修　士　論　文

題　　目　　実プロジェクトを対象にした
　　トレーサビリティリンク構築手法の提案

主任指導教員　　水野　修　教授

指導教員　　崔 恩瀞　助教

京都工芸繊維大学 大学院工芸科学研究科

情報工学専攻

学生番号　　18622007

氏　　名　　植村　佳治

令和２年２月10日提出

様式 2 号

学位論文内容の要旨（和文）

令和 2 年 2 月 10 日

京都工芸繊維大学
大学院工芸科学研究科長　殿

大学院工芸科学研究科 情報工学専攻
平成 30 年入学
学生番号　18622007
氏　名　植村　佳治　㊞

（主任指導教員　水野　修　㊞　）

本学学位規則第 4 条に基づき、下記のとおり学位論文内容の要旨を提出いたします。

1.　論文題目

実プロジェクトを対象にしたトレーサビリティリンク構築手法の提案

2.　論文内容の要旨（400 字程度）

ソフトウェア開発の各工程では，要求仕様書や設計書，ソースーコードなど様々な文書が作成される．開発中には要求変更がしばしば起こり，それに伴いソフトウェアの仕様も変更され得る．開発工程で作成された各種文書が相互に関連付いていない場合，全ての変更箇所を手動で探索することは容易ではない．仕様書やソースコードなどの修正箇所の特定，および実装と仕様の間の矛盾検出や一貫性検証を容易にするため，文書を関連付けることは重要である．

我々は，仕様書とソースファイルを入力として生成した文書ベクトルの類似度からトレーサビリティリンクを構築する手法を提案する．本研究では，仕様書とソースファイル を 1 つの文書とみなして，文書ベクトルを生成手法として Bow，TF-IDF，Doc2Vec を用いて，仕様書と JSP では TF-IDF 手法によって 76%の精度ではトレーサビリティリンクを構築することができた．JSP と Java では TF-IDF 手法によって 75%の精度ではトレーサビリティリンクを構築することができた．また，文書ベクトル生成の精度に文書の単語数が寄与するという知見を得た．

今後の課題として，他の文書ベクトル生成手法との精度の比較や他の実プロジェクトにも適用した場合の一般性の検証をなどが挙げられる．

Constructing Traceability Links between Specification and Source Code in Proprietary Projects

2020 18622007 *UEMURA Yoshiharu*

Abstract

In each process of software development, various documents, such as requirement specifications, design documents, and source codes, are created. During development, the requirements often change, and the software specifications can change accordingly. If the various documents created in the development process are not related to each other, it is not easy to search for all changes manually. It is essential to correlate documents in order to identify corrections such as specifications and source codes and to quickly detect inconsistencies and verify consistency between implementation and specifications.

We propose a method to construct a traceability link based on the similarity between a specification and a document vector generated from source files. In this research, the specification and source file are regarded as one document, and Bow, TF-IDF, and Doc2Vec are used as the generation method of the document vector. Traceability with 76% accuracy in the specification and JSP by the TF-IDF method. The link could be constructed. In JSP and Java, a traceability link could be constructed with 75% accuracy by the TF-IDF method. We also found that the number of words in a document contributed to the accuracy of the document vector generation.

Future tasks include comparing the accuracy with other document vector generation methods and verifying the generality when applied to other real projects.

目　次

1. 緒言

ソフトウェア開発の各工程では，要求仕様書や設計書，ソースーコードなど様々な文書が作成される．開発中には要求変更がしばしば起こり，それに伴いソフトウェアの仕様も変更され得る．開発工程で作成された各種文書が相互に関連付いていない場合，全ての変更箇所を手動で探索することは容易ではない．仕様書やソースコード などの修正箇所の特定，および実装と仕様の間の矛盾検出や一貫性検証を容易にするため，文書を関連付けることは重要である．しかし，昨今のソフトウェアの大規模化に伴って，仕様書やソースコード 間の関連が失われているプロジェクトが多数存在する [1]．ある文書に変更が生じたときに関連する他の文書やコードの各部分の修正をすること無く，片方の文書だけを修正してしまうことで， 文書間に矛盾が生じる可能性がある．

仕様書やソースコード などのソフトウェア開発過程で生産される文書間を紐づけた情報をトレーサビリティリンクと呼ぶ．ある文書に含まれる項目に対して対応する箇所を探すためには，他の全ての文書を探索しなければならないため，トレーサビリティリンクを構築する作業はコストが高い．また，トレーサビリティリンクを構築を支援するツール [2] や手法 [3–8] も提案されているが，これらには事前に手動のリンクやタグ付けが必要であったり，仕様書とソースコード の構造に共通性があることが前提であるなど，汎用性，利便性に欠ける．よって，開発者によるタグ付けなどの努力を必要とせず，自動的にトレーサビリティリンクを構築することが望まれる．

我々は，仕様書とソースファイルを入力として生成した文書ベクトルの類似度からトレーサビリティリンクを構築する手法を提案する．この手法では，仕様書に出現する単語やソースコード に出現する識別子から，仕様書とソースファイル 間のトレーサビリティリンクを自動で生成でき，開発者による努力を必要としないという利点がある．

ソフトウェア開発において，仕様がソフトウェアのどの箇所で実現されているかを追跡できることを前方追跡性，ソフトウェアの構成モジュールがどの仕様を実現するものかを追跡できることを後方追跡性と呼ぶ．本研究では，仕様書とソースファイル を 1 つの文書とみなして，Bow，TF-IDF，Doc2Vec を用いて文書ベクトルを生

成する．全ての仕様書とソースコード の文書ベクトルを求め，仕様書の 1 つの文書ベクトルに対して，全てのソースファイルの文書ベクトルとのコサイン類似度を計算し，類似度が高い順に対応する文書の候補とする．得られた候補に対して Top k Accuracy [9] を用いて，対応付けの精度を評価する，これを仕様書とソースコードを入れ替えて行うことで，前方追跡性と後方追跡性の両方の精度も評価する．

2. 提案手法

2.1 準備

提案手法において，前提となる事項を説明する．

2.1.1 JavaServerPages(JSP)

JavaServerPages（以下 JSP と呼称）は，HTML 内に Java のコードを埋め込んでおき，Web サーバで動的に Web ページを生成してクライアントに返す技術である [10]．特定の OS やハードウェアの仕様には依存せず，JSP の実行環境を整えた Web サーバソフトウェアならば原則としてどれでも同じように稼働させることができる．JSP のソースコード の例を図 2.1 に示す．図 2.1 から，1 つの JSP ファイルに<body>,</body>などの HTML 形式の記述や<%, %>で囲まれた Java のコードが存在することがわかる．

2.1.2 対象プロジェクト

本研究で対象としたプロジェクトは株式会社 SUNREX において開発された商業店舗の受発注アプリケーションが 2 件である．それぞれ，Excel シートで記述された内部設計書と，それに基づいて Java で開発されたサーバアプリケーションと JSP で開発された表示アプリケーションそれぞれのソースコードである．

ここで仕様書，JSP，Java それぞれの単位文書を定義しておく．仕様書は Excel の 1 つのシートを，JSP と Java は 1 つのソースファイルを単位文書として扱う．以下でも各文書群の単位文書はこの定義に従う．

2.1.3 トレーサビリティリンク構築

トレーサビリティリンク構築とは，仕様書や設計書やソースコードなどの文書の関連を紐付けた情報（トレーサビリティリンク）を構築することである．

仕様書には，システムの説明や仕様書修正履歴，テスト仕様，各機能ごとの要求仕様が書かれている．これらは主に自然言語で書かれているほか，ソースコード に出現するような識別子も含まれる．一方で，ソースコードには一般的にメソッド名

```
<%@ page contentType="text/html; charset=windows-31j" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/exapmle.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;
       charset=windows-31j" />
  <title>HelloWorld</title>
 </head>

<%
  int i = 10;
  int j = 5;
%>

 <body>
  <p>
   <%= i %> + <%= j %> = <%= i + j %> です。
  </p>
 </body>
</html>
```

図 2.1　JSP のソースコード 例

や変数名などを識別するために識別子と呼ばれる英字や数字からなる文字列のほか，コメントと呼ばれるプログラムの説明や補足するための自然言語も含まれる．そのため，仕様書とそれに対応するソースコードは，語彙が共通するものや語彙の集合が似通ったものが存在することが多いと考えられる．

2.2 文書ベクトルの生成手法

本研究では各文書から文書ベクトルを生成し，文書ベクトルの類似度からトレーサビリティリンクを構築する手法を提案する．文書群ごとに前処理を施し，データセットを取得する．文書ベクトルの生成方法は，BoW（Bag of Words），TF-IDF，Doc2Vec を使用し，文書ベクトルのコサイン類似度が高いものを対応する文書とすることでトレーサビリティリンク構築とする．各文書ベクトル生成手法の概要を以下に示す..

2.2.1 Bag of Words(BoW)

Bag of Words [11] とは，文書内の単語の語順を無視し，出現する回数を数え，その数を文書の特徴とする手法である．本研究では，文書群ごとに各文書の BoW を計算し，文書ベクトルを生成する．

2.2.2 TF-IDF

TF-IDF は，文書中に出現する特定の単語がどのくらい特徴的であるかを識別するための指標で，情報検索 [12]，文書クラスタリング [13]，特徴語抽出 [14]，映像中のオブジェクトマッチング [15] など，幅広いアプリケーションで利用することができる. また，単語と文書の共起性や情報検索の確率モデルなどの観点 [16–18] から理論的説明が与えられており，多くの人が TF-IDF を利用する根拠となっている. TF-IDF のような語の重み付け手法は，一般的に，局所的重み付けと大域的重み付けの二つの要素からなる. 局所的重み付けは，ある文書に対する語の出現頻度から計算され，対象とする文書によって重みは変化する. 一方，大域的重み付けは，文書集合全体における語の文書頻度から計算され，語の重みは対象とする文書によらず一定である.TF-IDF は具体的には，以下の式（2.1）により語 t の文書 $d \in D$ における重みを計

算する．

$$TF\text{-}IDF(t,d) = tf(t,d) \cdot \log \frac{|D|}{df(t)} \tag{2.1}$$

ここで，$tf(t,d)$ は文書 d における語 t の出現頻度，$df(t)$ は文書集合 D における t の文書頻度，$|D|$ は D の文書数である．

この式（2.1）の中で，局所的重み付けは以下の式（2.2）で表される．

$$TF(t,d) = tf(t,d) \tag{2.2}$$

また，大域的重み付けは以下の式（2.3）で表される．

$$IDF(t) = \log \frac{|D|}{df(t)} \tag{2.3}$$

特に，大域的重み付けである IDF [16] は様々な語の重み付け手法で採用されている．IDF が様々な語の重み付け手法として採用されている理由として，式（2.3）のような簡潔さとロバスト性が挙げられる．実際に，IDF は様々な文献 [16,17,19] において理論的にロバストであることが示されている．

本研究では，各文書群ごとに文書の単語 TF-IDF を計算し，文書ベクトルを生成する．

2.2.3 Doc2Vec

Doc2Vec [20] とは，単語を数百次元程度の実数ベクトルで表現する手法である Word2Vec を，文書単位へと拡張した手法である [20,21]．Word2Vec [21] はニューラルネットワークモデルであり，中心の単語から周辺の単語を予測するモデルである Skip-gram と，周辺の単語から中心の単語を予測するモデルである CBoW(Continuous Bag-of-Words) に分かれる．Doc2Vec においては，Skip-gram を拡張したものは PV-DBOW(Paragraph Vector with Distributed Bag-of-Words)，CBoW を拡張したのもは PV-DM(Paragraph Vector with Distributed Memory) と呼ばれる．

Doc2Vec は分散表現として単語ベクトル (Word Vector) とパラグラフベクトル (Paragraph Vector) を保持し，これらはニューラルネットワークモデルの中間層の重みで表現される．パラグラフベクトルは局所的な文脈の中で失われた情報を代表するものであり，パラグラフのトピック等抽象度の高い情報を記憶する機能を持

つ [20]．離散的で高次元，かつスパースな BoW による表現と比較して，分散表現は低次元で密な表現であり，単語や文書間の距離をより高い精度で計算可能である．

本研究では，1 つの文書群を訓練データとしてモデルを生成する．対応付けを図る文書群は，モデルに文書を入力し，文書ベクトル推論によって文書ベクトルを得ることができる．

2.2.4 コサイン類似度

式で説明

2.3 全手法に共通する処理

2.3.1 文書の言語の統一

2.1.2 項で述べた通り，対象プロジェクトは仕様書・JSP・Java の 3 つの文書群から構成される．そして各文書群に着目すると，仕様書は主にシステムの画面構成及びその画面での仕様が日本語で記されており，Java ファイルは JSP ファイル内に埋め込まれた Java コードに応じて実行される処理が記されている． そして，JSP ファイルはその中間に位置し，仕様書の通りに画面を構成する HTML 部とクライアントのリクエストに応じて動的に行われる java サーバ側での処理が書かれたスクリプト部から成る．そして，仕様書と JSP の HTML 部は，仕様書で定められた画面構成を JSP の HTML 部で実装するため，単語の共起性が高いと考えられる．一方，JSP のスクリプト部と Java のソースコードは主に識別子などのトークンの共起性が高いと考えられる．また，仕様書は日本語が主であり，JSP ファイルは主に英字と数字，および記号からなるソースコード と画面を構成する日本語とコメント，Java ファイルは主に英字と数字，および記号からなるソースコード が主であり，補足的にコメントが存在する．そのため，仕様書と JSP の HTML 部では日本語の単語の共起性が高く，JSP のスクリプト部と Java のソースファイルでは識別子などの英単語の共起性が高いと考えられる．したがって，仕様書と JSP の対応付けは仕様書と JSP の日本語のみを抽出したデータセットを，JSP と Java の対応付けは双方のソースコードから識別子などの英字の文字列のみを抽出したデータセットを用いて出現する単語から文書ベクトルを生成する．

仕様書と Java の対応付けは本研究では行っていない．上述した通り，仕様書は主に日本語から構成のに対し，Java のソースコードは主に英字の文字列から構成される．また，仕様書は自然言語であるのに対し，Java のソースコードはプログラミング言語である．これらのことから，仕様書と Java の文書間の単語の共起性は極めて低く，文書の単語のみから対応付けは困難であると考えられる．しかし，仕様書と JSP，JSP と Java のトレーサビリティリンクが構築されれば自動的に仕様書と Java も対応付けることができるため，仕様書と Java を直接対応付けなくても文書間のトレーサビリティリンクは構築されると考えられる．

2.4 文書の前処理手法

2.4.1 日本語データの前処理手法

日本語の文書は英語のようにスペースで単語が区切られていないため，形態素解析を行って単語に分割する必要がある．形態素解析エンジンは MeCab [22] を，形態素解析に用いる辞書は新語，固有名詞に強い mecab-ipadic-NEologd [23] を利用した．このうち，文書の特徴を表す単語として名詞のみを取り出してデータセットを作成した．

2.4.2 ソースコード の前処理手法

ソースファイルを文書として扱うために単語に分割する．英語などのテキストは基本的に空文字によって単語を分割する. ソースコードを単語に分ける際の規則は以下のようにする.

- 基本的にスペース区切りで単語を分ける．
- 以下の記号で区切る．

  ! ? ” ’ # $ % & | ( ) { } [ ] = < > + - * / \ ~ ^ @ : ; , .

これにより，ソースコード を単語の集合として扱うことができる．しかし，識別子や値が文字列である場合，複数の単語の組み合わせであることが多い. 実際に，識別子に関しては 1 文字や略語で構成されるものよりも複数の辞書語で構成されたものの方が理解しやすい [24,25] という研究結果がある．識別子を複数の語で構成する

ことはソースコードの可読性及び保守性の向上の観点から多くのプロジェクトで 使われている. また，識別子を複数の語で構成する際に，単語ごとの可読性を上げるためにいくつかの命名記法がある. 代表的なものとしては，CamelCase 記法，snake_case 記法，Kebab-case 記法が挙げられる．CamelCase 記法とは，単語の先頭を大文字にして複数の単語を繋げる記法である. 次に，snake_case 記法と Kebab-case 記法は，単語の間にそれぞれアンダースコア，ハイフンを入れることで単語を繋げる記法である．上述した規則に加えて，これらの記法に基づいて単語を分解することで，さらに最小単位の単語を得ることができる．実際に識別子の分解を行った例を表 2.1 に示す．識別子の分解が精度向上に寄与するか確かめるため，単語の分解を行わずに固有名詞のように扱ったデータセットと単語の分解を行ったデータセットの両方を作成した．

2.4.3 データセットの取得

2.3.1 項述べた通り，仕様書と JSP，JSP と Java に分けて前処理手法を適用することでデータセットを得る．仕様書と JSP の全文書にから日本語単語のデータセットを取得する．JSP と Java の全文書から英字文字列のデータセットを取得する．

表 2.1　識別子分解の例

記法	分解前	分解後
CamelCase	HogeHogeFoobar	Hoge, Hoge, Foobar
snake_case	hoge_foo_bar	hoge, foo, bar
Kebab-case	hoge-foo-bar	hoge, foo, bar

3. 評価実験

3.1 研究設問

実験は以下の研究設問に回答するために設計される．

RQ1 実プロジェクトを対象とした仕様書と JSP 間のトレーサビリティリンクはどの程度の精度で構築できるか

RQ2 実プロジェクトを対象とした JSP と Java 間のトレーサビリティリンクはどの程度の精度で構築できるか

RQ3 トレーサビリティリンクの構築の精度が最も高かった手法はどれか

3.2 実装

実験のための実装は Python™3.6 [26] を開発言語として用い，前処理にはデータ分析ライブラリ pandas（バージョン 0.25.3）[27]，その他のライブラリとして，BoW，TF-IDF の算出は scikit learn（バージョン 0.22.1）[28] の CountVectorizer [29]，TfidfVectorizer [30] を，Doc2Vec は gensim（バージョン 3.8.0）の Doc2Vec [31] を利用した．

3.3 実験方法

実験の手順を手法ごとに以下に示す．

3.3.1 BoW，TF-IDF

1. 対応付けを行う 2 種類のデータセット D_1，D_2 に対して，BoW，TF-IDF 手法を用いて文書ベクトルを生成する．文書ベクトルの要素はその文書に出現する単語の BoW，TF-IDF となり，次元は文書内の単語の種類数となる．
2. D_1 の各文書ベクトルに対して，D_2 の全文書の文書ベクトルとのコサイン類似度を計算し，類似度が高い順に 5 つの文書を得る．これは，D_1 の各文書に対応する文書の候補を 5 つ得たと言える．

3. 前方追跡性と後方追跡性の両方を評価するために，D_1 と D_2 を入れ替えて，手順 2 を行う．

この実験手順を，(D_1，D_2）＝（仕様書，JSP），(JSP，Java）として行う．

3.3.2 Doc2Vec

1. 対応付けを行う 2 種類のデータセット D_1，D_2 に対して，D_1 を訓練データとし，Doc2Vec モデルを生成する．
2. モデルに D_2 の文書を入力として文書ベクトル推論により文書ベクトルを得る．
3. 実験手順 2 で得られた文書ベクトルと D_1 の各文書ベクトルとのコサイン類似度を計算し，類似度が高い順に 5 つの文書を得る．
4. D_1 と D_2 を入れ替えて，手順 1~ 3 を行う．

この実験手順を，(D_1，D_2）＝（仕様書，JSP），(JSP，Java）として行う．

3.3.3 評価用データ

本研究において，モデル評価，実験結果を評価するために，予め正解の対応付けがラベル付けされた評価用データを要する．今回，実験結果の仕様書と JSP は開発者から評価を得られたが，それ以外のモデルの評価及び実験結果の評価を行うための評価用データは得られなかった．この評価は，プログラミング経験が 6 年の著者が 1 人で行う．目視により評価用データを作成し，モデル及び実験結果の評価を行った．Doc2Vec モデルのパラメータ は [32] を元に，($window$， $vector\_size$，$sample$，) 変化させて，評価が最も良かったものを採用する．

3.3.4 評価指標

提案手法のトレーサビリティ構築精度の評価指標として Top k Accuracy を利用する．Top k Accuracy とは，入力に対して最もらしい回答を k 個出力し, その出力の中に含まれる正しい回答の割合である．

本研究では，実験により得られた文書候補 5 つに対して Top 5 Accuracy を評価し，$k = 1$〜5 について精度を評価する．

4. 結果

本章では，対応付けを行った文書群ごとに実験結果を示し，最後に研究設問に回答する．

4.1 仕様書と JSP

ここでは，仕様がどの JSP ファイルで実現されているかを追跡できることを前方追跡性，JSP ファイルがどの仕様を実現するものかを追跡できることを後方追跡性とする．3 つの文書ベクトル生成手法を用いて得られた対応付けの Top k Accuracy 評価の結果を両方の追跡性ごとに示す．

4.1.1 前方追跡性

図 4.1 は，仕様書と JSP における前方追跡性の精度を Top 5 Accuracy により評価した結果である．3 つの手法の精度の差は小さく，$k = 5$ の場合，Doc2Vec が示した 25%が最も高い精度となった．この時の Doc2Vec モデルのパラメータは，($window = 3$, $vector\_size = 300$, $sample = 1e - 2$）であった．また $k = 1$ に比べて $k = 2$ で精度が上がっていることが読み取れるが，$k = 2$ 以上では大きな精度の上昇は見られなかった．$k = 2$ 以上では TF-IDF よりも BoW の方が精度がわずかに高い．これは，全文書に頻出する単語であっても，対応する単語は出現回数が近いことを意味している．

4.1.2 後方追跡性

図 4.2 は，仕様書と JSP における後方追跡性の精度を Top 5 Accuracy により評価した結果である．$k = 5$ では TF-IDF が 76%，BoW が 73%の精度を示し，TF-IDF が最も高い精度となった．また，k の値が高いほど精度も上昇しているのがわかる．一方で，Doc2Vec は $k = 5$ においても 58%に留まっており，他の 2 手法と比べて低い精度となった．この時の Doc2Vec モデルのパラメータは，($window = 5$, $vector\_size = 300$, $sample = 1e - 3$）であった．

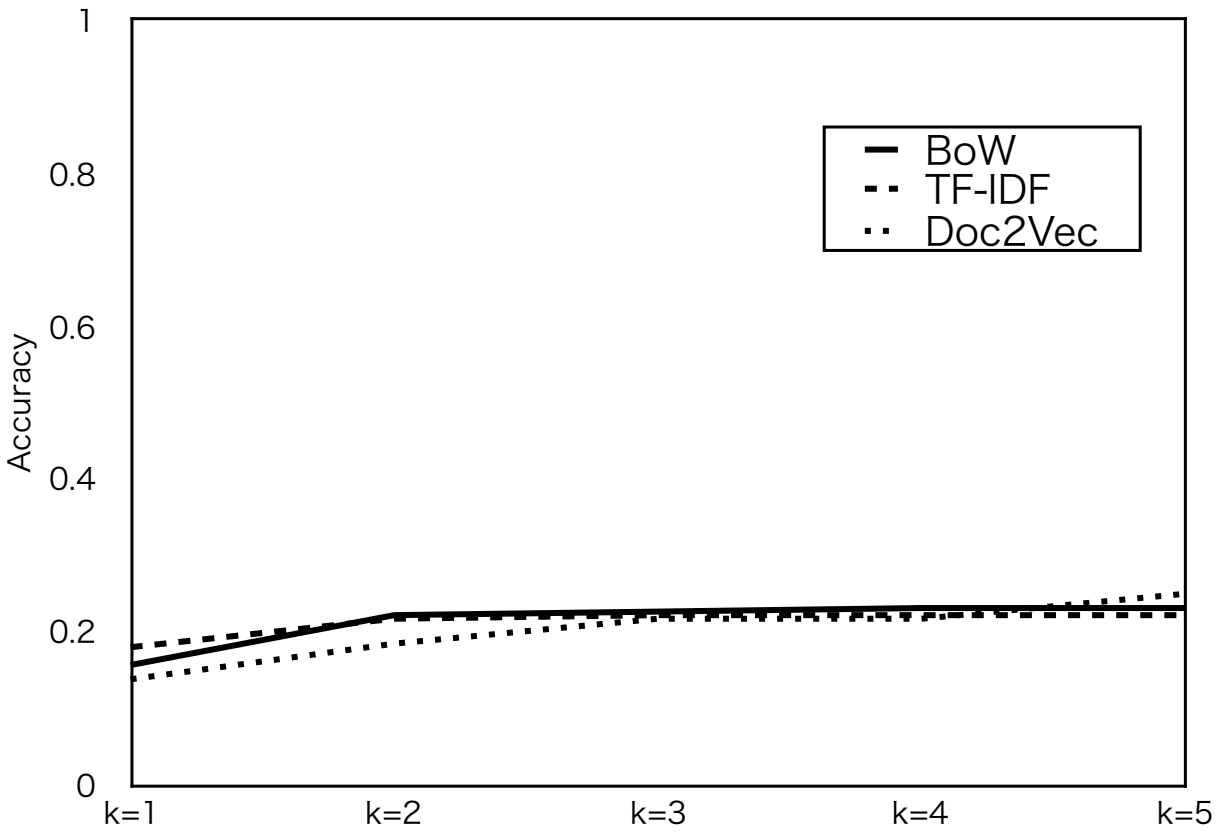

図 **4.1**　仕様書と **JSP** における前方追跡性の **Top 5 Accuracy** の評価結果

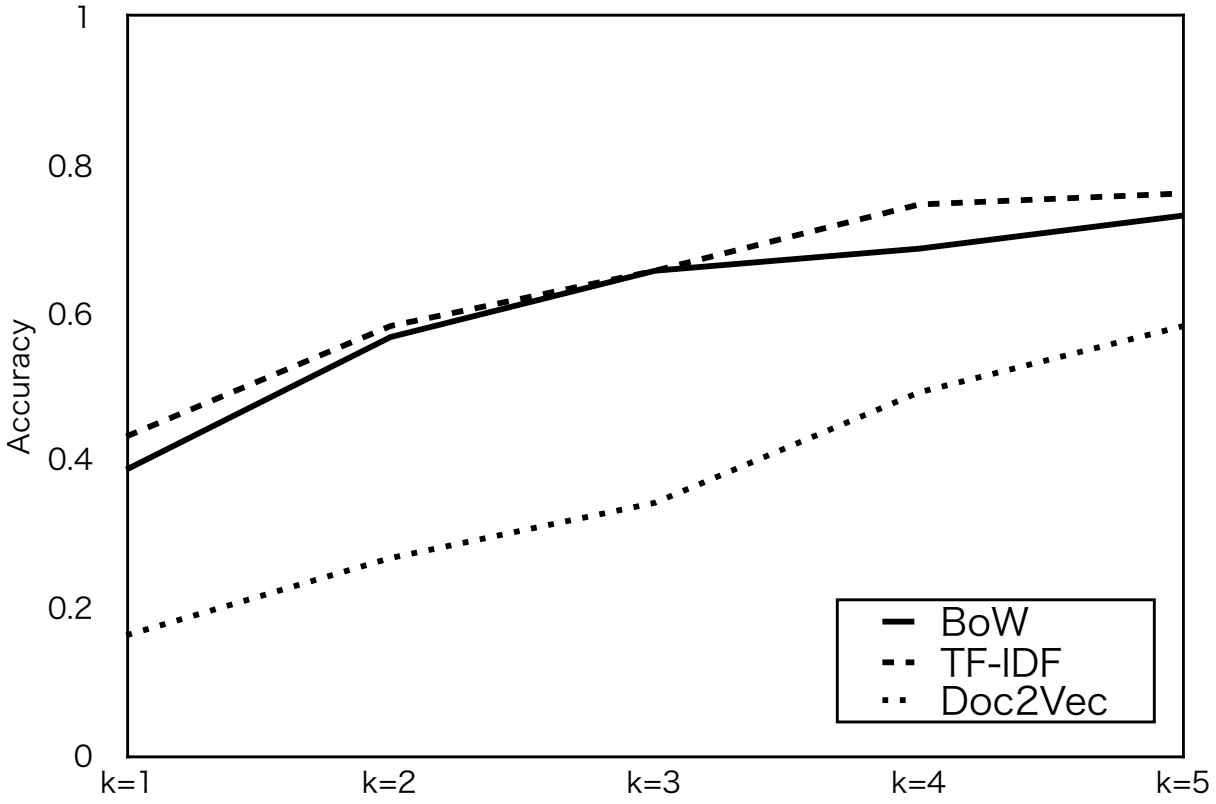

図 **4.2**　仕様書と **JSP** における後方追跡性の **Top 5 Accuracy** の評価

4.2 JSP と Java

ここでは，JSP がどの Java ファイルと対応するかを追跡できることを前方追跡性，Java ファイルがどの JSP ファイル対応するかを追跡できることを後方追跡性とする．識別子の分解を行ったデータセットとそうでないデータセットに対して，３つの文書ベクトル生成手法を用いて得られた対応付けの Top 5 Accuracy 評価の結果を両方の追跡性ごとに示す．

4.2.1 前方追跡性

図 4.3 は，識別子の分解を行った，図 4.4 は，識別子の分解を行なわなかった JSP と Java における前方追跡性の精度を Top 5 Accuracy により評価した結果である．識別子の分解処理を行った時の Doc2Vec モデルのパラメータは，($window = 1$, $vector\_size = 300$, $sample = 1e-1$)，行わなかった時の Doc2Vec モデルのパラメータは，($window = 5$, $vector\_size = 300$, $sample = 1e-4$）であった．であった．識別子の分解処理の有無に関わらず TF-IDF が最も精度が高く，識別子の分解を行った場合，$k = 5$ では 75%の精度を示した．続いて Doc2Vec が 58%，BoW が 36%となった．また，識別子の分解を行ったデータセットの方がどの手法も精度が上昇した．

4.2.2 後方追跡性

図 4.5 は，識別子の分解を行った，図 4.6 は，識別子の分解を行なわなかった JSP と Java における後方追跡性の精度を Top 5 Accuracy により評価した結果である．識別子の分解処理を行った時の Doc2Vec モデルのパラメータは，($window = 5$, $vector\_size = 300$, $sample = 1e-4$)，行わなかった時の Doc2Vec モデルのパラメータは，($window = 2$, $vector\_size = 300$, $sample = 1e-2$）であった．であった．識別子の分解を行った場合では，Doc2Vec が最も精度が高く，$k = 5$ で 64%，続いて TF-IDF が 48%の精度を示した．BoW の精度は他の 2 手法に比べて極めて低く，$k = 5$ においても 13%の精度に留まった．識別子の分解を行わなかった場合，TF-IDF が最も精度が高く，その精度は $k = 5$ で 40%であり，続いて Doc2Vec が 22%，BoW13%となった．どの手法も識別子の分解を行った方が精度は高くなった．

4.2.3 識別子の分解の寄与

文書ベクトルの生成手法ごとの識別子の分解処理による精度の上昇率を表 4.1 に示す．上昇率は，k の値ごとに求めた各文書ベクトル生成手法の上昇率の平均をとした．識別子の分解処理はどの文書ベクトル生成手法においても精度の上昇に寄与することが示唆された．

4.2.4 トレーサビリティリンク構築精度

表 4.2 は $k = 1$ の精度に着目し，精度が最も高いものを示している．つまり，得られた 1 つの文書の候補が真に対応する文書であった割合であり，開発者の努力を必要としないトレーサビリティリンクの構築精度を示していると言える．仕様書と JSP の対応付けでは TF-IDF 手法による後方追跡で 43%の精度を示し，JSP と Java の対応付けでは識別子分解処理を行った TF-IDF による前方追跡で 50%の精度を示した．

4.3 研究設問への回答

4.3.1 RQ1:提案手法によってトレーサビリティリンクは構築できるか

仕様書と JSP 間は **TF-IDF による後方追跡で 76%**，JSP と Java 間は**識別子の分割処理を行った TF-IDF 手法による前方追跡で 75%**の精度で得られる文書の候補からトレーサビリティリンクを構築できる．

4.3.2 RQ2:提案手法の構築精度はどの程度か

4.2.4 項に示した通り，仕様書と JSP は **BoW** が **73%**，**TF-IDF** が **76%**，**Doc2Vec** が **58%**の精度で対応する文書を得た．また，JSP と Java は **BoW** が **36%**，**TF-IDF** が **75%**，**Doc2Vec** が **64%**の精度で対応する文書を得た．

識別子の分解処理は 3 つの文書ベクトル生成手法全てにおいて精度向上に寄与した．

仕様書と JSP においては，3 つの文書ベクトル生成手法全てで後方追跡性が高い結果を示した．JSP と Java においては，BoW と TF-IDF は，前方追跡性の方が高く，

表 4.1　識別子分解処理による精度上昇率 (%)

	前方追跡性	後方追跡性
BoW	168.9	9.9
TF-IDF	20.3	157.6
Doc2Vec	126.1	12.0

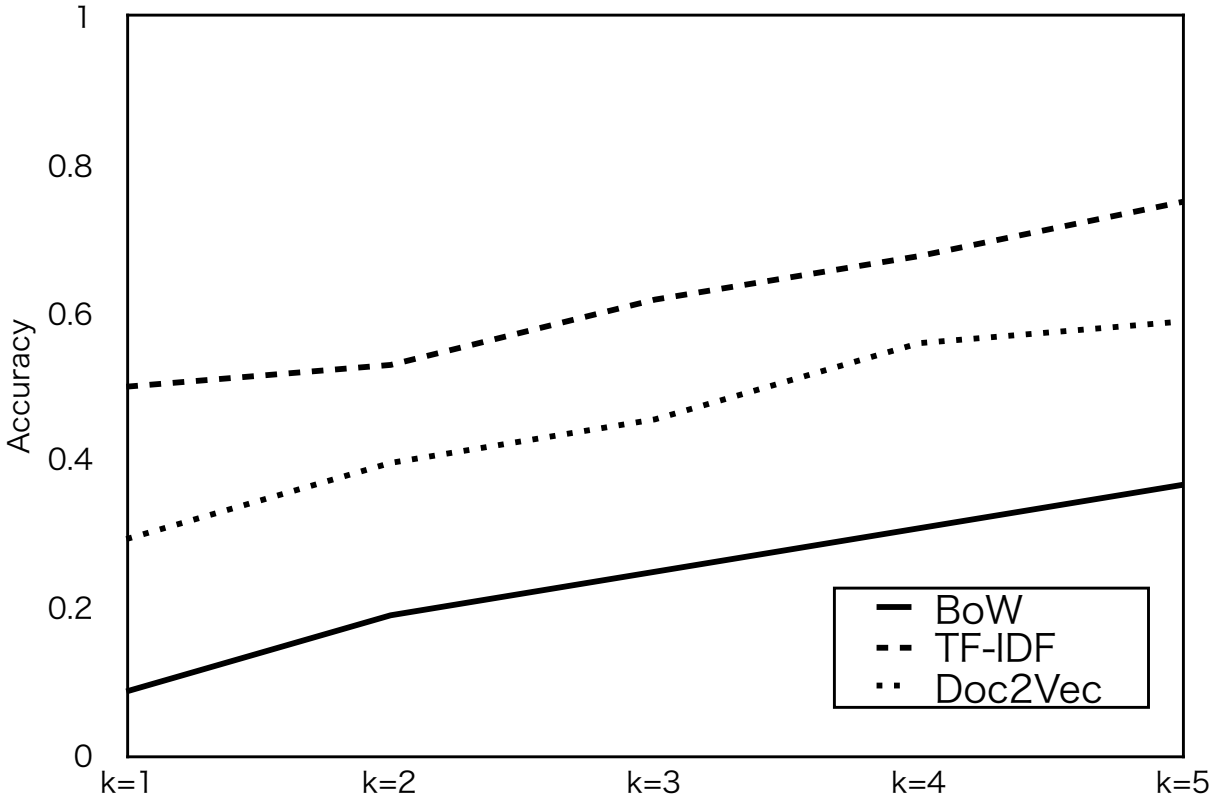

図 4.3　識別子の分解を行った JSP と Java における前方追跡性の Top 5 Accuracy の評価結果

表 4.2　トレーサビリティリンク構築精度

文書群	Accuracy	手法
仕様書と JSP	0.43	TF-IDF による後方追跡
JSP と Java	0.50	識別子分解をした TF-IDF による前方追跡

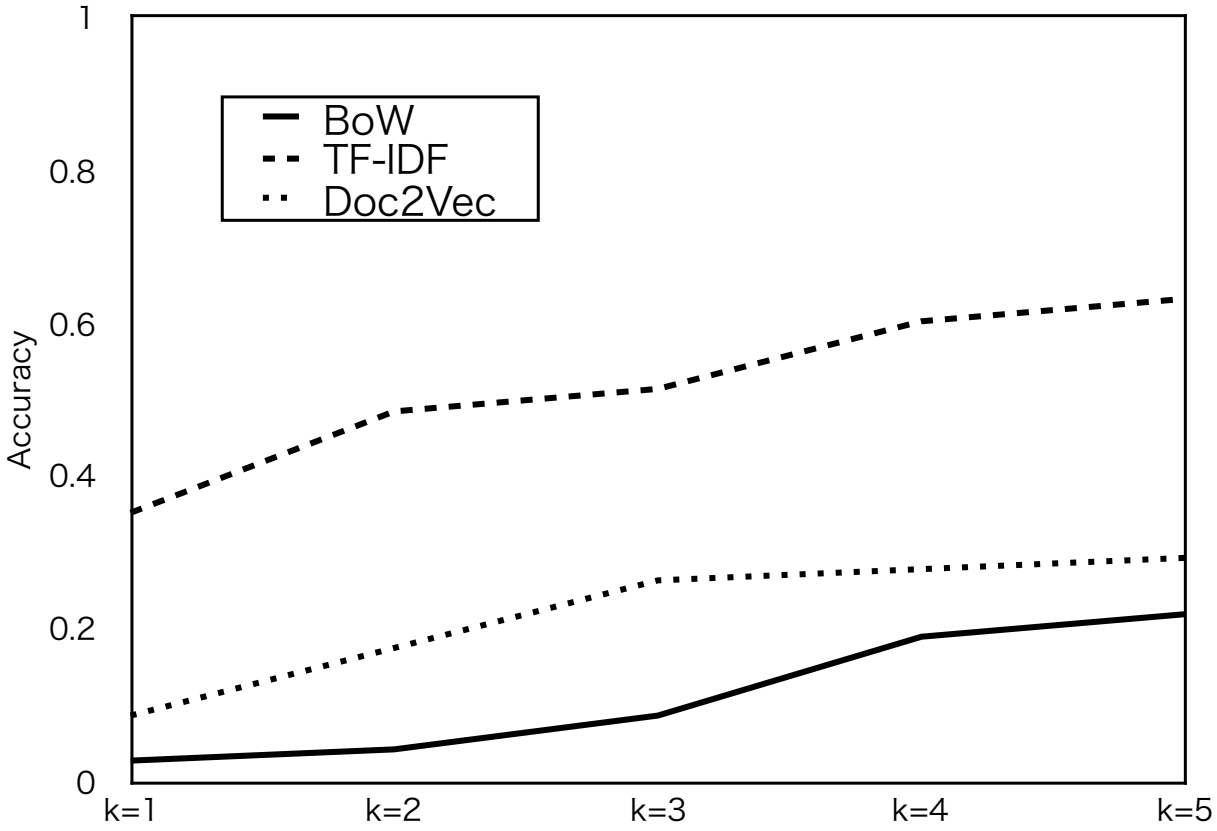

図 **4.4**　識別子の分解を行わなかった **JSP** と **Java** における前方追跡性の **Top 5 Accuracy** の評価結果

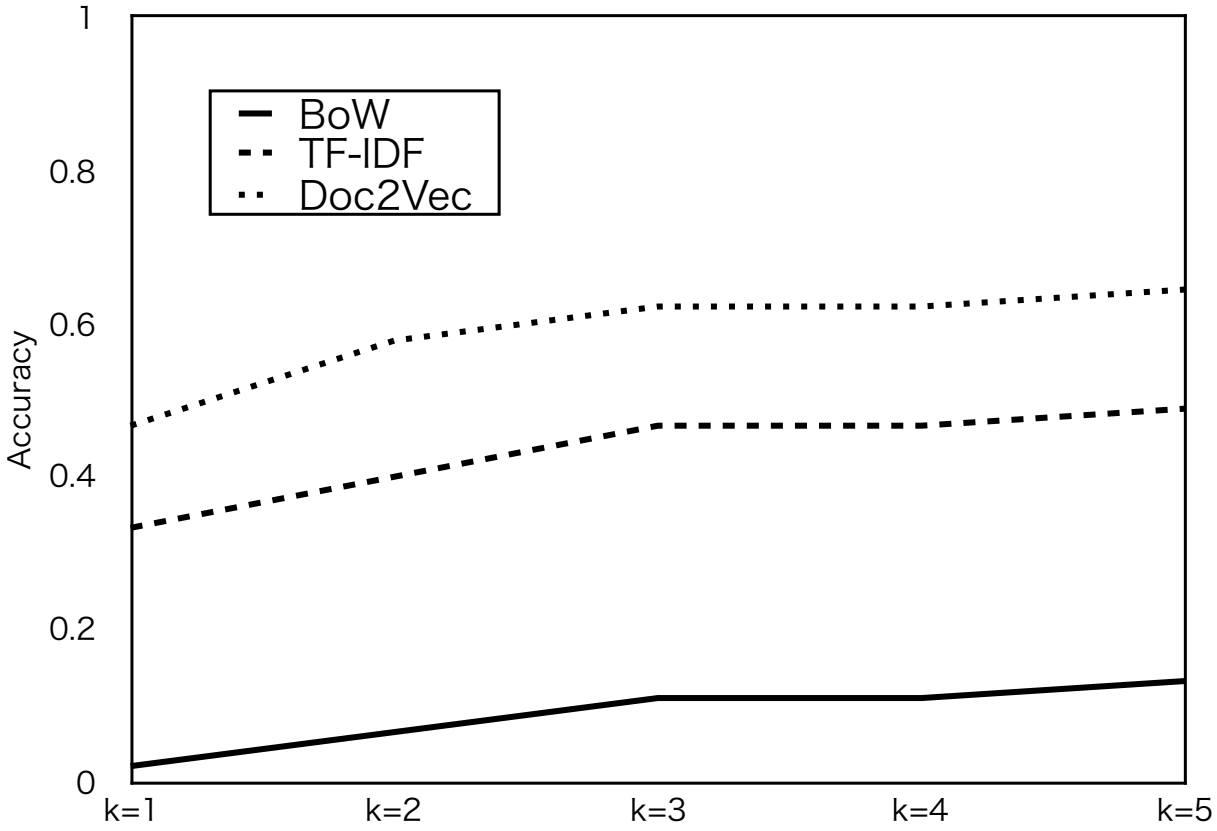

図 **4.5**　識別子の分解を行った **JSP** と **Java** における後方追跡性の **Top 5 Accuracy** の評価結果

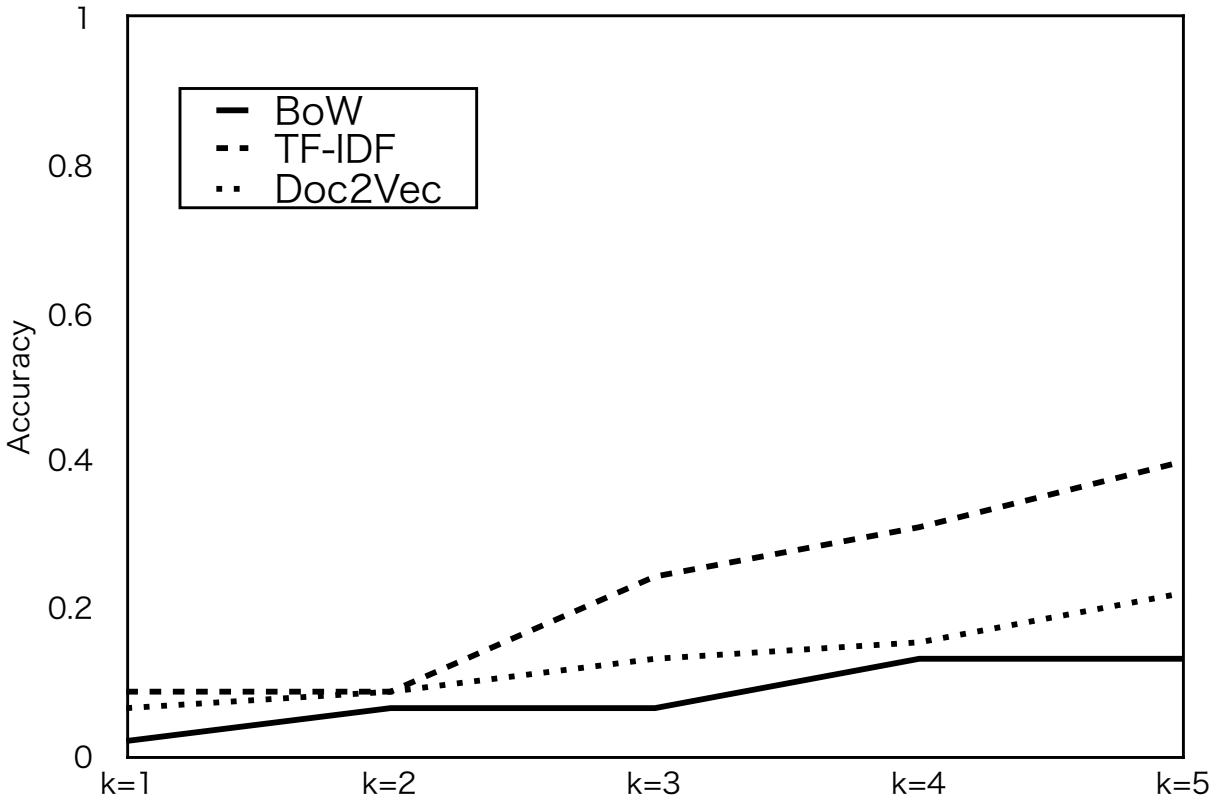

図 **4.6**　識別子の分解を行わなかった **JSP** と **Java** における後方追跡性の **Top 5 Accuracy** の評価結果

Doc2Vec は識別子の分解処理を行った場合，後方追跡性がわずかに高く，行わなかった場合，前方追跡性が高い結果となった．

4.3.3 RQ3:トレーサビリティリンクの構築に適した手法はどのような手法か

3 つの文書ベクトルの生成手法を比較すると，JSP と Java における識別子の分解処理を行った後方追跡以外では，TF-IDF 手法が最も高い精度を示している．また，JSP と Java 間においては，どの手法にも識別子の分解処理は精度向上に寄与した．したがって，**トレーサビリティリンクの構築に適した手法は TF-IDF 手法**である．

5. 関連研究

トレーサビリティリンクは，ソフトウェア開発の工程で生産される仕様書とソースコード などの文書間を紐付けた情報である．トレーサビリティリンクについて，リンクの複雑さに関する研究 [33]，リンク抽出や復元に関する研究が行われている [34, 35]．また，トレーサビリティリンクの構築は，これまで様々な手法が提案されている．仕様書とソースコード を文書の一種と考え，TF-IDF とソースコード から構築したコールグラフを用いて，ソースコード 上の関数群を対応付ける手法 [36] や，仕様書の章構造とソースコード の AST 構造を利用した手法 [8]，予め文書ごとに付けられたタグからトレーサビリティリンク構築を支援するツールなども開発されている [2]．しかし，これらの既存の手法は，事前に開発者による文書へのタグ付けや仕様書の章構造とソースコード の木構造の共通性などの前提条件があり，構造に依らず自動で仕様書とソースコード の対応付けを行うことは困難であると考えられる．

本研究は，開発者によるへのタグ付けなどの努力を必要としない点，文書間に構造上の共通点がなくても，出現する単語のみからトレーサビリティリンクを構築できる点で既存手法との優位性がある．一方で，単語のみから文書をベクトルで表現するため，文書の言語やプロジェクトを構成する文書の種類によっては適用できない可能性がある．データセットに応じて，適切な前処理手法の適用や文書ベクトルの生成手法を選択する必要がある．また，機械翻訳を用いて異言語の文書同士にも適用できるようにする，他の文書の分散表現を得る手法を適用し精度を比較するなど改善の余地がある．

6. 妥当性への脅威

6.1 構成概念妥当性

1 つの文書に対して，対応する文書の候補を獲得することでトレーサビリティリンクの構築を図った．しかし，中には対応する文書がないものも存在した．本研究では，そういった文書に対しても対応する文書の候補を提示し，対応する文書を候補内に提示することができなかったものとして精度を算出した．しかし Top k Accuracy は，文書群中のある文書に対して k 個の候補に真に対応するものがある文書の組み合わせが存在する割合であるため，本来ならば対応がない文書に対しては，文書間類似度に閾値を設けるなどして，対応がないことを示す出力を上位に候補として提示することで精度を評価すべきである．今回の場合，全ての文書に対して文書間に単語の共通性，共起性がないことを目視で確認して評価することとなり現実的ではない．

6.2 外的妥当性

本研究では，仕様書と JSP ファイルと Java ファイルの 3 文書群から構成される実プロジェクトを対象とした．また，仕様書と JSP には日本語という共通言語があったため日本語のみを抽出することで対応付けを行った．しかし，仕様書と Java のみで構成されるプロジェクトや，英語の仕様書とソースコード で構成されるプロジェクトなど他のプロジェクト構成に対して一般化することは困難であると考える．

また，同じプロジェクト構成でも生産される文書は開発者に依存しやすく，大規模なプロジェクトで開発者の数が増えるほど文書も複雑になると予想できる．このような場合，データセットに応じた前処理手法や文書ベクトル生成手法を検討する必要があり，外的妥当性への脅威となり得る．

6.3 内的妥当性

本研究では，モデル評価用データと JSP と Java の対応付けの評価用データを目視で作成した．これは著者の知見および経験，各言語の構文規則などを参考に手動で

作成した．その妥当性には疑問の余地があり，真の対応付けではなかった可能性がある．これは，本研究の今後の課題であり，開発者に

7. 結言

本研究では，仕様書と JSP と Java ファイルからなる実プロジェクトを対象とし，開発者の努力を必要とせず自動的にトレーサビリティリンクを構築する手法を提案した．異なる前処理手法と文書ベクトル生成手法を用いて，得られた文書ベクトルのコサイン類似度から対応する文書の候補を獲得し，Top k Accuracy を用いて，手法ごとの精度，前方追跡性，後方追跡性について評価した．仕様書と JSP，JSP と Java 共に 75%の精度でトレーサビリティリンク構築できることを示した．また，前処理手法の精度への寄与，前方追跡・後方追跡によって精度が高くなる手法は異なるという知見を得た．

今後の課題として，開発者による全ての文書群間の評価用データの作成やデータセットに対する適切な前処理手法の提案，他の文書ベクトル生成手法との精度の比較が挙げられる．

参考文献

[1] J. Frederick P. Brooks, “No Silver Bullet Essence and Accidents of Software Engineering,” IEEE Computer, pp.10—-19, 1987.

[2] 宮本貴之，渡辺政彦，高田広章，鶴保征城，“オープントレーサビリティツールプラットフォーム TERAS,” SEC journal，vol.9，no.4，pp.176–181，2014.

[3] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, “Tesseract: Interactive visual exploration of socio-technical relationships in software development,” Proceedings - International Conference on Software Engineering, pp.23–33, 2009.

[4] W. Zhao, L. Zhang, Y. Liu, J. Sun, and F. Yang, “SNIAFL: Towards a Static Noninteractive Approach to Feature Location,” ACM Trans. Softw. Eng. Methodol., vol.15, no.2, pp.195–226, apr 2006.

[5] J.H. Hayes, A. Dekhtyar, S.K. Sundaram, and S. Howard, “Helping analysts trace requirements: an objective look,” Proceedings. 12th IEEE International Requirements Engineering Conference, 2004., pp.249–259, Sep. 2004.

[6] J.H. Hayes, A. Dekhtyar, and S.K. Sundaram, “Advancing candidate link generation for requirements tracing: the study of methods,” IEEE Transactions on Software Engineering, vol.32, no.1, pp.4–19, Jan. 2006.

[7] Y. Zhang, “An ontology-based approach for the recovery of traceability links,” Proc. 3rd Int, Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering, 2006, pp.●●–●●, 2006.

[8] 田原貴光，林　晋平，佐伯元司，“仕様書と Java ソースコードの構造の類似性に基づく対応付け,” 情報処理学会研究報告ソフトウェア工学（Se），vol.2008，no.29(2008-SE-159)，pp.139–146，2008.

[9] S. Boyd, C. Cortes, M. Mohri, and A. Radovanovic, “Accuracy at the top,” in Advances in Neural Information Processing Systems, eds. F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, vol.2, pp.953–961, Curran Associates, Inc., 2012.

[10] S. Microsystems, Java Server Pages Specification, （オンライン），入手先

〈http://java.sun.com/products/jsp/index.html〉（参照 2020-1-20）.

[11] C.D. Manning and H. Schtze, “Foundations of statistical natural language processing,” The MIT Press, pp.●●–●●, 1999.

[12] H.C. Wu, R.W.P. Luk, K.F. Wong, and K.L. Kwok, “Interpreting TF-IDF Term Weights as Making Relevance Decisions,” ACM Trans. Inf. Syst., vol.26, no.3, pp.●●–●●, jun 2008.

[13] B. M, F. Ke, and W. Ester, “Hierarchical Document Clustering Using Frequent Itemsets,” Proceedings of SIAM International Conference on Data Mining(SDM), pp.●●–●●, 2003.

[14] K.S. Hasan and V. Ng, “Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art,” Trabajos de Prehistoria, vol.58, no.1, pp.171–186, 2001.

[15] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” Proceedings of the IEEE International Conference on Computer Vision, vol.2, no.Iccv, pp.1470–1477, 2003.

[16] A. Aizawa, “An information-theoretic perspective of tf–idf measures,” Information Processing & Management, vol.39, no.1, pp.45–65, 2003.

[17] S. Robertson, “Understanding Inverse Document Frequency: On Theoretical Arguments for IDF,” Journal of Documentation - J DOC, vol.60, pp.503–520, 2004.

[18] D. Hiemstra, “A probabilistic justification for using tf × idf term weighting in information retrieval,” International Journal on Digital Libraries，vol.3，no.2，pp.131–139，2000.

[19] D. Metzler, “Generalized Inverse Document Frequency,” Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp.399–408, 2008.

[20] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” 31st International Conference on Machine Learning, ICML 2014, vol.4, pp.2931–2939, 2014.

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations ofwords and phrases and their compositionality,” Advances in Neural Information Processing Systems, pp.1–9, 2013.

[22] T. Kudo, K. Yamamoto, and Y. Matsumoto, “Applying Conditional Random Fields to Japanese Morphological Analysis,” ●●, pp.230–237”, jul 2004.

[23] 佐藤 敏紀, GitHub, （オンライン）, 入手先〈https://github.com/neologd/mecab-ipadic-neologd〉（参照 2020-1-30）.

[24] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, “What’s in a name? a study of identifiers,” 14th IEEE International Conference on Program Comprehension (ICPC’06), pp.3–12, June 2006.

[25] J. Hofmeister, J. Siegmund, and D.V. Holt, “Shorter identifier names take longer to comprehend,” 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.217–227, Feb. 2017.

[26] Python Software Foundation, Python.org, （オンライン）, 入手先〈https://www.python.org/〉（参照 2018-1-30）.

[27] The pandas project, pandas: Python Data Analysis Library, （オンライン）, 入手先〈https://pandas.pydata.org/〉（参照 2020-1-30）.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in python,” Journal of Machine Learning Research, vol.12, pp.2825–2830, 2011.

[29] scikitlearn, CountVectorizer, （オンライン）, 入手先〈https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html〉（参照 2020-1-20）.

[30] scikitlearn, TfidfVectorizer, （オンライン）, 入手先〈https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html〉（参照 2020-1-30）.

[31] R. Rehurek and P. Sojka, “Software framework for topic modelling with large corpora,” In LREC, pp.●●–●●, 2010.

[32] J.H. Lau and T. Baldwin, “An empirical evaluation of doc2vec with practical insights into document embedding generation,” Proceedings of the 1st Workshop on Representation Learning for NLP, pp.78–86, Aug. 2016.

[33] 充晴北村，正則高木，敬三山田，淳佐々木，“ソフトウェア開発におけるトレーサビリティの複雑さの表現と単純化方法の提案,” 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス，vol.112，no.275，pp.135–142，2012.

[34] 良介土屋，弘宜鷲崎，良彰深澤，正恭加藤，真澄川上，健太郎吉村，“派生プロダクト群における要求・実装間のトレーサビリティリンク抽出,” 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス，vol.112，no.275，pp.123–128，2012.

[35] 上田健之，“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,” 電子情報通信学会論文誌. D, 情報・システム，vol.97，no.3，pp.●●–●●，2014.

[36] W. Zhao, L. Zhang, Y. Liu, J. Sun, and F. Yang, “SNIAFL: Towards a static noninteractive approach to feature location,” Proceedings - International Conference on Software Engineering, vol.26, no.2, pp.293–303, 2004.

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢や本報告書の作成，データの提供に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系 水野修教授，崔恩瀞助教に厚く御礼申し上げます．また、貴重な開発データのご提供や実験結果の評価にご尽力いただきました株式会社 SUNREX の皆様に深く感謝いたします。最後に，本報告書執筆にあたり貴重な助言を多数頂きました本研究室の上北裕也君，北村紗也加さん，國領正真君，近藤将成先輩，里形洋道君，塩津拓真君， 杉浦智基君，西浦生成先輩，廣瀬早都希さん，舟山優君，洪浚通先輩，山本凱君，若林奎人君，脇上幸洋君， 渡辺大輝君をはじめとする本学情報

工学専攻，情報工学課程の皆様，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します.