5K-07

ソースコードの字句を利用したコメントの分類

池上 綾乃† 馬場 睦也‡ リュウ ショウウ‡ 波多野 賢治†
†同志社大学 文化情報学部 ‡同志社大学大学院 文化情報学研究科

1 はじめに

ソフトウェアの保守作業では，プログラマは既に完成した膨大な量のソースコードを理解し編集する必要があり，多くの時間がプログラム理解のために消費される. したがって，可読性の低いソースコードは作業効率を低下させる恐れがある．このような背景から，ソースコードの可読性を高める方法の一つである，コメントに着目した研究が行われている [1]．コメントとはソースコード中に処理の説明を記述することができ，プログラミング能力に関係なく処理内容の理解を補助するものである．そのため，記述コストの削減に向けたコメントの自動生成に関する研究が行われている．このような研究では，独自に定義したコメントのカテゴリを基に分類したコメントが必要となるが，膨大な数のコメントを人手で分類することはコストが大きく，現実的ではない．

そこで本研究では，先行研究で定義されたカテゴリに基づいて，コメントの自動分類に関する研究を行う．Pascarella et al. はコメントのカテゴリを定義し，機械学習を用いた分類手法を提案したが，精度に課題があり，モデルの実用化には至っていない [1]．それは，コメントの中には処理に関する記述がなされたものとそうでないものがあるにも関わらず，ソースコードが考慮されていないためであると考えられる．そこで，分類精度向上のためにソースコードの特徴量を新たに考慮することを提案する．

2 先行研究

Pascarella et al. は，用途の異なるコメントのカテゴリを定義することを目的に，網羅的かつ排他的に分類できるカテゴリを大カテゴリ，小カテゴリに分けて定義し，機械学習によるコメント分類モデルの構築を試みた [1]．コメントに含まれる単語の頻度を特徴量として，ランダムフォレスト，ナイーブベイズを用いて分類した結果，データが少ないカテゴリであっても分類できるナイーブベイズの有用性を主張している．しかし，データセットによって精度がばらつく問題がある．これはソースコードと関連したカテゴリがあるにも関わらず，ソースコードが考慮されていないためであると考えられる．

Classifying Code Comments using Java Lexemes
†Ayano IKEGAMI ‡Tokiya BABA ‡Zhaoyu LIU †Kenji HATANO
†Faculty of Culture and Information Science, Doshisha University
‡Graduate School of Culture and Information Science, Doshisha University

3 提案手法

本研究ではコメントの特徴量に加え，ソースコードとコメントの関連性として，コメント中の単語とソースコード中の字句で共通して使用されている語を扱う．これは，宣言された変数やメソッドに関して，コメントで言及している事例が存在するためである．しかし，コメントとソースコードの字句は，同じ語ではなく，類義語で表現する場合があるため，単純に共起を集計するだけでは情報を取り損ねる場合がある．そこで，本研究では単語の類義語推定を用いた特徴量の作成方法を提案する．

単語の類義語推定には，本研究ではコメントに関連する単語を追加するためにコメントの単語から類義語の推定を行うモデル，ソースコードから出現しうるコメントの単語の推定を行うモデルの二つを使用する．コメントでの類義語推定モデルは，Word2Vec[2] にコメントのみを入力で与えることで構築し，類義語推定に用いる．なお，Word2Vec は一つの単語から類義語を推定するタスクで用いられる Skip-gram を採用し，必要なウィンドウサイズや次元数などのハイパーパラメータはデフォルト値を用いた．

ソースコードとコメントを用いたモデルは，行がコメントに出現する単語，列がソースコードに出現する字句である重み行列で表現する．要素はソースコードとコメントが共起する確率 w_{ij} を式 (1) で算出する．

$$w_{ij} = P(c_i|p_j) = \frac{c_i \cap p_j}{p_j} \tag{1}$$

i は重み行列の行番号，j は重み行列の列番号を表し，c_i はコメントで出現する単語，p_j はソースコードで出現する字句を表す．つまり，w_{ij} はソースコードの字句 p_j が出現した時に，コメントの単語 c_i が出現する事後確率を表す．ソースコードをベクトル化する手法 [3] や単語をベクトル化する手法 [2] は存在するが，これらを対応させてベクトル化する手法は見られなかったため，このような形で表現する．コメントと対応するソースコードの範囲は，コメントが括弧の直下にある場合はコメントが含まれるスコープ全体，括弧の直上にある場合は，直下のスコープ全体，ソースコードの横にある場合はその文 1 行を対象とする．

表 1: 小カテゴリの分類精度

	要約	一部の詳細	背景	非推奨勧告	使用方法	例外の理由	ToDo	不完全	コメントアウト	目印	フォーマッター	ライセンス	オーナーシップ	ポインタ	自動生成されたコメント	その他
既存:適合率	0.60	0.05	0.07	0.39	0.65	**0.81**	0.12	**0.20**	0.13	**0.95**	**0.88**	0.63	0.47	0.50	0.43	0.00
提案:適合率	**0.68**	**0.09**	**0.11**	**0.82**	**0.68**	0.74	**0.25**	0.12	**0.27**	0.92	0.07	**0.75**	**0.60**	**0.78**	**0.80**	**0.01**
既存:再現率	0.32	0.29	0.10	**0.81**	0.70	0.46	0.15	**0.98**	0.01	**0.98**	**0.14**	**0.98**	**0.42**	**0.47**	**1.00**	0.00
提案:再現率	**0.49**	**0.58**	**0.13**	0.52	**0.74**	**0.51**	**0.20**	0.94	**0.09**	**0.98**	0.01	**0.98**	0.24	0.39	0.99	**0.05**
データ件数	4,200	199	256	54	2,105	244	978	87	328	978	53	828	446	1,029	205	19

この二つのモデル構築には，過去の研究で収集された Java プロジェクトに含まれる，1,712,819 個のソースファイルからコメントとソースコードを取得した [3].

二つの類義語推定モデルを用いて，コメントに出現する単語の類似度と，分類するコメントとソースコードに含まれる単語から他の単語が出現する確率を取得する．そして，これらの値をコメントに出現する単語の頻度と合わせて特徴量を作成する．分類モデルの構築には，有用性が主張されているナイーブベイズを用いる．先行研究との比較により，本研究で提案する特徴量の作成方法が有効かどうかを検証する．また，カテゴリの非推奨勧告，使用方法，不完全，目印，ライセンス，自動生成されたコメントについては，特定の単語の出現によって多くのコメントが分類可能だったため，特定の単語を含むかどうかによる検出処理を分類器に入力する前に行う．

4 評価実験

本研究で追加した特徴量の有効性を評価するため，分類精度を算出する．データは先行研究で作成された，6 種類の Java プロジェクトで構成されるデータセットを使用し，カテゴリを正解ラベルとして使用した [1]．特徴量作成のために，前処理でスペースと改行文字による単語分割，半角英数字記号以外の文字の除去，ストップワードの削除，キャメルケースによる単語分割，単語の語幹以外を削除するステミングを行った．

精度の評価には，先行研究で用いられた適合率と再現率を用いる．データの分割はモデルの実用性を評価するため，評価するプロジェクトのデータが学習データに含まれないように行う．したがって，6 種類の Java プロジェクトのうち，5 種類のプロジェクトでコメント分類モデルを構築し，残り 1 種類を評価データとしてモデルに適用することで評価を行う．また，表 1 で示すようにカテゴリによってデータ件数が不均衡のため，そのまま学習すると分類結果に影響が出てしまう．そのため，先行研究と同様にランダムオーバーサンプリングと呼ばれる手法 [4] によって，データの不均衡性を解消し，実験を行った．

表 1 に小カテゴリで分類した結果を示す．この結果から，ほとんどのカテゴリに対して精度の向上が見られたことがわかる．特に，処理を説明する要約カテゴリのような処理と関連し，かつ特定の単語の出現での分類が難しいカテゴリに関しては精度の向上が見られた．これは，提案手法によってソースコードとの関連を取ることで分類に寄与できたためであると考える．一方で，ソースコードの実装背景に関するコメントである背景カテゴリについては，あまり精度が向上しなかった．これは，実装背景に情報がソースコードから読み取れず，コメントとソースコードの関連がうまく取れなかったためであると考える．したがって，ソースコードの関連以外にも考慮すべき特徴量が残されていると考えられる．

5 おわりに

本研究ではソースコードにおけるコメントの自動分類を行う手法を提案した．類義語推定を行うことによって，多くのカテゴリで精度の向上が見られたが，実用的なモデルの構築には課題が残る結果となった．今後は Word2Vec のハイパーパラメータの調整を行い，さらなる精度向上を目指す．

謝辞

本研究の一部は JSPS 科研費 JP18H03342 の助成を受けたものである．

参考文献

[1] Luca Pascarella, Magiel Bruntink, and Alberto Bacchelli. Classifying Code Comments in Java Software Systems. *Empirical Software Engineering*, Vol. 24, No. 3, pp. 1499–1537, 2019.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, Vol. 26, pp. 3111–3119. Curran Associates, Inc., 2013.

[3] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2Vec: Learning Distributed Representations of Code. *Proc. ACM Program. Lang.*, Vol. 3, pp. 40:1–40:29, 2019.

[4] Nitesh V. Chawla. Data Mining for Imbalanced Datasets: An Overview. In *Data Mining and Knowledge Discovery Handbook*, chapter 40, pp. 875 –886. Springer, 2009.