Vol.2018-SE-200 No.11
2018/12/3

自然言語処理を用いたソースコード上の論文引用の自動検出

井ノ口 輝[1,a)]　畑 秀明[1,b)]　石尾 隆[1,c)]　松本 健一[1,d)]

概要：ソフトウェア開発プロジェクトに利用される技術は様々だが，どのような技術がどのプロジェクトで利用されているかは管理されていないため，把握は困難である．利用されている技術の情報源として，開発者による論文の引用が挙げられる．開発者はソースコード上のコメントで論文を引用し，アルゴリズムの実装や，実装の詳細を他の開発者に伝えるなどの活用をしている．しかし，引用される論文は管理されておらず，引用の形式は様々であるためキーワード検索等による検出も困難である．論文の引用をコメントの目視によって分類するには，膨大なコストを要する問題がある．このため，本研究ではソースコード上のコメントから，論文引用の機械的な検出に取り組む．分類のために自然言語処理の手法である固有表現抽出を利用する．固有表現抽出は文章から事前に学習したモデルを用いて固有表現を抽出する．これにより，論文の引用に関する固有表現をラベルとして学習，抽出し，論文の引用検出を試みる．モデルを学習させるために，Linux や Android のような大規模な 11 の OSS プロジェクトから約 330 万件のコメントを用意する．コメント内の論文引用の有無によって一部のコメントを抽出し，学習，評価に利用する．この結果，引用を含むコメントのほとんどを検出できたが，引用を含まないコメントも多く含んだ．検索結果の分析により，検出性能の向上が期待できると確認できた．

1. はじめに

ソフトウェア開発プロジェクトにおいて，最終成果物であるソースコードがどのような要求，設計，実装技術から成り立っているかを追跡するトレーサビリティの実現が重要である [3]．実装技術の 1 つであるアルゴリズムは，ソフトウェアのパフォーマンスを大きく左右することから，関連したアルゴリズムを必要に応じて検索，学習できるようにすることが重要である [1]．ソースコードとアルゴリズムの間のトレーサビリティが実現されれば，開発者が自身のプロジェクトや，関連分野のプロジェクトで使われるアルゴリズムを網羅的に把握し，自身のプロジェクトが利用するアルゴリズムを改善できる．

アルゴリズムの重要な情報源の 1 つが論文である [1]．論文はアルゴリズムを体系立てて詳細に記述しており，アルゴリズムの管理に適した媒体である．実際に，開発者は論文をソースコードコメント上で引用し，実装の詳細を他の開発者に共有するといった活用が見られる [6]．このような論文の引用を把握することで，プロジェクトで利用される技術・アルゴリズムの一部を把握できると考えられる．

しかし，引用されている論文の把握は困難である．文献引用としてコメント内に明確に記述されているにも関わらず，その記述方法は自然言語によるものであり，機械的な検出が行われていない．ソースコードコメントの数は一般に非常に多く，人間によるソースコードコメントの目視による確認は，膨大なコストを要する．

そこで本研究では，ソースコードコメントから論文引用の機械的な検出に取り組む．具体的には，固有表現抽出という文中から固有名詞を抽出する技法 [2][5] を適用し，英語で書かれたコメントの中から，論文の著者名や論文誌名といった固有名詞を認識することを試みる．

以降，関連研究と文献引用に関するコメントの例を紹介した後，適用実験の結果について述べる．

2. 関連研究

小西らは，OSS（Open Source Software）のソースコードコメント上で，どの論文が多く引用され，ソフトウェア開発に貢献しているかを明らかにした [6]．コード検索サイト Ohloh[*1] に登録されている C 言語と Java 言語のプロジェクト，約 15 万件を対象にキーワード検索と目視によって分析した．キーワードはソフトウェア関連論文誌名の Impact Factor 上位 50 までを対象とし，“ACM”を含む

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a)] inokuchi.akira.hw0@is.naist.jp
[b)] hata@is.naist.jp
[c)] ishio@is.naist.jp
[d)] matumoto@is.naist.jp

[*1] https://www.openhub.net/，最終アクセス：2018/11/05

```
/*Constants for use with OD_DIVU_SMALL().
  See \cite{Rob05} for details on computing these constants.
  @INPROCEEDINGS{Rob05,
    author="Arch D. Robison",
    title="{N}-bit Unsigned Division via {N}-bit Multiply-Add",
    booktitle="Proc. of the 17th IEEE Symposium on Computer Arithmetic
     (ARITH'05)",
    pages="131--139",
    address="Cape Cod, MA",
    month=Jun,
    year=2005
  }*/
```

図 1 BibTeX 記法を用いた論文引用コメントの例（gecko-dev より）

```
/**
 * A JavaScript implementation of the DeltaBlue constraint-solving
 * algorithm, as described in:
 *
 * "The DeltaBlue Algorithm: An Incremental Constraint Hierarchy Solver"
 *   Bjorn N. Freeman-Benson and John Maloney
 *   January 1990 Communications of the ACM,
 *   also available as University of Washington TR 89-08-06.
 *
 * Beware: this benchmark is written in a grotesque style where
 * the constraint model is built by side-effects from constructors.
 * I've kept it this way to avoid deviating too much from the original
 * implementation.
 */
```

図 2 テキストで記述された論文引用コメントの例（gecko-dev より）

```
/*   A. Abrham, ACM Transactions on Mathematical Software, Vol 11 #3, */
```

図 3 テキストで記述された短い論文引用コメントの例（gecko-dev より）

論文誌の引用が多いと明らかにした．目視による分類で調査を行った理由として，機械的な検出が 2 つの理由で困難であるからとしている．一つは論文引用の形式が統一されていないこと，もう一つは論文誌名と無関係なコメントがヒットすることとしている．

3. 論文の引用記法

論文引用の記述方法は，通常，論文誌等の媒体ごとに定められている．たとえば情報処理学会論文誌の原稿執筆案内 [7] では，論文タイトルは原題をそのまま記述すること，著者名については少なくとも先頭の 3 名を示すことが指定されている．一方で，国際会議における予稿集の名称には会議の正式名称と略称のどちらを使ってもよいというように，書き手に対するある程度の自由度も与えられている．

ソースコードコメントにおける論文引用の記述においては，このようなガイドラインを与える組織が存在しないため，開発者それぞれが自由に記述することになる．本研究の実施にあたってソースコードコメントを手作業で分析した範囲では，BibTeX で利用される文献情報ファイルの形式と，論文に関する情報を並べる形式の 2 つが主な記述方法である．

3.1 BibTeX 記法

BibTeX 記法は，著者名，タイトル，出版年などの情報を，属性の名称と内容の組という構造化されたテキストで記載する方式である．ソースコードコメントへの BibTeX 記法の出現例を**図 1** に示す．例は gecko-dev から得られた[*2]．近年の論文誌であれば正式なデータが学会や電子図書館等のシステムから取得可能であり，通常は，引用している文献を一意に識別することのできる情報を格納している．

テキストを構造化するためのキーワードがコメント中に出現するため，本研究ではこの記法の論文引用についてはキーワード検索による検出を行うことを前提とし，本研究で提案する自然言語解析に基づく手法の適用対象とは考えない．

[*2] https://github.com/mozilla/gecko-dev/blob/master/third_party/aom/av1/common/odintrin.c，最終アクセス：2018/11/05

3.2 論文に関する情報を並べる記法

ソースコードコメント上の論文引用で多く見られる記法で，論文のタイトルや著者名，掲載誌などの情報を並べていく方式である．**図 2**（gecko-dev より[*3]）はそのようなソースコードコメントの 1 つであり，ソースコードがある論文に記述されたアルゴリズムの実装であることを述べ，論文のタイトル，著者名，掲載誌の情報を記述している．

コメントにおける論文引用は，記述した開発者によって，記される項目の種類や量，順番や区切り方は様々である．**図 3**（gecko-dev より[*4]）に示すように，タイトルを省略しても，著者名，論文誌名，ページ番号から論文を識別するような記述を行っても，該当論文を識別するという目的は達成されている．そのため，特定のキーワードだけで論文引用を検出することは困難である．本研究ではこのような引用を含むコメント群を，自然言語処理の技術を用いて，自動的に識別することを目指す．

4. アプローチ

ソースコードコメントから機械的に論文の引用を検出するためにデータセットを用意し，固有表現抽出を用いる．固有表現抽出とは，自然言語処理の手法を用いて，文中から固有名詞を抽出する手法である [2][5]．固有表現抽出により，コメント文から論文のタイトルや著者名，論文誌名などの抽出を試みる．固有表現抽出を行うためにモデルを用意し，固有表現について学習させる．学習のために引用を含むコメントと，コメント文のうち，どの語句がタイトルや著者名などに該当するかをデータセットとして用意する．学習したモデルが正しく固有表現を抽出し，論文の引用を検出できるか評価する．評価のために引用を含むコメントと含まないコメントを用意し，それぞれからの抽出結果を分析する．

[*3] https://github.com/mozilla/gecko-dev/blob/master/js/src/octane/deltablue.js，最終アクセス：2018/11/05

[*4] https://github.com/mozilla/gecko-dev/blob/master/intl/icu/source/i18n/decNumber.cpp，最終アクセス：2018/11/05

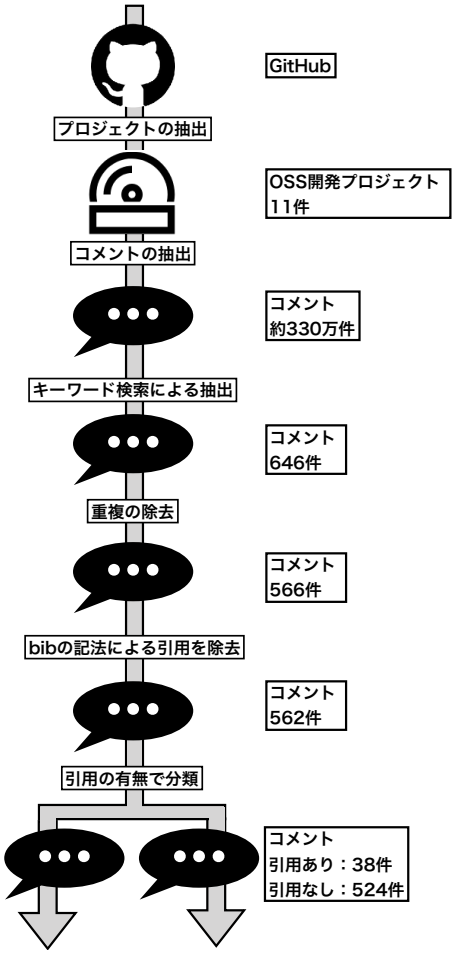

図 4　データセット作成の工程

4.1　spaCy

検出のために自然言語処理ライブラリである spaCy[*5] を用いる．spaCy は Python で利用できる自然言語処理ライブラリで，商業的な利用を想定して作られている．他の自然言語処理ライブラリと比較すると，固有表現抽出を含め，多くの処理が高速に行える．

4.2　データセットの作成

固有表現を行う spaCy のモデルを学習させるために，ソースコードコメントのデータセットを用意する．データセット作成の工程を図 **4** に示す．

まず，GitHub 上に公開されている OSS 開発プロジェクトを収集し，それぞれの 2017 年の特定のバージョンから，C/C++, Java, JavaScript のソースファイルを収集した．コメントの抽出そのものは ANTLR4 [4] の字句解析器を用いて，各言語の文法に従って行った．このコメント抽出の結果，11 個のプロジェクトから約 330 万件のコメントを得た．対象としたプロジェクトと抽出したコメント数を表 1 に示す．

ソースコードに含まれるコメント全体から見て，論文を引用するコメントはごく一部である．本研究ではデータセットを手作業で作成するため，キーワード検索による絞り込みを行った．キーワードは先行研究 [6] が明らかにした，ソースコードコメントでの引用回数が多い論文誌名の接頭辞として出現する学会名 “ACM” を用いた．検索した結果，646 件のコメントがヒットし，完全一致で重複するコメントを除去すると 566 件となった．次に，BibTeX 記法による引用を含むコメントを除去した．BibTeX 記法による引用は，通常のテキストによる引用形式と大きく異るため学習に不向きであると考えられるためである．この結果，コメントが 562 件となった．得られたコメントの中に，論文誌名である “ACM” を含むが，論文の引用を含まないコメントが多く含まれる．このため，これらを引用の有無について目視により分類した結果，引用ありのコメントが 38 件，引用なしが 524 件となった．これより，キーワード検索のみでは十分に論文引用を検出できないと確認できる．これら 38 件と 524 件のコメントをデータセットとする．

表 1　データセットに含まれるプロジェクトの一覧

プロジェクト名	コメント数
android-easymock	385
android-junit	707
android-slf4j	1,756
bsort	6
easymock	1,058
gecko-dev	1,884,684
junit4	1,317
libpng-code	7,735
linux	1,340,863
postgresql	71,605
slf4j	1,657
合計	3,311,773

4.3　データセットの前処理

得られたコメントから spaCy のモデルを構築するために，以下の前処理を行う．

(1) 前処理対象のコメントを用意する．
(2) すべての単語を小文字にする．
(3) 空白および記号を手がかりとしてトークン列に分解する．
(4) 一部の記号と一致するトークンをを除去する．
(5) 空白文字で結合する．
(6) 前処理結果が同一となったコメントを除去する．
(7) 固有表現を手動でラベル付けする．

工程の (4) で除去する記号は `*`，`/`，`’’`，`‘‘`とした．C/C++, Java, JavaScript においてコメントを表現するための記号と，タイトル等を囲む引用符である．

加工の工程 (1) から (5) までの例を図 **5** に示す．例に用

[*5] `https://github.com/explosion/spaCy`，最終アクセス：2018/11/05

```
(1) // integers." ACM Sigplan Notices 45.6 (2010): 233-243.
(2) // integers." acm sigplan notices 45.6 (2010): 233-243.
(3) ['//', 'integers', '.', "'", 'acm', 'sigplan', 'notices', '45.6', '(', '2010', ')', ':', '233-243', '.']
(4) ['//', 'integers', '.', 'acm', 'sigplan', 'notices', '45.6', '(', '2010', ')', ':', '233-243', '.']
(5) // integers . acm sigplan notices 45.6 ( 2010 ) : 233-243 .
```

図 5 前処理の各段階におけるテキストの状態

```
(6)
// integers . acm sigplan notices 45.6 ( 2010 ) : 233-243 .
 年：2010
 ボリューム：45
 論文誌名：acm sigplan notices
 ナンバー：6
 ページ：233-243
```

図 6 ラベルの割り振り例

いるコメントは gecko-dev より抽出した[*6]．これらの加工により，単語と記号の結びつきの解消や，学習に不要と考えられる記号の除去などができる．工程 (6) における重複の除去は，データセットの作成段階でも行っていたが，前処理を行うことで重複コメントが増幅するケースがあるため行う．これはコメントのコピー&ペーストを利用した場合の書き足しや，記号による表記ゆれなどが該当していると考えられる．これにより，引用を含むコメントは 2 件，引用を含まないコメントは 4 件除去した．この結果，引用ありが 36 件，引用なしが 520 件となった．

工程 (7) として，論文引用文のうち，どの語句がどの固有表現に該当するかを示すラベルを付ける．本研究では，BibTeX の記法で記載される項目を参考に，以下の 9 つのラベルを用いた．

- タイトル
- 著者
- 年
- 月
- ジャーナル名
- ボリューム（巻）
- ナンバー（号）
- ページ番号
- URL

工程 (6) のラベルの割り振り例を**図 6** に示す．

4.4 モデルの学習と固有表現の抽出

引用を含むコメントを用いて，モデルの学習と性能の評価を行うために学習データとテストデータに分割する．分割は**図 7** に示すように，以下の手順で行う．

(1) 引用を含まないコメントを「引用なし」テストデータにする．
(2) 引用を含むコメントを 10 分割する．
(3) (2) で分割したうちの 1 つを「引用あり」テストデータにする．
(4) (2) で分割したうちの残り 9 つを「学習データ」にする．
(5) 学習データを用いてモデルの学習を行う．
(6) (1) と (3) で得られた各テストデータを用いてモデルを評価する．
(7) (3) で利用するコメントを，学習データにしていない異なるものに変更する．
(8) (3) から (7) を繰り返す．

引用なしテストデータと引用ありテストデータの 2 つをテストデータにすることで，引用の有無による固有表現抽出の違いを比較する．(7) によるテストデータと学習データの変更は，より多くのコメントをテストデータに用いて評価するために行う．

[*6] https://github.com/mozilla/gecko-dev/tree/master/gfx/angle/src/tests/third_party/rapidjson/include/rapidjson/internal/diyfp.h，最終アクセス：2017/10/24

表 2 引用有りテストデータの結果例

ラベル	テスト結果	正しいラベル
年	2010	2010
ボリューム	45.6	45
論文誌名	acm sigplan	acm sigplan notices
ナンバー	-	6
ページ	233-243	233-243

5. 結果と考察

5.1 抽出例

引用ありテストデータによる固有表現抽出の結果を**表 2** に示す．表 2 で用いるコメント例は前処理，ラベル付で用いた図 5，図 6 と同じコメントである．

表 2 より，この例では年，ページのラベルは正しく抽出できているが論文誌名，ボリュームは抽出部分のずれがあり，ナンバーは抽出できない結果となった．他の抽出結果を確認すると，例と同様に完全な抽出はできていないが部分的に正しい抽出が多く見られた．

5.2 固有表現抽出結果の概要

引用ありテストデータ，引用なしテストデータを用いた固有表現抽出結果の概要を**表 3** に示す．

表のコメント数はテストとして利用したコメントの総数を示す．引用有りテストデータと学習データを変更し，10 回検出を行ったが，引用無しは毎回 520 件のコメントを評価に利用したため合計 5,200 件となった．検出ラベル数は，各テストデータから得られたラベルの総数を示す．ここではラベルの成否は考慮していない．引用ありテストデータ

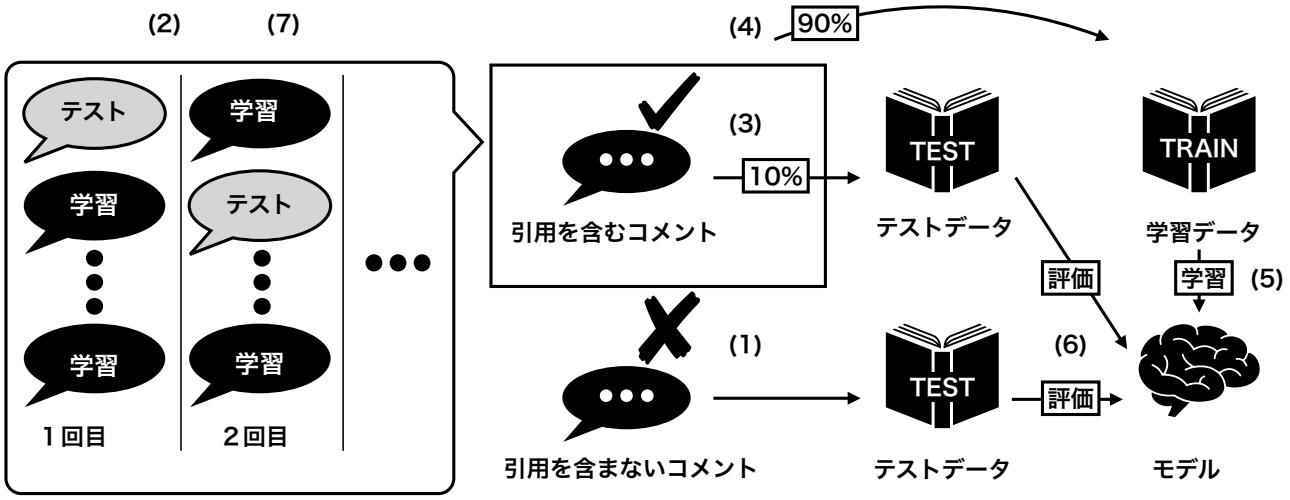

図 7　学習と評価の概要図

表 3　結果の概要

	引用あり テストデータ	引用なし テストデータ
コメント数	36	5,200
検出ラベル数	211	5,853
平均検出ラベル数	5.9	1.1
検出ありコメント数	35	2,009
検出有り平均ラベル数	6.0	2.9

ではコメント数が 36 件，検出ラベル数が 211 件であったため，コメント 1 件あたりの平均検出ラベル数は 5.9 件となった．同様に，引用なしテストデータではコメント数が 5,200 件，検出ラベル数が 5,853 件であったため，平均検出ラベル数は 1.1 件となった．これより，引用を含むコメントからの固有表現抽出のほうが，より多くのラベルを検出できたことがわかる．また，ラベルを 1 件以上検出できた場合を抽出すると，引用有りテストデータでは抽出したコメント数が 35 件となり，コメント 1 件あたりの平均検出ラベル数は 6.0 件となった．同様に，引用なしテストデータではコメント数が 2,009 件，平均検出ラベル数は 2.9 件となった．これより，ラベルが 1 件以上検出できた場合，引用なしテストデータの該当件数が半分以下にまで減少した．すなわち，コメントからラベルを 1 件以上検出した場合に，論文引用している可能性が高いと判断した場合，キーワード検索の結果から，ヒット数の大部分を除外できる．また，引用ありテストデータからラベルの検出数が 0 件だったコメントは 1 件だった．したがって，ラベルの検出が 1 件以上を条件としたフィルタによる論文引用の検出もれは，0 ではないが多くはない．このため，本研究による提案手法は有効であると考えられる．

テストデータを用いた検出結果を F 値（F-measure）を用いて評価する．F 値とは，テストの結果を評価する指標の一つで，精度（Precision），再現率（Recall），F 値を用いて表す．精度は，正と予測した結果のうち，実際にである割合を示す．再現率は，実際に正であるうち，正と予測した割合を示す．F 値は精度と再現率の調和平均を示す．各尺度は TP，FP，TN，FN の 4 つの値によって導き出される．TP は正であると予測し，実際に正であった数，FP は実際は負であった数を示す．TN は負であると予測し，実際に負であった数，FN は実際は正であった数を示す．各指標は次の式で導き出される．

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

予測を，1 つのコメントから 1 つ以上のラベルが検出できた場合，そのコメントは論文を引用しているものとする．このとき，表 3 より各値，指標は次のようになる．

$$TP = 35$$
$$FP = 2,009$$
$$TN = 3,191$$
$$FN = 1$$
$$Precision = 0.017$$
$$Recall = 0.972$$
$$F - measure = 0.034$$

この結果，*Precision* が低く *Recall* が高いと分かる．すなわち，実際に論文を引用していないコメントの多くを，論文を引用していると予測するが，実際に論文を引用しているコメントを高い割合で検出できる．本研究においては，OSS 開発プロジェクトにおける論文引用の網羅的な把

握を重視しているため，*Precision* よりも *Recall* を重視するべきだと考える．このため，*Precision* と *Recall* の調和平均である $F-measure$ の値は低いが，*Recall* の値が高いため，目的に応じた検出ができると言える．

5.3 予測条件の分析

論文を引用していると予測する条件を，1 つのコメントから 1 つ以上のラベルを検出したときとしているが，条件とするラベルの検出数を変更した場合について分析する．

条件とするラベルの検出数を 1 から 10 まで変更した場合について，各値，指標を表 5.3 に示す．表中の条件は，引用していると予測する条件のラベルの検出数を，*Prec* は *Precision* を，*Reca* は *Recall* を，F は $F-measure$ を示す．

表 4 検出条件を変更した場合の指標

条件	TN	FP	FN	TP	$Prec$	$Reca$	F
1	3,191	2,009	1	35	0.017	0.972	0.034
2	4,160	1,040	3	33	0.031	0.917	0.060
3	4,585	615	7	29	0.045	0.806	0.085
4	4,797	403	15	21	0.050	0.583	0.091
5	4,963	237	20	16	0.063	0.444	0.111
6	5,042	158	20	16	0.092	0.444	0.152
7	5,087	113	24	12	0.096	0.333	0.149
8	5,118	82	26	10	0.109	0.278	0.156
9	5,143	57	30	6	0.095	0.167	0.121
10	5,153	47	30	6	0.113	0.167	0.135

表 5.3 より，*Precision* と *Recall* の調和平均を示す $F-measure$ が最も高くなる条件は，ラベルの検出数が 8 のときである．条件とする検出数を増加させることにより，$F-measure$ の値は高くなる傾向にあるが，*Recall* の値は条件とする検出数が 4 以上としたときから大幅に減少している．一方で *Precision* の値は条件とする検出数に合わせて上昇しているが，大きな上昇は見られない．*Recall* を大きく減少させず，*Precision* を増加させるためには条件とする検出数，すなわち 1 つのコメントから 2〜3 のラベルが検出できた場合に設定すると，論文を引用するコメントの検出漏れが少なく，無関係なコメントを除外できると考えられる．

5.4 予測ラベルの内訳

引用ありテストデータ，引用なしテストデータを用いたラベルの検出結果の内訳を表 5.4 に示す．表 5.4 中の比率は，各テストデータから得られたラベルにおける割合を示す．また，表 5.4 をグラフにしたものを図 8 に示す．図 8 中の黒色のバーは引用ありテストデータを，灰色のバーは引用なしテストデータにおける各ラベルの比率を示す．

表 5.4 と図 8 より，引用ありテストデータと引用なしテストデータを用いたラベルの検出結果を比較すると全体的

表 5 ラベルの内訳

	引用有りテストデータ		引用なしテストデータ	
ラベル	件数	比率	件数	比率
タイトル	19	9%	801	14%
著者	56	27%	1042	18%
年	47	22%	2075	35%
月	17	8%	186	3%
ボリューム	6	3%	56	1%
論文誌	36	17%	985	17%
ナンバー	3	1%	58	1%
ページ	21	10%	503	9%
URL	6	3%	147	3%

に比率が高い項目，低い項目は一致している．しかし，年や著者のラベルは約 10%，タイトルと月のラベルは約 5%の差が出ている．このため，コメントから検出したラベルの内容に応じて重み付けをすることで，論文引用の抽出精度が高まると期待できる．

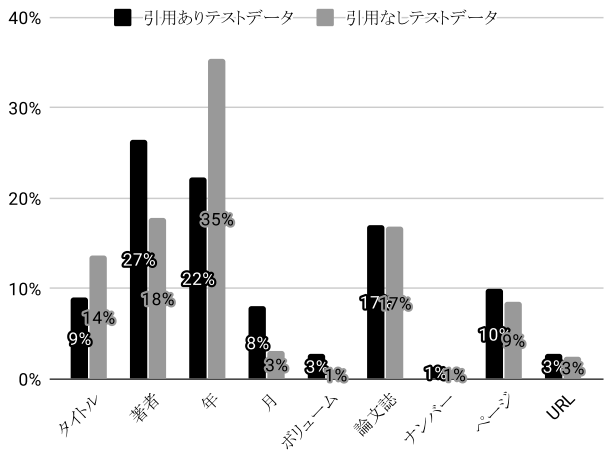

図 8 テストデータに含まれるラベル比の比較

6. 今後の展望とまとめ

本研究はソースコードコメント上の論文引用を検出するために，固有表現抽出を用いて，コメントから論文引用を示すラベルの抽出に取り組んだ．固有表現抽出のために自然言語処理ライブラリである spaCy を利用し，11 の OSS 開発プロジェクトからデータセットを用意した．データセットは約 330 万件のコメントから，ソースコードコメント上で引用頻度が高い論文誌名でキーワード検索を行い，検索結果を目視で分類した．この結果，引用を含むコメント 36 件と引用を含まないコメント 520 件が得られた．引用を含むコメントの引用部分に，タイトルや著者などのラベルを付けた．得られたコメントを学習データとテストデータに分割し，学習データを用いて固有表現抽出を行うモデルを学習させ，テストデータを用いてモデルを評価した．この結果，モデルによって引用を含むコメントの多くを検出できたが，引用を含まないコメントも多く含むと分かった．

また，検出するための条件や，ラベルの重み付けなどにより精度の向上が期待できると確認できた．

今後の展望として，本論文では検出したラベルの正誤について触れていないため，考慮した分析を行いたい．また，利用したデータセットについて，引用を含むコメントが少ない点や，単一の論文誌名をキーワードとして含むものに限定している．このため，モデルの性能を正しく評価できていない懸念がある．したがって，データセットの量の調節や見直しが望まれる．

謝辞 本研究は JSPS 科研費 16H05857，17H00731，18KT0013 の助成を受けたものです．

参考文献

[1] Bhatia, S., Tuarob, S., Mitra, P. and Giles, C. L.: An Algorithm Search Engine for Software Developers, *Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, SUITE '11, New York, NY, USA, ACM, pp. 13–16 (online), DOI: 10.1145/1985429.1985433 (2011).

[2] Grishman, R. and Sundheim, B.: Message Understanding Conference-6: A Brief History, *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, Stroudsburg, PA, USA, Association for Computational Linguistics, pp. 466–471 (online), DOI: 10.3115/992628.992709 (1996).

[3] Marcus, A. and Maletic, J. I.: Recovering Documentation-to-source-code Traceability Links Using Latent Semantic Indexing, *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, Washington, DC, USA, IEEE Computer Society, pp. 125–135 (online), available from ⟨http://dl.acm.org/citation.cfm?id=776816.776832⟩ (2003).

[4] Parr, T.: *The Definitive ANTLR 4 Reference*, Pragmatic Bookshelf, 2nd edition (2013).

[5] Tjong Kim Sang, E. F. and De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, pp. 142–147 (2003).

[6] 小西文章，門田暁人， 畑秀明，松本健一：オープンソースソフトウェアにおける学術論文の引用状況の分析，ソフトウェア工学の基礎, 日本ソフトウェア科学会 FOSE2013，pp. 179–184 (2013).

[7] 情報処理学会：論文誌ジャーナル（IPSJ Journal）原稿執筆案内，`https://www.ipsj.or.jp/journal/submit/ronbun_j_prms.html` (2017).