためにいくつかの特徴量の候補に基づくマイニングを試みた．

3.3 本論文の提案手法の詳細

本論文ではZhimin[12]がerror prone モジュールの予測に適用した手法マイニングの手法をトレーサビリティリンク回復の手法オプションマイニングに適用する．一言でいうと，

プロジェクトからサンプリングしたドキュメントに対してリンクの正解を用意し，考えうる種々の手法の組み合わせを元にリンク回復の予備実験を総当たりで行い，その精度と特徴量と手法との関係のマイニングを行う事で，ある特徴量をもつドキュメント対についての適切な手法を予測するというものである．この適切な手法であるという予測の精度が十分なものであれば，精度の悪い手法によるリンク回復の実施を避けることができる．

提案する手法オプションマイニングの概要を示した図 4 を元に詳細を以下に示す．ここで，矢印はデータの利用関係を表す．

1. 正解リンク情報を持った参照データ集合を用意し，ドキュメントの特徴量を抽出する．
2. 参照データ集合の各データに対して，用意した一連の手法のオプションをつかってリンク回復を行い，参照データ集合の正解を元にリンク回復の精度を求める．

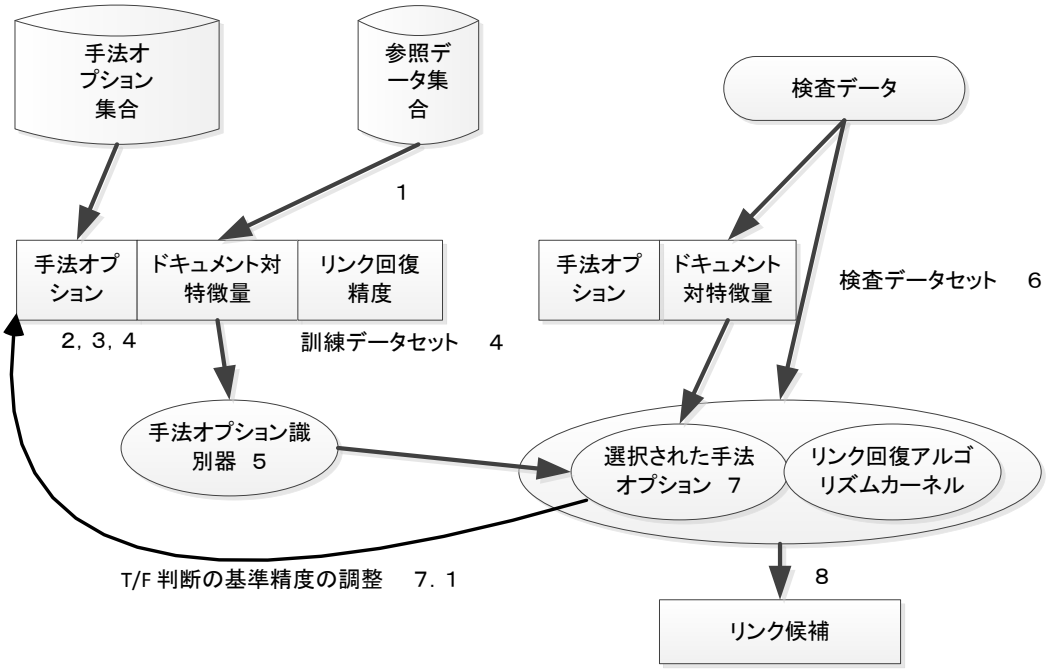

図 4　トレーサビリティリンク回復のための手法オプションマイニング

Figure 4　Method Option Mining for Traceability Link Recovery

3. 現場が求めるリンク回復の精度より，よい精度がでているか否かに応じて，2. のリンク回復の精度を True/False（以下 T/F と略す）に変換する
4. オプション選択，特徴量，T/F を組み合わせて訓練データ集合を作る．
5. 訓練データ集合を学習器にかけ，識別器を作る．
6. 検査対象データについて前述のオプション選択と特徴量から検査データ集合を作る．
7. これを識別器にかけ手法オプションを決定する．

7.1. T となるオプションが存在しない場合，現場で許容できるのであれば 3. の精度

を下げて T/F を再度決定し，3. 以下を実行する

7.2. 設定した精度では手法の選択ができないならば,このドキュメント集合についてはトレーサビリティリンク回復の実施を諦める.

8. 選択された手法オプションを使ってリンク回復を実践する.

以上が本論文で提案する手法オプションマイニングである．手法オプションとは後述する直交する選択肢の組み合わせである．このように選択された手法オプションを使ってリンク回復を実践することで，精度の良いリンク回復が可能となる．正確に表現すると，このように選択された手法オプションは，マイニングの結果として現場が求めるリンク回復の精度より良い精度を与えることが期待されるものであるので，マイニングの識別器の精度が十分によければこのリンク回復の精度は現場が満足できるものであることが，識別器の精度によって保証される.

ここで，例えば現場が非現実的な精度（適合率が 1.0 等）を求めたなどの理由で最初に現場で求めたリンク回復の精度を与える手法オプションが存在しない場合を考える．現場の理想は非現実的な精度であるのだが，実際にはそれよりも低いリンク回復の精度を受け入れる余地はあり，例えばリンクと予測したものの半分が実際にリンクであれば実用上許容可能であるにもかかわらず，理想的に適合率 1.0 を求めたような場合である．このように，実際には許容されるリンク回復の精度は固定的には考えられないので，その時はリンク回復の精度を現場が許容できる範囲で可能なオプションが現れるまで下げる．具体的には以下の判断を前述の手法の 7. 1 に記したプロセスの通り，現場で許容できるのであれば 3. の精度を下げて T/F を再度決定し，3. 以下を実行する．この時，設定した精度では手法の選択ができないならば，7.2 の通りこのドキュメント集合についてはトレーサビリティリンク回復の実施を諦める.

3.4 リンク回復精度と識別精度

ここで，二種類の異なる精度を議論するので混乱を避けるために用語を定義しておく.

リンク回復精度

定義:上記手順の 2.で述べた参照データ集合に対するリンク回復の結果を正解値と比較して得られる精度.

意味：この精度は，ある特徴量を持つドキュメント対について，与えられたオプションを使って行ったリンク回復の結果の精度と考えられる.

識別精度

定義：上記手順の 7.で述べた識別器の識別の精度.

意味：識別精度はTであると予測した選択肢のうち実際にTである選択肢の割合で評価することが適切と考え適合率を使う.

提案手法の識別器による予測の適合率が有意な値(例えば 0.8)であれば，このように選択したオプションによって実施したリンク回復精度は，8 割の確率で必要とする精度を満たしていると考えられる.

3.5 手法の選択肢

手法の選択肢には 2 章で述べた以下を用意した．全てが直交しており，これらの各々どれを採用するかで個別のオプションが決まる.

- ステミング：実施する，実施しない
- ストップワードの除去：実施する，実施しない
- キャメルケースの展開：実施する，実施しない
- LSI の利用：行う，行わない（圧縮率は 50%）
- 文書重み付け: 出現頻度，True/False，TF-IDF
- 類似度の計算方法: コサイン類似度，非対称コサイン類似度
- リンク候補の判定方法: ランクによる(5 位,10 位,15 位,20 位),閾値による(10%，20%，30 %，40%，50%，60%，70%)

これらは例示列挙であり，これらの他に LDA(Latent Dirichlet Allocation)，オントロジー等他にも種々考えうるが，実験の目的はそのような識別が可能であることを示すことでありオプションの網羅性を高めることはその事実を示す事が出来るか否かとあまり関係がないこと，LDA の適切な実装を用意することが間にあわなかったこと，オントロジーの適切な実装が困難であること等から今回の実験では採用していない.

3.6 ドキュメントの特徴量

ドキュメントの特徴量として今回は以下をリンク回復精度に影響するものと仮定した.

- ドキュメント対の平均類似度
- 類似度が 0 になるものを除いた，平均類似度
- ドキュメント対の種別：要求記述，テストケース記述，設計文書，コードの４種類の観点で同類かそうでないか
- ドキュメント対を構成するファイル数
- ドキュメント対全体の単語の種類の数
- ドキュメントの記述言語：英語，イタリア語
- 対を構成する文書の種類:shall 書きの要求文書,shall のない自然言語の要求文書，構造化された要求文書，シナリオありの use case，シナリオなしの use case，相互

作用図から文字列を抽出したもの, テストケース記述, Java のクラスのスケルトン, コメント有り java コード

手法の選択にたいしてどのような特徴量が最も適切で有効なのかはそもそもわからない. しかし, なんらかの特徴量と関係がありマイニングが可能であることを確認することがこの論文の目的であるので, 参照データセットとして利用した CoEST データセットのもつこれらの特性を元に, 手法の適用が可能であることを示すことを目的として, 今回はこれらを採用した.

3.7 識別の精度向上について

また, 提案手法では手法オプションの識別結果がマイニングにおけるクラス変数, つまり識別される対象になる. 識別という観点から値は T/F で考えるが何を持って T とするかの選択によって, 作成される識別器自身の精度が影響を受ける. この識別器の精度をどのように上げるのかという問題もあるが, 今回はそのような識別器が確率論的に手法オプションを識別できる事を示すことが目的なので, 識別器の精度向上については深くは考察しない.

3.8 結言

本章では情報検索手法によるトレーサビリティリンクの回復には数々の手法オプションが存在すること, それらの適用の可否はドキュメントの特徴量によるものと思われるものの具体的に適切な手法を選択する研究は従来なく, そこでドキュメントの特徴量から適切な手法を選択するための手法オプションマイニングの提案を行った. 次章ではこの提案手法の有益であるという仮説を設け, それについて実験による検証をおこなうことで実証的に検証する.

第４章
実証実験及び評価

4. 実証実験及び評価

4.1 緒言

第 3 章では，情報検索手法によるトレーサビリティリンク回復の手法オプションについて，ドキュメントの特徴量より適切な手法オプションを選択する方法についての提案をおこなった．

本章では，第 3 章の提案手法及び第 2 章で提案した非対称類似度についてそれが必要でありかつ有効である旨の仮説を設け，参照ドキュメントを使った実験により仮説検証を行う事により実証的に検証を行う．

4.2 検証したい仮説

本論文の提案手法の有効性について下記の仮説を用意し，実験で検証する．これらの仮説は全体として提案手法の必要性とその有効性を仮定するものである．つまり，提案手法の必要性と有効性を実験で仮説検証することが本論文の目的である．

仮説１：すべてのドキュメント対について良いリンク回復精度を与える一意な手法は存在しない．

提案手法のような識別手法の必要性についての仮説であり，これが成立しない，つまり全てのドキュメント対に常によいリンク回復精度を与える万能な手法が存在するのであればその手法を選択すればよく，本論文で主張するようなマイニング手法は不要であるということになる．種々の論文でも指摘されている仮説であるが，再度その検証を行う．

仮説２：手法オプションの適否はドキュメントの特徴量に依存する．

手法オプションの個々の適用についてその適否はドキュメントの特徴量により決まるという仮説である．提案手法のような手法が有効に機能することについての仮説であり，これが成立しない，つまり全てのドキュメントに対して常によいリンク回復精度を与える万能手法は存在しないものの，手法オプションの選択の適否がドキュメントの特徴量に依存せず，まったくランダムな物であるのであれば，そもそも本論文で主張するようなマイニングをおこなうことは無意味である．

仮説３：リンクの種別によっては非対称類似度がリンク回復精度の向上に有効である．

非対称類似度が有効であるようなリンク種別を持つようなドキュメントが存在することの仮説である．もちろん，仮説１，仮説２ですでに述べたとおり万能の手法は存在せず，非対称類似度も万能ではなく，それが有効であるようなドキュメントにおいてのみ２章で提案した非対称類似度の有効性を仮定する物である．

仮説４：本提案手法によって，妥当なリンク回復精度を与える手法オプションを有意な識別精度で識別することができる.

本論の根幹をなす仮説であり，提案手法の有効性を仮定する物である．尚，仮説４の有意な識別精度とは，手法の利用可能性より適合率が 0.8 以上と考える．この仮説により本手法によるリンク回復精度の保障の可能性を示すことになる

仮説５：目標とするリンク回復精度と識別精度には関連がある.

提案手法の有効性の限界を仮定する物で，仮説 4 の有為な識別精度は一定のリンク回復精度のもとで得られる物であり，目標としていくらでも良いリンク回復精度を設定できるわけではなく，高すぎる目標の識別精度に対しては有為な識別精度がえられるわけではないことを確認するために仮定するものである.

4.3 仮説検証のための実験

4.3.1 参照データ集合

上記の仮説を検証するために CoEST で提供されている参照データ集合に基づいて実験を行った．利用したデータ集合は６個のプロジェクトデータからなり，各プロジェクトデータは複数のドキュメント集合とドキュメント間のリンクの正解から成る.

実験は，CoEST のデータ集合（表１）の内，表２の３５個のドキュメント対で実施した．表 1 が示す通り，これらのドキュメントは何らかの偏り，例えば要求記述に偏るなどの偏りもなく，実験で確認しようとする仮説の検証の目的に対して必要な一般性を備えている物と考える.

表２は各ドキュメント集合の特徴量である．ファイル数はそれぞれ数十～百数十程度の規模で，総単語数も数百語から多いものでも 2000 語程度である．link 数は参照プロジェクトが用意した正解値の数である．link 数は 18 個(waterloo.grp34)から 1044(SMOS)まで幅が大きい.

表 1　CoEST で公開されている参照プロジェクト

Table 1 Detail of Reference Data Set in CoEST

プロジェクト名	言語	ドキュメント名	ファイル個数	内容
EAnic	イタリア語	Uc	140 個	構造化された要求文書
		Cc	55 個	コメント有り java code
EasyClinic	英語	UC	31 個	シナリオありの use case
		ID	28 個	相互作用図から文字列を抽出したもの
		TC	63 個	テストケース記述
		CC	47 個	Java のクラスのスケルトン
Gantt	英語	High	17 個	shall のない自然言語の要求文書
		Low	69 個	shall のない自然言語の要求文書
SMOS	イタリア語	Uc	67 個	構造化された要求文書
		Cc	100 個	コメント有り java code
Waterloo ※23 個のグループからなる	英語	High	1321 個	shall のない自然言語の要求文書
		Low	426 個	シナリオなしの use case
WV_CCHIT	英語	Requirements	116 個	shall 書きの要求文書
		Regulatory_code	1064 個	shall 書きの要求文書(なぜ"code" なのかは不明)

- EAnic: UC - CC
- EasyClinic: CC - TC, ID - CC, ID - TC, ID - UC, TC - CC, UC - CC, UC - ID, UC - TC
- Gant: high - low
- SMOS: uc - cc
- waterloo: 23 グループの high - low
- WV-CCHIT: Requirements - Regulatory code

表 2　参照プロジェクトの特徴量

Table 2 Characteristics of Reference Data Set

プロジェクト	ドキュメント1	ドキュメント2	ドキュメント1のファイル数	ドキュメント2のファイル数	総link数	# of word
EAnci	UC	CC	140	55	482	2676
EasyClinic	CC	TC	47	63	204	832
EasyClinic	ID	CC	20	47	69	823
EasyClinic	ID	TC	20	63	83	661
EasyClinic	ID	UC	20	30	26	702
EasyClinic	TC	CC	63	47	204	832
EasyClinic	UC	CC	30	47	93	869
EasyClinic	UC	ID	30	20	26	702
EasyClinic	UC	TC	30	63	63	681
Gantt	high	low	17	69	68	351
SMOS	UC	CC	67	100	1044	2178
waterloo.grp01	high	low	58	26	30	437
waterloo.grp02	high	low	42	13	52	275
waterloo.grp03	high	low	70	28	94	417
waterloo.grp05	high	low	54	30	47	419
waterloo.grp06	high	low	39	21	55	362
waterloo.grp08	high	low	85	22	92	452
waterloo.grp09	high	low	30	19	53	289
waterloo.grp10	high	low	76	8	69	466
waterloo.grp11	high	low	79	9	70	507
waterloo.grp13	high	low	43	8	31	325
waterloo.grp14	high	low	46	5	33	346
waterloo.grp15	high	low	69	27	93	489
waterloo.grp17	high	low	57	7	51	305
waterloo.grp18	high	low	53	8	35	219
waterloo.grp19	high	low	61	15	124	383
waterloo.grp20	high	low	93	14	139	530
waterloo.grp21	high	low	36	26	41	345
waterloo.grp23	high	low	32	20	34	182
waterloo.grp24	high	low	51	29	56	379
waterloo.grp30	high	low	48	20	35	388
waterloo.grp32	high	low	86	21	135	512
waterloo.grp33	high	low	65	11	61	431
waterloo.grp34	high	low	28	16	18	311
WV_CCHIT	Requirements	Regulatory_cod	116	1064	587	2387

以下，個々のドキュメントの内容を例示する

Nome caso d’uso

AutenticazioneAmministratore

Attori partecipanti

Iniziato da amministratore

Flusso di eventi

1. L’amministratore accede al sistema. 2. Il sistema visualizza il form di login per l’immissione dei dati (nome utente e password). 3. L’ amministratore inserisce i dati e sottomette la richiesta di autenticazione. 4. Il sistema riceve il form, verifica i dati e consente l’accesso all’amministratore al sistema con i permessi

例１　Eanic UC のサンプル

Exp 1　The sample of Eanic UC

```
package Bean;
/**
* La classe Accesso permette la gestione degli accessi
* La classe Accesso non ha dipendenze
* @author Federico Cinque
*/
public class Accesso {
        private String Login;
        private String Password;
```

例 2　Eanic CC のサンプル

Exp 2　The sample of Eanic CC

```
Input anagrafica laboratory
It allows the operator to enter
                the anagrafica of a laboratory
                analysis or any data that the
                characterize
                Has an interest to enter the age of
                laboratory within the S I O
The Operator has been recognized by `
                System (See UcValOp) and has
                all the data that characterize
```

例 3　EasyClinic UC のサンプル

Exp 3　The sample of EasyClinic UC

Operator Login Scenario validation run
The service was launched following the express request
by the actor Operator The access to the screen
on the management of the system (for instance
GUILogin) enter login and password and check the function
The validation then passes control to the instance of
GUILoginHandler which delegates to the instance of OperatoreManager
the task of managing the latter deals with
validate the data entered by using the panel
The successful operator is notified of a

例 4　EasyClinic ID のサンプル
Exp 4　The sample of EasyClinic ID

Test case Operator Login Date:
CO1's login through 20/06/2003
correct and incorrect password
torque (login password)
in S I O
Version:
0 02 000
Use Case He performs the functions
UcValOpe necessary to authenticate an
Operator

例 5　EasyClinic TC のサンプル
Exp 5　The sample of EasyClinic TC

Class GUILogin Date: 18/09/2003

Version:

0 01 000

Description Make the mask that guides the patient
the inclusion of the code of hospital card
and PIN

Attributes

	Name	Access	Description
Private	btnAnnulla		button used to return
			implemented by the mask

例 6　EasyClinic CC のサンプル

Exp 5　The sample of EasyClinic CC

4.3.2 実験手順

以下で，実験１，実験２，実験３の各々についてその手順と目的を述べる．

実験１　各種ドキュメントに対して各種手法の選択の組み合わせを用いてリンク回復の実験を行い，正解値との比較により文書，手法のオプション毎の精度を求める．この実験により仮説１及び仮説３を検証するとともに，この実験結果を元に実験２，実験３を行う．

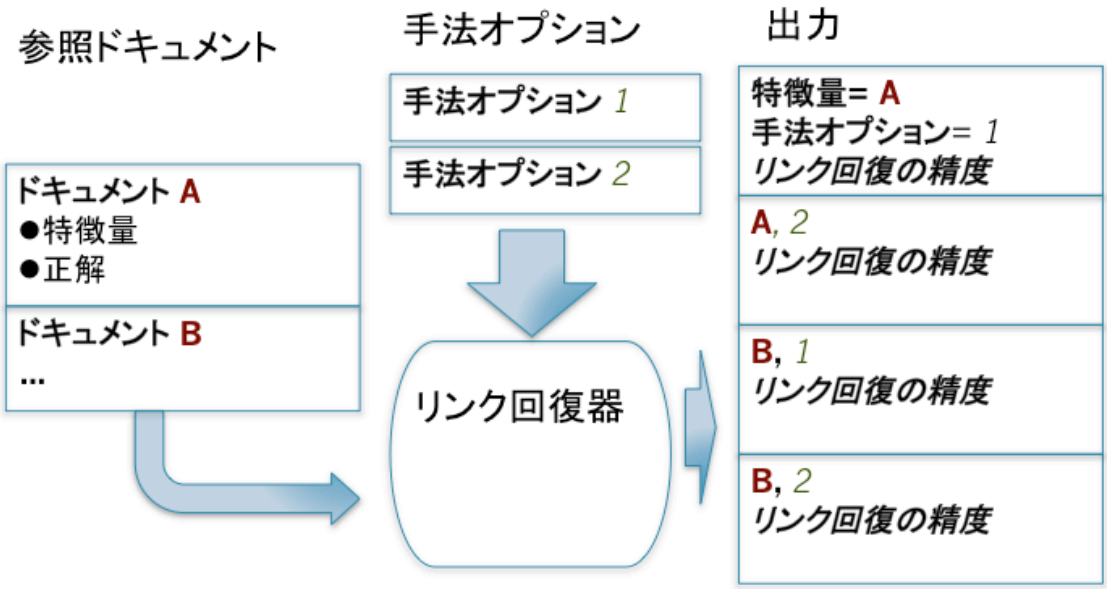

図５　実験１

Figure 5　Experiment1

実験１の詳細手順を図５に記す．参照ドキュメントを元に文書の特徴量とリンクの正解を用意する．そして，用意した複数の手法オプションの組み合わせでドキュメントをリンク回復器にかけてリンク回復を行い，回復の結果を用意したリンクの正解と比較することで，ある特徴量を持つドキュメントを特定のオプションの元でリンク回復をおこなった際のリンク回復の精度を結果として得る．つまり，実験１の結果として，特徴量，オプション，リンク回復の精度の組を得る．

今回の実験では手法のバリエーションの網羅性については考慮しない．実験の目的があくまで提案手法が有効に機能することを示すことであり，網羅的なバリエーションの中で最適値を示すことではないからである．もちろん現場で提案手法を実践する際は，網羅的なバリエーションが必要である．このことに鑑み，この実験では，３章で述べた手法のバリエーション（組み合わせの総数 1056 通り）のもとでリンク回復の実験を実施する．ドキュメント対集合は３５種であるので，総計 36960 回の回復実験を行う．

本論文でステミングには Porter[31]のアルゴリズムを用い，ストップワードには IR linguistic utilities[32]で提供されている stop word list を用いた．ドキュメントが

イタリア語で記述されている EAnic 及び SMOS についても特にイタリア語版のステマーやストップワードリストを用意することなく，他のプロジェクトに用いた物と同じ英語用のステマー及びストップワードリストを用いた．以外なことなのだが後に見るがそのことによるリンク回復の精度への影響はなく，他のドキュメントが英語で記述されているプロジェクトと同様に，ステミングやストップワード除去を行ったほうが平均するとリンク回復の精度が少しだけ良いという結果であった．

ランクによってリンク候補の判定をおこなった場合，対象となるドキュメントの数がランク値より小さかった場合は全ての文書がリンク候補として選ばれてしまうが，これは精度を下げるだけでリンク回復精度で制限されるので問題はない．

特徴量として３章で述べたものを利用した．こちらも同様に網羅的ではない．

実験２　実験１の結果を使って構成した訓練データ集合（36960 個の結果からなる）の交差検証を行い仮説４を検証するとともに作成した識別器の決定木の条件が文書対の特徴量になっている事を確認することと，その識別器を用いて有意な精度で識別が出来ることをもって仮説２を確認する．

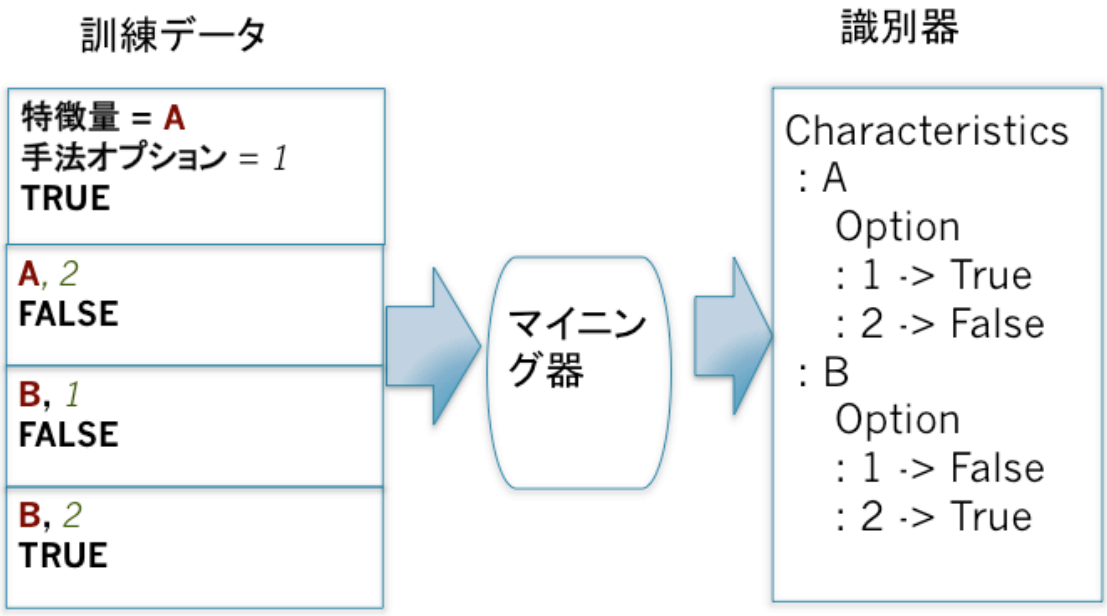

図 6　識別器の生成

Figure 6　Generation of the miner.

図 6 及び図 7 を元に手順の詳細を説明する．図 6 のように，実験１の出力である特徴量，手法オプション，及びリンク回復の精度の３つ組について，リンク回復の精度をある基準精度より高いか低いかで TRUE と FALSE に変換した物を入力としてマイニング器にかけてマイニングを行うことにより，結果として識別器を作成する．この識別器は，ドキュメ

ントの特徴量を元に適切な，つまり基準精度よりよい精度でリンクを識別することが期待される手法オプションを予測する識別器になる．このように作成した識別器と訓練データを，図 7 のように交差検証を行う事で識別の精度の評価を行う．

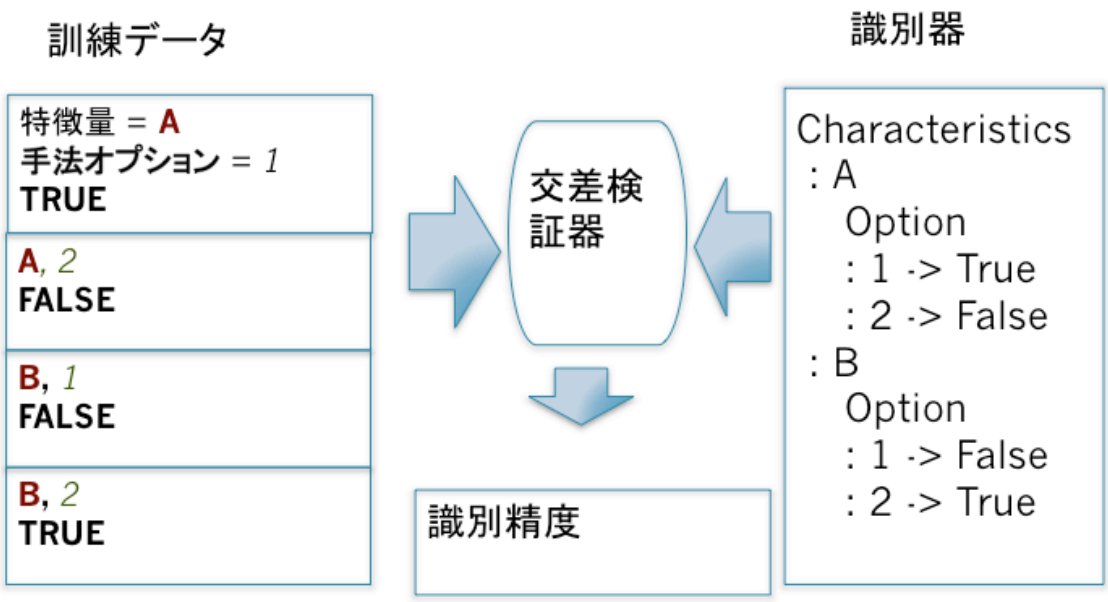

図 7 交差検証による識別器の評価

Figure 7　Evaluation of the miner by cross validation.

実験 1 の各々の試行の結果として，手法，特徴量の組に対するリンク回復の結果の精度が得られる．これについて基準精度（以下，回復の基準精度と呼ぶ）を

　適合率 ＞ 0.3 かつ再現率>0.7

とし，これを満たす試行を T，それ以外を F とした．この回復の基準精度は，Hayes[11]らが企業での経験から得られた受容可能な値として紹介している基準である．この基準を満たせない場合は，手法のステップ 7.1. でリンク回復精度を下げて実験を繰り返す．

　このように分類した各々の試行を訓練データとして識別器の作成を行い，その交差検証の精度を調べることで仮説 4 を検証する．

　識別器のアルゴリズムには精度のよい識別器を高速に作る事ができることが知られている J48 と，比較検討の為に Naïve Bayes，Logistic，Randome Forest を用いた．これらのアルゴリズムは weka によって提供されているものを利用した．

実験 3　実験 1 で作成した訓練データ集合を，プロジェクト毎に，自らの文書対によるデータだけを除き，他の全てのデータを訓練データとして作成した識別器の精度を自らの文書対に適用して調べる．これにより，識別精度に対するプロジェクトの影響を評価し，プ

ロジェクトを跨る条件下での仮説４の検証を行う.

実験２とおなじく識別器のアルゴリズムは weka によって提供されている J48, Naïve Bayes, Logistic, Random Forest を用いた.

4.4 評価

4.4.1 実験 1

各手法毎のリンク回復の F 値の平均値の比較を表 3 に記す. その手法を用いた結果が用いなかった場合に比べて 1 割以上向上しているものについて背景色を黄色とした. 表 3 の見方だが, 例えば行 EAnci の UC - CC で, 列ステミングの Yes の列の値は, ドキュメント EAnic の UC と CC との間で, ステミング以外はいろいろなオプションの組み合わせでリンク回復の実験を行った際の, ステミングを実施した場合についての F 値の平均値であり,同様にステミングの No の列の値はステミングを実施しなかった場合についての F 値の平均値である. 同様にストップワードの Yes, No はストップワード除去を実施した場合及び実施しなかった場合の, キャメルケースの Yes, No はキャメルケース展開を実施した場合及び実施しなかった場合の, LSI の Yes, No は LSI を実施した場合及び実施しなかった場合の, 単語計数の frequency, TF, TFIDF は単語係数方法としてそれぞれ頻度, True / False, ti-idf を用いた場合の, 類似度計算の対称, 非対称は類似度計算方法として対称コサイン類似度を用いた場合と非対称コサイン類似度を用いた場合での, それぞれ F 値の平均である.

ステマーやストップワードリストの対象言語が英語であるにもかかわらず, ドキュメントの記述言語がイタリア語である EAnic 及び SMOS についても, ステミングにしてもストップワード除去にしても実施したほうが僅かに精度が良く, これはドキュメントが英語で記載されれている他のプロジェクトと同じ特徴を示した.

表 3 実験 1 の結果：各手法毎の平均 f-measure 値の比較

Table 3 An Extraction of Results of Experiment 1: The comparison of average value of f-measure for each method option.

			ステミング		ストップワード		キャメルケース		LSI		単語計数			類似度計算	
プロジェクト	ドキュメント 1	ドキュメント 2	Yes	No	Yes	No	Yes	No	Yes	No	frequency	TF	TFIDF	対称	非対称
EAnci	UC	CC	0.117	0.108	0.122	0.104	0.113	0.113	0.116	0.11	0.066	0.051	0.109	0.11	0.116
EasyClinic	CC	TC	0.085	0.081	0.085	0.081	0.083	0.084	0.083	0.083	0.059	0.052	0.055	0.082	0.085
EasyClinic	ID	CC	0.128	0.117	0.134	0.112	0.124	0.121	0.131	0.114	0.066	0.058	0.122	0.122	0.123
EasyClinic	ID	TC	0.091	0.09	0.104	0.078	0.09	0.091	0.097	0.085	0.065	0.049	0.068	0.096	0.086
EasyClinic	ID	UC	0.117	0.108	0.122	0.104	0.113	0.113	0.116	0.11	0.066	0.051	0.109	0.11	0.116
EasyClinic	TC	CC	0.106	0.103	0.106	0.103	0.104	0.104	0.1	0.108	0.072	0.064	0.072	0.099	0.11
EasyClinic	UC	CC	0.117	0.108	0.122	0.104	0.113	0.113	0.116	0.11	0.066	0.051	0.109	0.11	0.116
EasyClinic	UC	ID	0.077	0.074	0.084	0.066	0.074	0.076	0.077	0.074	0.051	0.035	0.065	0.072	0.079
EasyClinic	UC	TC	0.106	0.103	0.106	0.103	0.104	0.104	0.1	0.108	0.072	0.064	0.072	0.099	0.11
Gantt	High	Low	0.11	0.093	0.112	0.091	0.101	0.102	0.105	0.098	0.063	0.065	0.076	0.1	0.103
SMOS	UC	CC	0.117	0.108	0.122	0.104	0.113	0.113	0.116	0.11	0.066	0.051	0.109	0.11	0.116
waterloo.grp01	High	Low	0.076	0.074	0.079	0.072	0.075	0.075	0.083	0.068	0.047	0.036	0.067	0.079	0.072
waterloo.grp02	High	Low	0.089	0.084	0.085	0.088	0.086	0.086	0.087	0.086	0.059	0.051	0.063	0.078	0.095
waterloo.grp03	High	Low	0.097	0.091	0.098	0.089	0.094	0.094	0.1	0.087	0.058	0.05	0.079	0.092	0.095
waterloo.grp05	High	Low	0.097	0.083	0.097	0.082	0.09	0.09	0.097	0.082	0.052	0.045	0.082	0.09	0.089
waterloo.grp06	High	Low	0.12	0.117	0.125	0.112	0.121	0.116	0.131	0.106	0.077	0.062	0.098	0.119	0.118
waterloo.grp08	High	Low	0.063	0.052	0.061	0.053	0.059	0.056	0.06	0.054	0.034	0.032	0.049	0.054	0.061
waterloo.grp09	High	Low	0.069	0.066	0.064	0.071	0.067	0.067	0.075	0.06	0.048	0.04	0.047	0.062	0.073
waterloo.grp10	High	Low	0.066	0.063	0.066	0.063	0.065	0.065	0.061	0.068	0.048	0.046	0.035	0.056	0.073
waterloo.grp11	High	Low	0.086	0.073	0.089	0.069	0.079	0.079	0.079	0.08	0.056	0.048	0.055	0.071	0.088
waterloo.grp13	High	Low	0.104	0.094	0.101	0.097	0.103	0.096	0.107	0.091	0.066	0.057	0.075	0.092	0.107
waterloo.grp14	High	Low	0.111	0.092	0.108	0.094	0.101	0.101	0.105	0.098	0.071	0.058	0.073	0.087	0.115
waterloo.grp15	High	Low	0.084	0.083	0.09	0.078	0.083	0.084	0.09	0.078	0.056	0.048	0.063	0.081	0.086
waterloo.grp17	High	Low	0.099	0.093	0.103	0.089	0.096	0.096	0.1	0.092	0.072	0.061	0.059	0.087	0.105
waterloo.grp18	High	Low	0.129	0.119	0.137	0.111	0.123	0.125	0.14	0.108	0.085	0.067	0.096	0.12	0.128
waterloo.grp19	High	Low	0.109	0.101	0.106	0.103	0.105	0.105	0.106	0.103	0.076	0.068	0.066	0.097	0.113
waterloo.grp20	High	Low	0.088	0.086	0.09	0.083	0.086	0.087	0.085	0.089	0.06	0.051	0.062	0.078	0.096
waterloo.grp21	High	Low	0.096	0.084	0.097	0.083	0.091	0.09	0.105	0.075	0.051	0.039	0.091	0.088	0.093
waterloo.grp23	High	Low	0.128	0.119	0.126	0.121	0.124	0.124	0.136	0.111	0.077	0.065	0.105	0.124	0.123
waterloo.grp24	High	Low	0.084	0.072	0.084	0.072	0.078	0.078	0.087	0.069	0.05	0.041	0.065	0.077	0.079
waterloo.grp30	High	Low	0.06	0.049	0.064	0.046	0.055	0.055	0.058	0.052	0.028	0.028	0.053	0.051	0.059
waterloo.grp32	High	Low	0.077	0.071	0.075	0.073	0.074	0.074	0.077	0.071	0.053	0.044	0.051	0.068	0.08
waterloo.grp33	High	Low	0.077	0.069	0.075	0.071	0.073	0.073	0.077	0.069	0.05	0.043	0.053	0.067	0.08
waterloo.grp34	High	Low	0.087	0.075	0.089	0.073	0.081	0.081	0.089	0.073	0.05	0.045	0.067	0.083	0.079
WV_CCHIT	Requirements	Regulatory_code	0.043	0.031	0.043	0.031	0.037	0.037	0.038	0.036	0.018	0.019	0.037	0.038	0.037

ここからは，ステミング，ストップワード，LSI については黄色になっている物があればそれは Yes に偏っている事よりプロジェクトによって用いたほうが用いないよりも向上するものがあると読み取れる．しかし，例えば waterloo.grp11 の High low についての LSI の有無での F 値の値は LSI 有りの場合が 0.079，LSI なしの場合が 0.08 と，わずかながらに LSI なしの場合の方が良い平均値を示す例もあり，必ず実施したほうが良い値を示しているわけではない．ここでみている物は各オプションの組み合わせの平均値であって，個々のオプションによっては相性の問題で高い精度を出している可能性がある．

そこで，ステミング，ストップワード，キャメルケース，LSI の各々の有無の 16 通りの組み合わせについて，その手法のバリエーションによるリンク回復の F 値が 35 個のプロジェクトの中で上位３位までに入った頻度と，T とする基準を適合率>0.1，再現率>0.5 の基準で T となった数を調べ表 4 に記す．後者で，回復の基準精度ではない基準で T の数を調べたのは，その基準では T が全体の 1%と少な過ぎたためであり，T の数が 26.5%になる前述の基準で確認した．表 4 の見方だが，いろいろなドキュメントについていろいろなオプションでリンク回復の実験をおこなった際の，特定のオプションの組み合わせについて，例えば最初の行であれば ステミングを実施し，ストップワード処理を実施し，キャメルケース展開を実施し，LSI を実施した物が，リンク回復精度の上位 3 位以内に現れた個数が 11 であることと，そのときリンク回復が T となった個数が 714 であったことを表している．

表 4 より，上位３位にあらわれた個数は最大でも 14 個であり，実験対象とした文書対３５個の半分以下である．つまり３５個の文書対の殆どのものについて常に上位となる有効な手法オプションは存在していない．

表 4 実験 1 の結果：手法オプションの出現頻度

Table 4 Result of Experiment 1: The frequency of appearance for method options

ステミング有無	ストップワード処理有無	キャメルケース展開有無	LSI 有無	上位3位に現れた個数(合計:105)	T になった個数（合計:9807）
有	有	有	有	11	714
有	有	有	無	12	696
有	有	無	有	14	724
有	有	無	無	9	695
有	無	有	有	8	626
有	無	有	無	10	568
有	無	無	有	4	626
有	無	無	無	8	565
無	有	有	有	5	630
無	有	有	無	5	597
無	有	無	有	2	622
無	有	無	無	5	602
無	無	有	有	0	560
無	無	有	無	5	516
無	無	無	有	2	552
無	無	無	無	5	514

但し，頻度には偏りがあり，この偏りが表２の平均値の差を生じさせたものと考えられる．表 4 でひとつだけ出現回数が０のものがあるが，他の手法による頻度の偏りや頻度の小ささなどを鑑み，これについてもこのような組み合わせが良い精度を生まない事を確認したわけではなく，今回の実験でたまたまみつからなかっただけと考えている．以上より仮説１は検証されたものと考える．

仮説３を考察するために表 3 に戻る．右端の類似度計算欄を確認すると，いくつかのプロジェクトで非対称類似度を用いたほうが F 値の平均値が良かった．このことより仮説３は検証されたものと考える．非対称な類似度は２章で説明した非対称なリンクパターンに対して有効であると思われるが，本論文ではパターン毎の有効性の確認はできておらず，今後の研究課題としたい．また他のリンクパターンに対する類似度計算方法についても今後の課題である．

4.4.2 実験 2

仮説２“手法オプションの適否はドキュメントの特徴量に依存する．“は,実験２のマイニングの結果の決定木の条件に特徴量が使われている事，つまりマイニングは特徴量によ

って行われていることを確認し, さらにそのマイナーを利用して有意な精度でその手法の適否を判断できることで検証した. 参考のため, 実験２で作成した決定木の一つの一部を以下に示す. この決定木はアルゴリズムとして J48 を用いてマイニングを行った結果である.

```
wordcount <= 180
|   linkjudge = rank: false (480.0)
|   linkjudge = thresh
|   |   filecount1 <= 42
|   |   |   averagesimilarity2 <= 0.131
|   |   |   |   thresh <= 0.3
|   |   |   |   |   thresh <= 0.1
|   |   |   |   |   |   islsi = nolsi: true (6.0)
|   |   |   |   |   |   islsi = lsi: false (4.0)
|   |   |   |   |   thresh > 0.1
|   |   |   |   |   |   islsi = nolsi: false (12.0)
|   |   |   |   |   |   islsi = lsi
|   |   |   |   |   |   |   thresh <= 0.2: false (4.0)
|   |   |   |   |   |   |   thresh > 0.2: true (4.0)
|   |   |   |   thresh > 0.3: false (40.0)
|   |   |   averagesimilarity2 > 0.131: false (434.0/2.0)
|   |   filecount1 > 42
|   |   |   variancesimilarity2 <= 0.029
|   |   |   |   thresh <= 0.3
|   |   |   |   |   variancesimilarity1 <= 0.026
|   |   |   |   |   |   thresh <= 0.1
|   |   |   |   |   |   |   variancesimilarity1 <= 0.007
|   |   |   |   |   |   |   |   isstem = nostem: false (2.0)
|   |   |   |   |   |   |   |   isstem = stem: true (2.0)
|   |   |   |   |   |   |   variancesimilarity1 > 0.007: false (20.0)
|   |   |   |   |   |   thresh > 0.1
|   |   |   |   |   |   |   variancesimilarity1 <= 0.007: false (8.0)
|   |   |   |   |   |   |   variancesimilarity1 > 0.007
|   |   |   |   |   |   |   |   islsi = nolsi
|   |   |   |   |   |   |   |   |   thresh <= 0.2
|   |   |   |   |   |   |   |   |   |   variancesimilarity2 <= 0.01: true (4.0)
|   |   |   |   |   |   |   |   |   |   variancesimilarity2 > 0.01: false (8.0)
|   |   |   |   |   |   |   |   |   thresh > 0.2
|   |   |   |   |   |   |   |   |   |   variancesimilarity2 <= 0.01: false (4.0)
```

決定木の一部

この決定木で，文書の特徴量として現れているものは以下である

- wordcount ：文書を構成する平均の単語数
- filecount ：文書を構成するファイル数
- avagagesimilarity：文書館の平均類似度
- variancesimilarity：文書館の類似度の分散

また，手法オプションとしてあらわれているものは以下である

- linkjudge = rank ：リンクの判断を上位何位までのランクで行うオプション
  rank：cut point とした閾値の値
- linkjudge = thresh：リンクの判断を上位からのパーセンテージで行うオプション
  thresh ：cut point とした閾値の値
- islsi：lsi の実施有無
  nolsi：lsi 実施せず
  lsi：lsi 実施
- isstem：ステミングの実施有無
  nostem：ステミング実施せず
  stem：ステミング実施

表 5 実験２の結果

Table 5 Result of Experiment 2

Tとする精度	適合率	0.3
	再現率	0.7
	T の%	1.0%
識別器の精度	適合率	0.87
	再現率	0.68
	F-measure	0.76

また，J48 による結果を表 5 に示す．上段は，リンク回復精度及びそのときの全体に占めるＴの比率であり,下段はその条件で作成した実験２の識別器の交差検証の結果である．T が 1%しか存在しない適合率>0.3，再現率 >0.7 という回復の精度基準についても 0.87 という高い適合率で識別をおこなうことが出来ている．これにより，仮説４で述べたとおり，回復の基準精度のような妥当なリンク回復精度を得る手法を，0.8 以上の識別精度で識別することが判明した．

一般に，識別の精度は識別器のアルゴリズム自身の性能や対象との相性の影響を受けるものと考えられる．その影響を確認するため，実験２の同じデータを用いて他のアルゴリ

ズムでマイニングを行った結果を表 6 に示す.

表 6 アルゴリズム毎の実験２の結果
Table 6 Result of Experiment 2 with several algorithms.

	適合率	0.3
	再現率	0.7
T とする精度	T の%	1.0%
識別器の精度（適合率）	**J48**	**0.87**
	Naïve Bayes	**0.08**
	Logistic	**0.58**
	Random Forest	**0.75**

表 6 の識別器の精度（適合率）が示す通り，提案手法による識別の精度は選択したアルゴリズムの影響を大きく受ける．表 6 は J48 の優位性を示しているように見えるが，常に J48 が妥当なアルゴリズムであるのかはこの結果だけではわからず，手法オプションと同様になんらかの相性がある可能性も残っていると考える．本論文の目的は提案手法のマイニングが可能であることを示すことであるため，ここでは上記についての詳細に立ち入ることはせず，今後の研究対象として残したい．

実験２は交差検証であるので，自分自身を訓練データと検査データに分けており，自分自身の結果も参照して手法オプションの選択を行っている可能性がある．従ってある意味理想的な条件下での結果とも言える．実験３ではさらに個別のプロジェクト毎の識別精度の変動を調べ，この手法がどこまで一般的に適用できるものなのかを評価した．

4.4.3 実験 3

4.4.3.1 結果の考察と提案手法によるリンク回復精度の低減

仮説４の一般性の検証のための識別精度の検証の実験である，実験３の結果を表 7 に記す．ここで精度を記載していない(斜線のある) EAnic, WV-CCHIT 等についてはリンク回復精度が T となる手法が存在しないので識別実験は実施していない．表 7 の各々の最初の３つの列は実験の対象を表し，それぞれ対象としたプロジェクト名及び対となるドキュメント１とドキュメント 2 の種別である．次の 3 列はそのドキュメントの特徴料の抜粋で，ドキュメント１のファイル数，ドキュメント２のファイル数，及びそれらの平均リンク数で

ある.次の二つのブロックはそれぞれリンク回復精度を回復の精度基準である“適合率>0.3 かつ再現率>0.7”にした時,及び十分な識別精度を得られなかった場合の提案手法の Step7 に従いリンク回復精度を下げて “適合率>0.1 再現率>0.5”にした時の T の数と, 識別器作成するアルゴリズムを J48 と, J48 に後述する重みをつけたもの, 及びアルゴリズムの差を確認するための実験として Naïve Bayes, Logistics のアルゴリズムで作成した際の実験３の結果の識別精度である.

表 7 実験 3 の結果

Table 7 An extraction of Results of Experiment 3

プロジェクト	ドキュメント 1	ドキュメント 2	file1	file2	平均 link 数	適合率>0.3 and 再現率>0.7				適合率>0.1 再現率>0.5			
						T の数	識別の 適合率			T の数	識別の 適合率		
						(Total 358)	J48 重み無	J48 重み有	NaiveBayes	(Total 9973)	J48 重みなし	J48 重み有	Logistic 重み無
EAnci	UC	CC	140	55	3.44	0				0			
EasyClinic	CC	TC	47	63	4.34	0				262	0.545	0.571	0.155
EasyClinic	ID	CC	20	47	3.45	89	0.763	0.639	0.131	428	0.937	0.897	0.617
EasyClinic	ID	TC	20	63	4.15	37	0.107	0.258	0.065	352	0.768	0.83	0.434
EasyClinic	ID	UC	20	30	1.3	13	0.05	0.045	0.025	282	0.942	0.801	0.638
EasyClinic	TC	CC	63	47	3.24	0				476	0.938	0.949	0.687
EasyClinic	UC	CC	30	47	3.1	47	0	0.191	0.096	415	0.861	0.812	0.613
EasyClinic	UC	ID	30	20	0.87	15	0	0.149	0.031	228	0.844	0.85	0.493
EasyClinic	UC	TC	30	63	2.1	16	0	0.118	0.029	313	0.617	0.576	0.389
Gantt	high	low	17	69	4	0				336	0.54	0.66	0.318
SMOS	UC	CC	67	100	15.58	0				53	0	0.046	0
waterloo.grp01	high	low	58	26	0.52	0				136	0.114	0.188	0.049
waterloo.grp02	high	low	42	13	1.24	0				402	0.83	0.761	0.735
waterloo.grp03	high	low	70	28	1.34	0				260	0.822	0.836	0.743
waterloo.grp05	high	low	54	30	0.87	2	0	0	0	260	0.804	0.87	0
waterloo.grp06	high	low	39	21	1.41	0				385	0.926	0.893	0.69
waterloo.grp08	high	low	85	22	1.08	0				37	0.127	0.13	0.192
waterloo.grp09	high	low	30	19	1.77	0				164	0.201	0.136	0.401
waterloo.grp10	high	low	76	8	0.91	0				234	0.848	0.858	0.861
waterloo.grp11	high	low	79	9	0.89	0				296	0.992	0.978	0.913
waterloo.grp13	high	low	43	8	0.72	0				395	0.89	0.896	0.738
waterloo.grp14	high	low	46	5	0.72	24	0	0	0.117	342	0.963	0.96	0.73
waterloo.grp15	high	low	69	27	1.35	0				320	0.926	0.998	0.82
waterloo.grp17	high	low	57	7	0.89	0				328	0.852	0.939	0.771
waterloo.grp18	high	low	53	8	0.66	78	0	0		460	0.958	0.956	0.776
waterloo.grp19	high	low	61	15	2.03	0				544	0.82	0.855	0.891
waterloo.grp20	high	low	93	14	1.49	0				376	0	0	0.93
waterloo.grp21	high	low	36	26	1.14	25	0	0	0.076	281	0.632	0.671	0.924
waterloo.grp23	high	low	32	20	1.06	12	0.094	0	0.051	454	0.842	0.664	0.536
waterloo.grp24	high	low	51	29	1.1	0				154	0.449	0.281	0.549
waterloo.grp30	high	low	48	20	0.73	0				64	0.217	0.208	0.169
waterloo.grp32	high	low	86	21	1.57	0				252	0.714	0.682	0.466
waterloo.grp33	high	low	65	11	0.94	0				402	0.769	0.71	0.689
waterloo.grp34	high	low	28	16	0.64	0				282	0.519	0.343	0.428
WV_CCHIT	Requirements	Regulatory_code	116	1064	5.06	0				0			

先ず，リンク回復精度を回復の精度基準とした時の，アルゴリズムを重みなしの J48 とした時の識別精度を見ると，例外的に EasyClinic の ID－CC で 0.763 という値が得られている以外，同じ EasyClinic でも ID － TC では 0.107，UC － CC では 0，TC － CC ではそもそも T となったリンク回復が 0 で識別不能など，ほとんどのドキュメントに対してまったく正しく識別できていないことがわかる．T となるものが全体の 1.0%ときわめて少なく，アルゴリズムの最適化のためにほぼ全てを F と評価してしまう識別器が作られてしまうことによる誤評価の可能性を考えた．以下にそのように作成された識別器の識別結果の Confusion Matrix をいくつか例示する．ここで列 a は T と識別された個数，列 b は F と識別された個数，行 a は結果が T であった個数，行 b は結果が F であった個数である．

EasyClinic UC ID の Confusion Matrix

```
    a    b   <-- classified as
    0   15 |    a = true
    0 1041 |    b = false
```

Waterloo grp05 の Confusion Matxrix

```
    a    b   <-- classified as
    0    2 |    a = true
    0 1054 |    b = false
```

Waterloo grp14 の Confusion Matxrix

```
    a    b   <-- classified as
    0   24 |    a = true
    0 1032 |    b = false
```

そこで，T となる訓練データに 50 倍の重みを与えて再度確認を行った．この時の結果を回復の基準精度（適合率>0.3 and 再現率>0.7）での列”J48 重み有り”に記載したが顕著な識別精度の向上は得られていない．また J48 の特性による可能性もあり Naïve Bayes での確認結果も右隣の列“NaiveBayes”に記載したが識別精度は低いままであった．

回復の精度基準では，実験２の結果はポジティブであったが，実験３の結果はネガティブなものとなった．つまり，理想的な条件ではこの基準でもよい精度で識別ができる可能性をしめしているが，現実の状況ではこの基準では有意な識別精度が得られない可能性を示している．そこで，提案手法のステップ 7.1.でリンク回復精度を下げて実験を繰り返し，適合率>0.1，再現率 >0.5 までさげた結果，表 7 の右側のように，ほとんどのドキュメント対に対して，0.8 以上の適合率で正しい手法を選択できた．ここまで下げた回復精度が現場で許容しうるかどうかは現場次第であろう．変更影響範囲解析の目的でのリンク回復

であれば，漏れのない高い適合率が必要となるが，フレームワーク理解の目的で関連ドキュメントをいくつか読む目的の場合，そもそも再現率が高い事が好ましく，かつ 0.5 程度の再現率であればかなり作業効率改善に貢献していると考えられる．

この結果より識別精度は目標とするリンク回復精度と関係があることがわかり，仮説５が確認された．

4.4.3.2 識別精度についてのついての考察

リンク回復精度と識別精度との関係を考察するために更にいくつかのリンク回復精度の元で実験 3 を行った結果を表 8 に示す．表 8 は，リンク回復の精度を適合率>0.5，再現率>0.7 から段階的に適合率>0.05，再現率>0.3 まで下げていきながら実験３を行い，それぞれのドキュメントについて列 ”# of T” でそのリンク回復精度で T となったリンク回復の数を，列 ”Precision” で識別の適合率を，列 “Recall” で識別の再現率を，及び列 “F-Measure” で識別の F 値を示す．

表 8 実験 3 の結果：リンク回復精度と識別精度の関係

Table 8 An extraction of Results of Experiment 3 : The relation between the link recovery criterion and identification performance.

precision of the link recovery	0.5				0.3				0.2				0.1				0.05			
recall of the link recovery	0.7				0.7				0.6				0.5				0.3			
	# of T	Precision	Recall	F-Measure	# of T	Precision	Recall	F-Measure	# of T	Precision	Recall	F-Measure	# of T	Precision	Recall	F-Measure	# of T	Precision	Recall	F-Measure
EAnci_UC_CC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	296	0.52	0.99	0.681
EasyClinic_CC_TC	0	0	0	0	0	0	0	0	14	0.06	0.4	0.1	262	0.55	0.53	0.536	532	0.8	0.82	0.809
EasyClinic_ID_CC	36	1	0.6	0.71	89	0.76	0.33	0.46	224	0.81	0.5	0.6	428	0.94	0.91	0.922	704	0.96	0.96	0.96
EasyClinic_ID_TC	2	0	0	0	37	0.11	0.08	0.09	146	0.08	0	0.05	352	0.77	0.88	0.822	573	0.87	0.77	0.82
EasyClinic_ID_UC	0	0	0	0	13	0.05	0.31	0.09	110	0.4	0.6	0.47	282	0.94	0.75	0.832	593	0.8	0.87	0.833
EasyClinic_TC_CC	0	0	0	0	0	0	0	0	157	0	0	0	476	0.94	0.82	0.877	645	1	0.97	0.987
EasyClinic_UC_CC	19	0	0	0	47	0	0	0	164	0.36	0.1	0.15	415	0.86	0.6	0.704	708	0.96	0.87	0.909
EasyClinic_UC_ID	0	0	0	0	15	0	0	0	68	0.36	0.5	0.43	228	0.84	0.66	0.742	535	0.95	0.59	0.726
EasyClinic_UC_TC	2	0	0	0	16	0	0	0	60	0.14	0.3	0.18	313	0.62	0.71	0.66	584	0.78	0.87	0.826
Gantt_high_low	0	0	0	0	0	0	0	0	32	0.11	0.9	0.19	336	0.54	0.96	0.692	668	0.82	0.37	0.511
SMOS_UC_CC	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	137	0.15	0.23	0.179
WV_CCHIT_Requirements_Reg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	136	0.11	0.18	0.136
waterloo.high_low.grp01	0	0	0	0	0	0	0	0	0	0	0	0	136	0.11	0.34	0.171	492	0.58	0.82	0.677
waterloo.high_low.grp02	0	0	0	0	0	0	0	0	0	0	0	0	402	0.83	0.95	0.886	708	0.91	0.92	0.913
waterloo.high_low.grp03	0	0	0	0	0	0	0	0	0	0	0	0	260	0.82	0.96	0.887	756	0.94	0.94	0.944
waterloo.high_low.grp05	0	0	0	0	2	0	0	0	48	0	0	0	260	0.8	0.29	0.42	608	0.93	0.66	0.772
waterloo.high_low.grp06	0	0	0	0	0	0	0	0	28	0	0	0	385	0.93	0.52	0.666	885	1	0.76	0.86
waterloo.high_low.grp08	0	0	0	0	0	0	0	0	0	0	0	0	37	0.13	0.78	0.218	486	0.74	0.98	0.845
waterloo.high_low.grp09	0	0	0	0	0	0	0	0	4	0	0	0	164	0.2	0.45	0.278	544	0.73	1	0.842
waterloo.high_low.grp10	0	0	0	0	0	0	0	0	2	0	0	0	234	0.85	1	0.918	498	0.92	1	0.958
waterloo.high_low.grp11	0	0	0	0	0	0	0	0	30	0	0	0	296	0.99	0.84	0.908	548	0.99	0.91	0.947
waterloo.high_low.grp13	0	0	0	0	0	0	0	0	47	0.32	0.8	0.46	395	0.89	0.86	0.875	667	0.93	0.83	0.876
waterloo.high_low.grp14	0	0	0	0	24	0	0	0	118	0.75	0.1	0.18	342	0.96	0.92	0.943	530	0.81	0.96	0.876
waterloo.high_low.grp15	0	0	0	0	0	0	0	0	10	0	0	0	320	0.93	0.86	0.893	722	0.93	0.93	0.93
waterloo.high_low.grp17	0	0	0	0	0	0	0	0	86	0.38	0.6	0.46	328	0.85	0.95	0.899	590	0.84	0.82	0.83
waterloo.high_low.grp18	24	0	0	0	78	0	0	0	189	0.68	0.4	0.48	460	0.96	0.79	0.868	624	0.77	0.88	0.817
waterloo.high_low.grp19	0	0	0	0	0	0	0	0	54	0.05	0	0.04	544	0.82	0.5	0.624	644	0.97	0.97	0.972
waterloo.high_low.grp20	0	0	0	0	0	0	0	0	36	1	0.1	0.11	376	0	0	0	633	0.98	0.97	0.975
waterloo.high_low.grp21	4	0	0	0	25	0	0	0	80	0	0	0	281	0.63	0.41	0.497	697	0.93	0.85	0.887
waterloo.high_low.grp23	0	0	0	0	12	0.09	0.5	0.16	110	0.44	0.4	0.4	454	0.84	0.42	0.563	920	1	0.85	0.92
waterloo.high_low.grp24	0	0	0	0	0	0	0	0	0	0	0	0	154	0.45	0.94	0.606	646	0.9	0.79	0.843
waterloo.high_low.grp30	0	0	0	0	0	0	0	0	0	0	0	0	64	0.22	0.56	0.313	414	0.62	0.97	0.753
waterloo.high_low.grp32	0	0	0	0	0	0	0	0	4	0	0	0	252	0.71	0.08	0.143	664	0.96	0.82	0.883
waterloo.high_low.grp33	0	0	0	0	0	0	0	0	0	0	0	0	236	0.77	0.96	0.853	574	0.87	0.69	0.77
waterloo.high_low.grp34	0	0	0	0	0	0	0	0	52	0.21	0.1	0.15	282	0.52	0.2	0.287	576	0.68	0.68	0.676
	87																			

表 8 からもまず個々のドキュメントについて見ると目標とするリンク回復精度を下げれば識別精度が上がるという関係が見て取れ，仮説５の正しさを示しているのだが，さらに識別精度に対する感度のドキュメント毎の多様性が見て取れる．T とするリンク回復精度を適合率 > 0.5，再現率 > 0.7 という大変厳しい値に設定した場合でも，唯一 EasyClinic_ID_CC のみが高い識別精度で識別が出来ているのだが，同様にこのリンク回復精度でも T となる試行を持つ EasyClinic_ID_TC，EasyClinic_UC_CC，EasyClinic_UC_TC，waterloo.high_low.grp18，waterloo.high_low.grp21 などでは識別は全くできていない．

表 8 で，リンク回復の精度を下げていくと識別の精度が上がっていく傾向が見て取れる．T とするリンク回復の精度を適合率 > 0.05 ，再現率 > 0.3 まで下げると，黄色で強調した SMOS と WV_CCHIT の二つを除きほとんどのドキュメント対で満足な識別精度を得られていることがわかる．

SMOS と WV_CCHIT について識別の精度が向上しない理由は，プロジェクトの特殊な特徴が文書の特徴量として表現しきれていなかったことによるものと考える．表 9 で，各ドキュメント対の平均文書数の一覧を挙げる

表 9 各ドキュメントのファイル数と平均リンク数

Table 9 The number of files and the average number of links each documents pair.

プロジェクト名及びドキュメント名	ファイル数	平均リンク数
EAnci_UC_CC	140,55	3.1
EasyClinic_CC_TC	47,63	4.34
EasyClinic_ID_CC	20, 47	3.45
EasyClinic_ID_TC	20, 63	4.15
EasyClinic_ID_UC	20, 30	1.3
EasyClinic_TC_CC	63, 47	3.24
EasyClinic_UC_CC	30, 47	3.1
EasyClinic_UC_ID	30, 20	0.87
EasyClinic_UC_TC	30, 63	2.1
Gantt_high_low	17,69	4
SMOS_UC_CC	67, 100	15.58
WV_CCHIT_Requirements_Regulatory_code	116, 1064	5.06
waterloo.high_low.grp01	58, 26	0.52
waterloo.high_low.grp02	42, 13	1.24
waterloo.high_low.grp03	70, 28	1.34
waterloo.high_low.grp05	54, 30	0.87
waterloo.high_low.grp06	39, 21	1.41
waterloo.high_low.grp08	85, 22	1.08
waterloo.high_low.grp09	30, 19	1.77
waterloo.high_low.grp10	76, 8	0.91
waterloo.high_low.grp11	79, 9	0.89
waterloo.high_low.grp13	43, 8	0.72
waterloo.high_low.grp14	46, 5	0.72
waterloo.high_low.grp15	69, 27	1.35
waterloo.high_low.grp17	57, 7	0.89
waterloo.high_low.grp18	53, 8	0.66
waterloo.high_low.grp19	61, 15	2.03
waterloo.high_low.grp20	93, 14	1.49
waterloo.high_low.grp21	36, 26	1.14
waterloo.high_low.grp23	32, 20	1.06
waterloo.high_low.grp24	51, 29	1.1
waterloo.high_low.grp30	48, 20	0.73
waterloo.high_low.grp32	86, 21	1.57
waterloo.high_low.grp33	65, 11	0.94
waterloo.high_low.grp34	28, 16	0.64

表 9 のとおり SMOS 以外のドキュメント対の平均リンク数(元文書 1 個あたりのリンク文書数）が 1〜3 程度なのに対して SMOS は平均リンク数が 15 と多い．そのため，リンク有と判断する類似度に対するランクの適切な値が他のプロジェクトと異なるため,SMOS 以外でマイニングした手法を適用しても誤選択になることが原因の可能性があると当初考えた．実際，調べて見ると SMOS の場合，リンクの判断基準にランクを採用するオプションで T となるものはなかった．他にはそのようなプロジェクトはないので，訓練データの不足が原因ではないかと考える．平均リンク数は正解が分っていて始めて知ることができる値なので，文書の特徴量としてマイニングの入力に使えず，そもそもドキュメントを作成する際にどの程度のリンクが存在するべきか，などについては標準もないため，平均リンク数は本質的に属人的，属プロジェクト的な特徴量となる．この例から，提案手法によって作成した識別器はプロジェクトを跨って普遍的には利用できない場合があることを示していると考える.尚,平均リンク数を予想値として特徴量に追加した際の識別の精度の変化等，今回は十分に検証できなかったが今後も検証していきたい.

同様に WV-CCHIT はドキュメントペアを構成するファイル数が 116 個，1064 個と他と比較して極端に多い．このため，同様の訓練データを得ることが出来ず，識別の精度が下がったものと考えられる.

上での考察は SMOS と WV-CCHIT の識別の精度が上がらなかった直接的な理由であると考えている．もう一つ間接的な理由として表 8 が示す通りこの二つはリンク回復の精度と，試行が T となった数の関係が他のドキュメント対とずれてる．SMOS, WV-CCHIT の両者はリンク回復の精度が適合率 ＞0.05，再現率 ＞0.3 の時，各々137 個と 136 個であるのだが，他のドキュメント対では T とする試行の数が百数十程度になるリンク回復の精度は適合率 >0.2，再現率 ＞0.6 近辺に多くあつまっていることがわかる．実験 3 では同じリンク回復精度による他のドキュメント対での試行結果を訓練データとしているため，SMOS と WC-CCHIT では訓練データと対象データで，全データに占める T となる試行のパーセンテージがそもそも異なっている事も識別の精度を下げている一つの原因になっていると考える.

この事の影響を確認するため，複数のリンク回復精度による複数のドキュメント対の試行の中から，T となった試行の数が百数十に近かった試行を選び，それら単独およびそれらの合計を訓練データとしてマイニングをおこない， 対象データには SMOS と WV-CHIT の試行で T となった試行数が同様（各々137 個，138 個）であったリンク回復精度である適合率 ＞ 0.05，再現率 ＞ 0.3 での試行をマイニングを行い精度を評価した．SMOS についての結果を表 10 に，WV-CCHIT の結果を表 11 に記す.

表 10 T となる試行の数の近い他プロジェクトの試行を訓練データとした SMOS の識別精度

Table 10 The identification performance of the SMOS using other projects which have similar number of T as the training data.

	プロジェクト	リンク回復の適合率, 再現率	試行の T の数	識別の適合率	識別の再現率	識別の F 値
1. Original result of SMOS		05-30	137	0.145	0.234	0.19
2. Use specific project as the teacher data.	EAnci_UC_CC	05-40	187	0.687	0.832	0.752
	EasyClinic_ID_CC	30-60	121	0.006	0.007	0.007
	EasyClinic_ID_CC	20-70	164	0.104	0.226	0.143
	EasyClinic_ID_TC	20-60	146	0.065	0.102	0.08
	EasyClinic_ID_UC	20-60	110	0.007	0.007	0.007
	EasyClinic_TC_CC	20-60	157	0	0	0
	EasyClinic_TC_CC	20-70	104	0	0	0
	EasyClinic_UC_CC	20-60	164	0	0	0
	EasyClinic_UC_CC	20-70	128	0	0	0
	waterloo.high_low.grp01	10-50	136	0	0	0
	waterloo.high_low.grp09	10-50	164	0.466	0.299	0.364
	waterloo.high_low.grp09	10-40	196	0.51	0.38	0.435
	waterloo.high_low.grp14	20-60	118	0.346	0.066	0.11
	waterloo.high_low.grp18	20-70	141	0.462	0.131	0.205
	waterloo.high_low.grp18	20-60	189	0.53	0.387	0.447
	waterloo.high_low.grp23	20-60	110	0.103	0.051	0.068
	waterloo.high_low.grp24	10-50	154	0.055	0.051	0.053
	WV_CCHIT_Requirements_Regulatory_code	05-30	136	0.224	0.394	0.286
3. Combination of 2. (without SOMS, WV-CCHIT)			2489	0.613	0.832	0.706

表 11 T となる試行の数の近い他プロジェクトの試行を訓練データとした SMOS の識別精度
Table 11 The identification performance of the SMOS using other projects which have similar number of T as the training data.

	プロジェクト	リンク回復の適合率, 再現率	試行の T の数	識別の適合率	識別の再現率	識別の F 値
1. Original result of WV-CCHIT		05-30	136	0.111	0.176	0.136
2. Use specific project as the teacher data.	EAnci_UC_CC	05-40	187	0.182	0.324	0.233
	EasyClinic_ID_CC	30-60	121	0.339	0.309	0.323
	EasyClinic_ID_CC	20-70	164	0.402	0.485	0.44
	EasyClinic_ID_TC	20-60	146	0.42	0.5	0.456
	EasyClinic_ID_UC	20-60	110	0.182	0.176	0.179
	EasyClinic_TC_CC	20-60	157	0.126	0.096	0.109
	EasyClinic_TC_CC	20-70	104	0.183	0.096	0.126
	EasyClinic_UC_CC	20-60	164	0.286	0.353	0.316
	EasyClinic_UC_CC	20-70	128	0.274	0.25	0.262
	waterloo.high_low.grp01	10-50	136	0.286	0.206	0.239
	waterloo.high_low.grp09	10-50	164	0.01	0.015	0.012
	waterloo.high_low.grp09	10-40	196	0.009	0.015	0.011
	waterloo.high_low.grp14	20-60	118	0	0	0
	waterloo.high_low.grp18	20-70	141	0.375	0.088	0.143
	waterloo.high_low.grp18	20-60	189	0.311	0.206	0.248
	waterloo.high_low.grp23	20-60	110	0.359	0.206	0.262
	waterloo.high_low.grp24	10-50	154	0	0	0
	SMOS_UC_CC	05-30	137	0.149	0.191	0.168
3. Combination of 2. (without SOMS, WV-CCHIT)			2489	0.179	0.206	0.192

表 10，表 11 で最も識別の精度の良かった結果を黄色で示した．SMOS では，リンク回復の精度として適合率 > 0.05，再現率 > 0.4 の EAnci_UC_CC を訓練データとして使用した際に，識別の精度として適合率 = 0.687，再現率 = 0.832，F 値 = 0.752 という大変よい精度での識別が出来ており，WV-CCHIT でもリンク回復の精度として適合率 > 0.2，再現率 > 0.7 の EasyClinic_ID_CC を用いることで適合率 = 0.402，再現率 = 0.485，F 値 = 0.44 という悪くない精度で識別ができている．このことから当初予想したとおり SMOS，

WV-CCHIT の両者の識別の精度が低かった理由がそもそもの訓練データに対する特徴量の特殊性にあったものと考える.

表 7 に戻って waterloo の grp05 や grp20 のように識別のアルゴリズムによって識別の精度が大きく変わっている事例が見つかっている. この原因についても今後の考察が必要と考えている.

4.5 結言

本章では２章で提案した非対称類似度及び３章で提案した提案手法についてそれが必要でありかつ有益である旨の仮説をたて, 参照ドキュメントを用いた実験を通じて仮説検証を行った.

次章では, 本論文の妥当性への脅威について考察を行う.

第 5 章
妥当性への脅威

5. 妥当性への脅威

5.1 内的脅威

5.1.1 仮説 1

仮説 1「すべてのドキュメント対について良いリンク回復精度を与える一意な手法は存在しない.」はいくつかの研究 [5, 6, 13, 16, 30]でも述べられている内容ではあるが，本論文でも実験 1 の結果について各手法毎の平均 f-measure 値の比較した表 3 で各々の手法を適用の可否のどちらかが常に良いリンク回復精度を与えるような事はないことを確認し，さらに各手法オプションの組み合わせについてリンク回復精度の上位 3 位への出現頻度がほぼ全ての組み合わせに現われることからも確認した．この結果を定性的に考察すれば，そもそも情報検索手法は文書を単語数で定量的に表現した際の類似度の大小からリンクを判断する手法であり，個々のオプションは本質的に文書の特徴を仮定し，単語の共起性を上げる目的で用意されている．例えばオントロジーも，類義語が多数存在するという仮定の元に，LSI も複数の単語によって構成されるトピックが存在するという仮定の元での効果を期待するオプションである．しかし，そのような仮定が成立しないドキュメントについては効果が期待されないばかりか逆効果になるであろう．定性的にはそのように導かれる仮説 1 が実験 1 によって再確認されたものと考える.

5.1.2 仮説 2

仮説 2「手法オプションの適否はドキュメントの特徴量に依存する.」は，実験 2 を J48 で行う際に作成された J48 の決定木の条件が文書の特徴量から構成されていたこと，及びそのように作成された決定木により適切な手法オプションを有意な精度で選択することができることから確認を行った．単に決定木が特徴量をパラメタとして作成されただけではなく，それにより有意な精度で適切な手法オプションを選択できたことにより，つまりドキュメントの特徴量を条件式として適切な手法オプションを選択できることにより，仮説 2 が述べるとおり手法オプションの適否はドキュメントの特徴量に依存していると考えることは妥当であろう.

このように作成された決定木によって有意な精度で識別ができたことは事実であるが，この実験で用意した特徴量がもっとも良い，つまり識別の精度にたいして最も直接的に作用するものであるとは言えない．つまり，識別精度に対して主要因となる特徴量は他にあった可能性は高い．ただ，この実験では識別精度の主要因として働く特徴量を特定していないとはいえ，手法オプションの適否がドキュメントの特徴量に依存している事実を示すことはできていると考えることは妥当と考える.

5.1.3 仮説 3

仮説 3「リンクの種別によっては非対称類似度はリンク回復精度の向上に有効である」について，実験 1 の結果について各手法毎の平均 f-measure 値の比較した表 3 で，いくつかの結果については対象類似度を用いた物より非対称類似度を用いたもののほうが識別の精度が高かった事実より確認された.

そもそも非対称類似度を導入した理由はドキュメント間の構造に非対称性が仮定され，一方のドキュメントのみに現れる単語の存在が仮定される場合にその単語が他方に現れないことによる類似度の低下を無視するためであった．そこで，非対称類似度が有効であったドキュメントがそのような非対称性を有していたことが確認できれば非対称類似度が有効となる必要十分条件が確認できた訳なのであるが，そこまでは確認できなかった.

5.1.4 仮説 4

仮説 4「本提案手法によって，妥当なリンク回復精度を与える手法オプションを有意な識別精度で識別することができる.」は実験 2 の結果，T が 1%しか存在しない適合率>0.3, 再現率 >0.7 という回復の精度基準についても 0.87 という高い適合率で識別をおこなうことが出来たことにより仮説 4 で述べたとおり，回復の基準精度のような妥当なリンク回復精度を得る手法を，0.8 以上の識別精度で識別することが判明した．また，当該手法の一般性を確認するために行った実験 3 の結果，上記回復の基準精度でのリンク回復精度を得ることはできなかったが，提案手法のとおり T とする回復の精度を順次下げていくことにより目標とする識別精度を得られることがわかった.

5.1.5 仮説 5

仮説 5「目標とするリンク回復精度と識別精度には関連がある.」は実験 3 の表 7,表 8 の結果が，リンク回復の精度を下げると識別精度が上がる関係を示している．しかし，そのような関係があることを実験が示唆してはいるが，具体的な関係式を導くには至っていない.

5.2 外的脅威

実験は複数のプロジェクトに渡り複数のドキュメント種別（要求記述，設計記述，テスト仕様，実装コード）の組み合わせで行っており，特定のプロジェクトの特性に支配されない条件で行われている.

実験は CoEST で提供される参照データを用いて行った．この参照データは提供されている段階で適当な単位に分割されている．この，CoEST で事前に用意された分割が適切であるかの評価はできていない．２章で述べたように，実プロジェクトでトレーサビリティリンクの回復を行う場合，ソースコードを含む各種ドキュメントをどのように分割するかという問題は未解決な大きな問題である．これについては今後もさらに研究を行いたい.

本論文では有意な識別精度で識別が可能である事例を実験で示したが，この識別精度が保証できる条件等の一般性についてさらに研究を行いたい．

実験 2 と実験 3 での識別精度の違いや，実験 3 の SMOS や WV-CCHIT のように識別精度の低い対象であっても訓練データをうまく選ぶことで識別精度をあげることができた結果などが示すように，本論文での提案手法による識別精度は検査プロジェクトと訓練データとの相性で変わってくる．そのため，相性が悪い場合は本論文で主張したような有意な精度での識別ができない場合がある．特に本論文での提案手法の inter project での利用を考えた場合，正解リンクの密度の違いなど，プロジェクトに特殊な事情を特徴量と出来ない事，及びマイナーな傾向を訓練データに取り込めなくなる限界も示している．極端なことを言えば，プロジェクトの担当者がリンクを意識せずに仕様書の段階的抽象化を行った場合や，文書のコピーペーストを繰り返し，一部のみ変更した文書を大量に作成する等，プロセスが一般的でなかったプロジェクトに対して，標準的なプロジェクトで作成した訓練データは有効に使えないものと思われる．逆に intra project での利用を考えた場合，例えばプロジェクトの初期のデータを訓練データとして進化途中のソフトウェアアーティファクトのトレーサビリティリンクの復元を行う事を考える場合，プロジェクトの特殊な事情やマイナーな傾向が進化を通じてあまり変化していない事が期待できるため，訓練データと検査プロジェクトの相性が良く，よい精度で識別ができる事が期待できる．

今回は手法の有効性を示す実例の提示の目的での実験であったため，手法のオプションや文書の特徴量を網羅的に考慮することを行っていないが，提案手法を現場で実践する場合にはそれらは当然，網羅的でなければならない．手法については数々紹介されているのだが，特徴量についての研究はまだ少なく，今後おこなっていきたいと考えている．例えば，前述のようにコピーペーストが過半数を占めるようなドキュメント対のような開発現場ではありがちな現実を特徴量として表現する方法，及びそれに有効な手法の組み合わせ等に興味が持たれる．

実験３の結果はプロジェクトによって識別精度に差があることを示していた．さらに識別のアルゴリズムによって変わるものと変わらないものの存在も示していた．では，よい識別のアルゴリズムをどのようにマイニングしたらよいかという話になるのだが，今回はそれについての考察はできていない．

今回は CoEST のデータを使ったが,すべてオープンソースのプロジェクトデータであり，また規模的にもそれほど大きくない．またリンク単位たるドキュメントの大きさ，要求文書等のリンク単位への分割，日本語の特異性などの検討が必要となる．

5.3 手法の適用の妥当性についての考察

手法の適用性の一般性についての外敵脅威の考察にも関連する話ではあるが，手法自体の実開発現場での位置づけに関する話題として個別に検討しておくほうが実用上の利便性はよいため，掲題の話題についてここで考察する．

実験３により，目標とするリンク回復精度と目標とする識別精度との間に関係があることがわかったが，どのような関係になっているのか，また，それが識別の精度の値の主要因であるのか否かの解析はまだ出来ていない．

リンク回復を有意な精度でおこなうためにはそもそもリンクの存在する文書間とそうでない文書間で類似度に有意な差が存在しなければならない．だとすると，どのような手法オプションを用いても有意な差を見出せない文書ではそもそもリンク回復をよい精度で行うことは難しい．従って文書の良否も含めて識別を行う必要がある．

また，外的脅威の中でも述べたが当該手法はユニークな特徴量を持つプロジェクトについては訓練データが有効に機能しないため精度が落ちることが実験 3 で改めて示された．このことより当手法の適用に当たっては，特徴量が似ているプロジェクト，例えば同一プロジェクトの開発の初期を訓練データとするような運用が好ましいと考える．

第 6 章
関連研究

6. 関連研究

トレーサビリティリンク回復はソフトウェア保守において重要な作業であり，多くの研究が行われている[3, 9, 4, 22]．おもなる用途は変更影響範囲解析であるが，Lucia[19]ではソースコード要求文書のトレーサビリティを考える事で，ソースコードの識別子を改善する手法を提案している．またほとんどの手法はゼロからの回復を考えているがSukanya[25]では進化に伴い，トレーサビリティリンクをどのように更新するかについて考察している．

情報検索以外の手法としては，テスト網羅性に基づく Widerseiner[27]の研究や，デザインパターンを利用した Cleland[9]の研究等がある．

情報検索手法はドキュメント間の語彙の共起性に基づくが，そのステップは大きく次の３つに分けられる．

- ステップ１　対象から適切な単語を抽出し，単語・ドキュメント行列を作る
- ステップ２　単語・ドキュメント行列に基づき，ドキュメント間の類似度を求める
- ステップ３　類似度に基づき，ドキュメント対の間にリンクがあるかどうかを判断する

ステップ１ではステミングやストップワードの扱い等が問題となる．本研究でも仮説 1 として再検証したとおり，それらについて万能の手法はなくドキュメントにより適切な手法が異なることを多くの研究が示している．Mahmoud[20]ではソースコードからの単語の抽出に対するオプションの評価を行っており，コメントの利用は有効である，ソースコードが対象の場合にはステミングは有効ではない，としている．Capobianco[5]では抽出する単語の品詞に着目し，名詞が有意であるとしている．Xiaofan[8, 7]においても単語の選択手法の改善を提案している．また Wiese[28]はトレーサビリティリンク回復が主目的ではないが，５つのステミングのアルゴリズムの性能を評価し，用途による使い分けが必要としている．また単語の共起性という点から同義語，類義語の扱いも問題となるとしている．Settimi[26]は WordNet を使ったシソーラスの利用法を試みているが，ほとんど効果はないとしている．一方 Zhang[29]では対象ドキュメントから抽出したオントロジーを使う事で若干の精度改善を実現している．Hayes[11]でもシソーラスを利用しているがドキュメントの種別により，その効果は異なるとしている．シソーラスによってオントロジーを扱う方法に代わって，ドキュメント全体の単語の出現状況から同義語，類義語の処理を行おうという手法に LSI(Marcus[21], Lucia[19], Lormans[15]), pLSI, LDA(Asuncion[4])があり，多くの研究でその利用の効果が研究されている．Lormans[15]では LSI を使った類似度に基づいてリンク対を判定する５つの手法の比較を行っている．得失があること，またドキュメントの種別によって異なるとしている．LSI 手法では次元の縮小という操作を行うが，どの程度の縮小が有効かについては Lucia[17]等で考察されている．本研究では縮小率が 50%の LSI のみを選択候補の手法オプションとして導入したが Lucia の考察にあるとおり

縮小率の値も文書の特徴量により変わることが考えられ，それを含めたオプションマイニングのし識別精度の検証についても今後の研究課題と考える.

ステップ２の類似度の計算では確率的に計算するJSM, PMやベクトル空間上の類似度(コサイン類似度）を利用するVSM等があり，その比較実験が行われている. Antoniol[3]では確率モデルとベクトル空間モデルの比較を行っているが，有意な差はないとしている. Oliveto[23]ではJS, VSM, LSI, LDAの４つの手法を比較評価している.結果としてJS, VSM, LSI はほぼ同じであるが，LDA はやや劣る，しかしLDAは他では見つけられないリンクを見つけられるとしている. Malcom[10]でもVSM, JSM, そして新たに提案するRTMという手法の直交性に着目し，それらを組み合わせた手法を提案している. また対象言語（英語とイタリア語）による相違についても言及している. Capobianco[6]はB-spline という手法を提案し，VSM やJSM と比較し，VSM より優れており，JSM とは同等か，優れているとしている. Abadi[2]ではVSM, LSI 等を比較し，LSIは効果はないとしている.

確率モデルでは類似度の方向性を考慮しているが,ベクトルモデルでは考慮していない. 本論文で提案した非対称コサイン類似度を考えている研究はない.

ステップ３では計算された類似度に基づいてリンク対の識別を行う.Lormans[15]では次の５つの判定法を比較評価している.

- cut point 上位N位を候補とする
- cut percentage 上位N%を候補とする
- constant threshold 類似度がC以上のものを候補とする
- variable threshold 全体の類似度の分布から決める値εよりも大きな類似度を持つものを候補とする.
- scale threshold 類似度の閾値をC＊最大類似度で決める. 但しCは0から1の間のある値

これらの手法の優位性はドキュメントの種別によって異なっており，Settimi[26]やLucia[18]ではいくつかの正解サンプルを用意し,それを使った評価から閾値を求める手法を提案している.Lucia[17]でも最適な閾値はドキュメントの種別により異なること,また同じ種類のドキュメント間では，異なった種類のドキュメント間よりも平均類似度が大きい，等からドキュメントの特性に応じた閾値の選択の必要性を述べている. このためにサンプルを用いて事前に評価して閾値を決める手法も提案している.

このように情報検索手法といっても適用のオプションは多様であり，その適切な選択は簡単ではない. Capobianco[6, 5], Xuchang[30], Lucia[16], Imtiaz[13]等でドキュメントの種別とオプションの有効性は関係があるとしている. しかしながら適切な選択をどのようにして行えばよいかについてほとんど研究されていない.Xuchang[30]では精度の改善のためのいくつかの単語の選択法を提案している. 必ずしもすべてのドキュメントに対して有効ではないということで，提案手法の有効性を評価するためのメトリクスも合わせて提案している.

精度の評価は再現率，適合率，F 値等を使って行われるが，手法の比較においては再現率/適合率曲線を使っているものが多い.しかしながら絶対的な基準を述べているものは少ない. Hayes[11]では企業での経験から得られた値として再現率が 0.60 から 0.69，適合率が 0.2 から 0.29 を受容可能としている. しかし手法の適否の基準ではないとしている. 本研究においてもこの Hayes の基準を参考に実験の目標識別精度を再現率 0.7，適合率 0.3 として実験を行った.Settimi[26]では対象となるドキュメントの数が精度に大きな影響を与えるとしている.

第７章
むすび

7. むすび

大規模なソフトウェアシステムを保守可能に保つためにはソフトウェアのソースコード，ドキュメント及びテストケースなどの間でのトレーサビリティリンクが確保されていることが重要であり，トレーサビリティリンクを記録し管理するための商用ツールも提供されている．しかし，ソフトウェア進化の課程で失われることがよくある．トレーサビリティリンクの回復には本来は人手を介してドキュメントの査読とリンクの抽出を行わなければならないのであるが，簡易な方法として幾つかの自動回復手法が提案されており，なかでも情報検索手法を用いた手法がその一般性から注目されてきている．情報検索手法を用いたトレーサビリティリンク回復については数々の手法が提案されており，それら手法の選択によるリンク回復の精度の向上には対象ドキュメントとの相性があることが言われてきているが，実際の現場で実践するにあたり，どのような手法がその現場に最も適切なのかを知る手段がないことに対して問題意識を持ち，本論文では Zhimin[12]が error prone モジュール予測のための適用したオプションマイニングの手法をトレーサビリティリンク回復のオプションマイニングに適用する方法を提案した．合わせてトレーサビリティリンクの回復の対象となるドキュメントについてその構造と意味を考えると，リンクを考察する対象となるドキュメント対は必ずしも対称ではなく，その場合，一方に現れる単語がもう一方に現れることにはリンクとしての意味があっても，その逆方向についての単語の共起性にはリンクとしての意味を持たないような構造があることを述べ，そのような場合に従来の対称類似度よりよいリンク回復精度を与えることが期待される非対称類似度を提案した．さらに，提案手法によらずに，一般にどのようなドキュメント対に対しても常によい精度をあたえるような手法オプションは存在せず，手法オプションの適否はドキュメントの特徴量に依存すること，非対称類似度は対称類似度同様に有効な，つまりドキュメント対によってはリンク回復の精度を改善する手法であること，並びに提案手法によるマイニングにより，識別の精度保証の可能性があること，リンク回復の精度は識別の精度と関係があり，どのように高い識別の精度を期待しても常によいリンク回復の精度がえられるわけではないという５つの仮説を設け，それらが正しいことを実験で検証した．実験２はある意味理想的な条件での検証であったため，現実の環境により近いと思われる実験３でさらに保証できる精度を確認し，基準とするリンク回復精度が妥協できる環境であれば有意な識別の精度を保証できることを確認した．

以上より．本論文での提案手法は実際のソフトウェア開発の現場でトレーサビリティリンク回復を行う上で有効な手段であるものと考える．

さらに実験２の理想的な状況で回復の精度基準が十分保証できていた事より，本論文の提案手法は intra project での実践で特に有効である可能性がある．例えば開発の初期にはリンクが確保できていたが,進化を通じてリンクが失われていった場合のリンク回復 の

ように，リンクが明確だった初期のドキュメント及び正解値を訓練データとして，リンクが失われた後のドキュメントでのリンク回復を実施する際，ドキュメントの特徴量やプロジェクトの特性の変化は少ないと期待できるため，高い精度での識別が期待でき，更なる実験で検証を行っていきたい.

また，実験３の考察を通じて文書の特徴量だけでなくプロジェクトの特性，具体的にはプロジェクトによって異なるドキュメント間のリンクの密度なども手法の選択の適否に影響をあたえる可能性がみつかった．そのような場合，特性が大きく異なるプロジェクトのドキュメントを訓練データとしてマイニングを行うと識別の精度が極めて低くなることがわかったが，そのような場合においても訓練データのリンク識別精度を調整し，対象データと真となるオプションのパーセンテージが近くなるように選択し，それを訓練データとすると識別の精度の向上がみられることを確認した．このように，プロジェクトの特性が異なるドキュメントを訓練データとしてマイニングを行う場合の補正方法等，今後の研究で充実させていきたい.

しかしながら手法のオプションについてもLDA等実現していないものが多くあり，また特徴量についても十分とは言えない．大きな問題としてベースとなるドキュメントをどのようにしてリンク対象となるドキュメント単位に分けるかという問題およびその粒度と手法の適否の問題がある．更に，実プロジェクトではトレーサビリティリンクが失われている状態であっても流用できる何らかのリンク関係が存在することが多い.具体的に言うと，同時に修正されてリポジトリに同時にコミットされたドキュメントの履歴，複数のチームからなる開発プロジェクトにおけるドキュメント群の保守担当チームの情報．一つのメールに同時に添付されていたドキュメントなど，流用でき，流用することで高い精度でのリンクの回復を可能にする情報が実プロジェクトには多く存在するものと考えられるのだが，そのような情報の流用方法，及びそれらを流用した場合のマイニングやリンク回復の手法選択への影響などについての考慮ができていない．公開されている参照ドキュメントデータのなかで今回，実験を行っていないデータ集合も多くあるので，これらについても実験をおこない，さらに有効性を確認したい．そして，実プロジェクトでの実践を通じてさらに研究を発展させていきたいと考える.

実際の開発現場において，トレーサビリティリンクを喪失したソフトウェアシステムの保守で最もよく使われる手段は追跡したいテーマをキーワードとする全文検索であり，現実にはある程度機能している．ベクトル空間モデルでのドキュメントの類似度に基づくトレーサビリティリンク回復は，そのようなキーワードを利用せずに一般的な語の共起性に基づいてトレーサビリティリンクの回復を試みる方法なのであるが，では特定のキーワード群を人手で抽出しておき，そのキーワードのみの共起をみる類似度を用意するとどのようになるのか等の半自動的な方法も今後のテーマとして取り組んでいきたい.

手法オプションの選択の適否について最も影響を与える文書の特徴量はなになのかということについて特定することを本論文では目標とはせず，かわりに何らかの特徴量から適

切な手法オプションの選択をマイニングすることができるということを示すことに努めた．このようなマイニングが可能であるという事は，結局は個々の手法オプションについてその選択の適否に深く関係する特徴量が存在することをしめしているものと考える .そこで，その特徴量を特定することができればマイニングという方法によらず決定論的に正しい手法オプションを選択することができる事になる．そのように決定論的な研究についても今後は行っていきたいと考えている．

最後になるが，今後もエンジニアとし現場でのソフトウェア開発に取り組んでいくにあたり，提案手法の現場での実証を続けることでその課題の抽出及び改善に取り組んでいきたいと考える．

文　　献

[1] Center of excellence for software traceability. http://www.coest.org/.

[2] Aharon Abadi, Mordechai Nisenson, and Yahalomit Simionovici. A Traceability Technique for Specifications. In International Conference on Program Comprehension, 103-112, 2008.

[3] Giuliano Antoniol, Gerald Canfora, Gerardo Casazza, Andrea De Lucia, and Ettore Merlo. Recovery of traceability links between software documentation and source code. IEEE Tr. on S.E, 28(2):970-983, 2002.

[4] Hazeline U. Asuncion, Arthur U. Asuncion, and Richard N. Taylor. Software Traceability with Topic Modeling. In International Conference on Software Engineering, 95-104. ACM, 2010.

[5] Giovanni Capobianco, Andrea De Lucia, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. On the Role of the Nouns in IR-based Traceability Recovery. In International Conference on Program Comprehension, 148-157, 2009.

[6] Giovanni Capobianco, Andrea De Lucia, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. Traceability Recovery using Numerical Analysis. In Working Conference on Reverse Engineering, 195-204, 2009.

[7] Xiaofan Chen and John Grundy. Improving Automated Documentation to Code Traceability by Combining Retrieval Techniques. In International Conference on Automated Software Engineering, 223-232. IEEE, 2011.

[8] Xiaofan Chen, John Hosking, and John Grundy. A Combination Approach for Enhancing Automated Traceability. In International Conference on Software Engineering, 912-915, 2011.

[9] Jane Cleland-Huang and David Schmelzer. Dynamically Tracing Non-Functional Requirements through Design Pattern Invariants. In International Workshop on Traceability in Emerging Forms of Software Engineering. IEEE, 2003.

[10] Malcom Gethers, Rocco Olivetot, Denys Poshyvanyk, and Andrea De Lucia. On integrating orthogonal information retrieval methods to improve traceability recovery. In International Conference on Software Maintenance, 133-142, 2011.

[11] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. IEEE Tr. on S.E , 32(1):4-19, 2006.

[12] Zhimin HE, Fengdi Shu, Ye Yang, Mingshu Li, and QingWang. An investigation on the feasibility of cross-project defect prediction. Autom Softw Eng, 19(2):167-199, 2012.

[13] Salma Imtiaz, Naveed Ikram, and Saima Imtiaz. Impact Analysis from Multiple Perspectives: Evaluation of Traceability Techniques. In International Conference on Software Engineering Advances, 457-464. IEEE, 2008.
[14] Waraporn Jirapanthong and Andrea Zisman. Supporting Product Line Development through Traceability. In Asia-Pacific Software Engineering Conferenc, 2005.
[15] Marco Lormans and Arie van Deursen. Can LSI help Reconstructing Requirements Traceability in Design and Test? In European Conference on Software Maintenance and Reengineering. IEEE, 2006.
[16] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, and Genoveffa Tortora. Enhancing an Artefact Management System with Traceability Recovery Features. In International Conference on Software Maintenance. IEEE, 2004.
[17] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, and Genoveffa Tortora. Recovering Traceability Links in Software Artifact Management Systems using Information Retrieval Methods. ACM Transactions on Software Engineering and Methodology, 16(4), 2007.
[18] Andrea De Lucia, Rocco Oliveto, and Paola Sgueglia. Incremental Approach and User Feedbacks: a Silver Bullet for Traceability Recovery? In International Conference on Software Maintenance. IEEE, 2006.
[19] Andrea De Lucia, Massimiliano Di Penta, and Rocco Oliveto. Improving Source Code Lexicon via Traceability and Information Retrieval. IEEE Tr. on S.E , 37(2):205-227, 2011.
[20] Anas Mahmoud and Nan Niu. Source Code Indexing for Automated Tracing. In International Workshop on Traceability in Emerging Forms of Software Engineering, 2011.
[21] Andrian Marcus, Jonathan I. Maletic, and Andrey Sergeyev. ECOVERY OF TRACEABILITY LINKS BETWEEN SOFTWARE DOCUMENTATION AND SOURCE CODE. Journal of Software Engineering and Knowledge Engineering, 15(5):811-836, 2005.
[22] Collin McMillan, Denys Poshyvanyk, and Meghan Revelle. Combining Textual and Structural Analysis of Software Artifacts for Traceability Link Recovery. In International Workshop on Traceability in Emerging Forms of Software Engineering, 41-48. IEEE, 2009.
[23] Rocco Oliveto, Malcom Gethersy, Denys Poshyvanyky, and Andrea De Lucia. On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery. In International Conference on Program Comprehension, 68-71, 2010.
[24] Balasubramaniam Ramesh and Matthias Jarke. Toward Reference Models for Requirements Traceability. IEEE Tr. on S.E, 27(1):58-93, 2001.

[25] Sukanya Ratanotayanon, Susan Elliott Sim, and Derek J. Raycraft. Cross-Artifact Traceability Using Lightweight Links. In International Workshop on Traceability in Emerging Forms of Software Engineering, 57-64. IEEE, 2009.
[26] Raffaella Settimi, Jane Cleland-Huang, Oussama Ben Khadra, Jigar Mody, Wiktor Lukasik, and Chris DePalma. Supporting Software Evolution through Dynamically Retrieving Traces to UML Artifacts. In International Workshop on Principles on Software Evolution. IEEE, 2004.
[27] Christian Wiederseiner, Vahid Garousi, and Michael Smith. Tool Support for Automated Traceability of Test/Code Artifacts in Embedded Software Systems. In International Conference on Trust, Security and Privacy in Computing and Communications, 1109-1117. IEEE, 2011.
[28] Andrew Wiese, Valerie Ho, and Emily Hill. A Comparison of Stemmers on Source Code Identifiers for Software Search. In International Conference on Software Maintenance, 496-499, 2011.
[29] Yonggang Zhang, Rene' Witte, Juergen Rilling, and Volker Haarslev. An Ontological Approach for the Semantic Recovery of Traceability Links between Software Artifacts. IET Software, 2(3):185-203, 2008.
[30] Xuchang Zou, Raffaella Settimi, and Jane Cleland-Huang. Improving automated requirements trace retrieval: a study of term-based enhancement methods. Emp. Soft. Eng., 15(2):119-146, 2010.
[31] The Porter Stemming Algorithm. http://tartarus.org/~martin/PorterStemmer/.
[32] IR linguistic utilities. http://ir.dcs.gla.ac.uk/resources/linguistic_utils.
[33] Rational DOORS. http://www-03.ibm.com/software/products/en/ratidoor
[34] CATIA Systems Engineering - Reqtify
http://www.3ds.com/products-services/catia/capabilities/systems-engineering/requirements-engineering/reqtify/
[35] ISO/IEC 14764:2006 Software Engineering -- Software Life Cycle Processes - Maintenance
[36] Hassine, Jameleddine, et al. Change impact analysis for requirement evolution using use case maps. In: Principles of Software Evolution, Eighth International Workshop on. IEEE, 2005. p. 81-90.
[37] Lindvall M., Sandahl K. Practical Implications of Traceability. Software Practice and Experience, 1996, 26.10: 1161-1180.
[38] Spanoudakis, George, and Andrea Zisman. Software traceability: a roadmap. Handbook of Software Engineering and Knowledge Engineering, 2005, 3: 395-428.
[39] Ramesh B., Jarke M. Towards Reference Models for Requirements Traceability. IEEE

Transactions in Software Engineering, 27(1), 58-93, 2001.
[40] Hazeline A., Arthur A., Richard T., Software traceability with topic modeling. ICSE '10 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 Pages 95-104
[41] Yonggang Z., René W., Juergen R., Volker H. Ontological approach for the semantic recovery of traceability links between software artefacts IET Software, Volume 2, Issue 3, June 2008, p. 185 – 203
[42] Deerwester, Scott C., et al. "Indexing by latent semantic analysis." JASIS 41.6 (1990): 391-407.
[43] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.

謝辞

本研究を遂行し学位論文をまとめるに当たり，多くの御支援と御指導を賜りました指導教官の海尻賢二教授に深く感謝しております．知的で温厚な先生からいつも優しく導いていただいたお陰で今日まで研究を続けることができた次第であり，時として厳しい御指導を頂き，私自身の至らなさを思い至った次第ですが今後の努力の糧になるものでございます．

研究室修士１年生の湯浅諒平君による協力に深く感謝致します．優秀な湯浅君のことなので，今後も次々と立派な成果をだされるものと期待しております．

最後に物心両面に渡り長い研究を支えてくれた妻に感謝します．

本研究を構成する研究

- 査読付き論文

  1. 情報検索手法に基づくトレーサビリティリンク回復のための手法オプションについてのマイニングの提案と評価
  電子情報通信学会　和文論文誌 D　Vol.J97-D,No.3,pp.414-426,Mar. 2014

- 国際会議発表論文

  1. IR based Traceability Link Recovery Method Mining.
  International Conference on Software Engineering Advances,pp 278-284 . IARIA, 2013.10

- 口頭発表

  1. ソフトウェア規模見積もりと進捗管理手法の提案
  情報処理学会研究報告 ソフトウェアエ学研究会報告. 2005(119):117-124 (2005)

  2. commit ログの新しい分類手法の提案と実験
  信学技報, vol. 112, no. 419, KBSE2012-66, pp. 47-52, 2013 年 1 月.

  3. Traceability link recovery のための method mining の提案
  信学技報, vol. 113, no. 71, KBSE2013-5, pp. 25-30, 2013 年 5 月.