The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

複数の作業文脈の統合深層モデルによるコミットメッセージ生成

Neural commit message generation integrating multiple work context encoders

岩永 知裕 *1 Tomohiro Iwanaga
眞名垣 優希 *1 Yuki Managaki
谷垣 宏一 *1 Koichi Tanigaki

*1福井工業大学
Fukui University of Technology

This paper proposes a new neural network model for commit message generation. Previous studies simply apply machine translation models to convert code diffs to commit messages, which lacks the mechanism to learn how to write intentions of code modifications, although it is often required in desirable commit messages. The proposed model introduces multiple encoders to efficiently capture different levels of working contexts, such as code modifications and original bug reports. Experimental results using GitHub dataset showed degradation when adopting issues as additional contexts, which suggests the commit style specificity of the dataset.

1. はじめに

ソフトウェア開発の現場では、過去のソースコードがしばしば参照される。例えば、ソフトウェア改修においてバグが混入したとき、あるいは仕様変更が生じたとき、過去のソースコードから目的のバージョンを特定し参照する作業が発生する。こうした作業を容易にするため、バージョン管理システムが広く用いられている。バージョン管理システムとは、開発物の変更履歴を記録管理するシステムである。バージョン管理システムは、ソースコードの変更をコミットという単位で管理している。各コミットには、ソースコードの変更内容や目的を自然言語で説明した文（コミットメッセージ）が紐づけられている。コミットメッセージを読む事で、過去の膨大なコミットから、必要なコミットを特定する事が容易になる。

しかし、有益なコミットメッセージを作成することは必ずしも全ての開発者にとって容易なことではない。この問題に対して、コード差分からコミットメッセージを自動生成する研究が試みられている [Jiang 17]。しかし、コード差分のみを入力とする手法では、コード変更の意図のような説明をメッセージ中に含めることが難しい。そこで本研究では、より有益で高品質なコミットメッセージ自動生成を狙い、問題報告とコード差分の 2 つの文脈を利用したメッセージ生成方式を提案する。

2. 関連研究

ソースコードを入力として自然言語を出力する研究は様々存在している。Xing らはソースコードから、コメント文を生成する手法を提案している [Schuster 18]。Jayavardhan らはプログラムの実行中に出力される実行ログのメッセージを自動生成する方式を提案している [Pinji 18]。Alireza らは android アプリのメソッドに対して要約文を生成している [Alireza 18]。これらの研究ではソースコードのみをメッセージ生成の文脈として用いている。それに対して本研究では、複数の文脈を複合的に用いた文生成を試みる。

3. 提案法

我々は、有益なコミットメッセージには変更意図などの説明が記載されており、そうした説明は、修正したコードのリテラルな内容よりも、変更に至った経緯のような抽象的文脈に強く依存して記載されると考える。そのような文脈を捉えるのに有効な情報として、問題報告 (Issue) を用いることができる。バージョン管理システムを利用した開発現場では、ソフトウェアの問題点や発見したバグ、追加すべき機能などを問題報告という機能を用いて共有している。開発者は、これら問題報告を参考にしてソフトウェアの変更を行う。そこで、本研究ではコミットメッセージ生成に問題報告を利用するモデルを提案する。

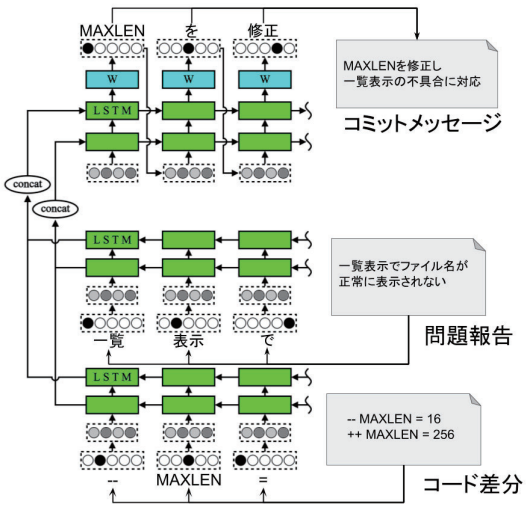

図 1　提案モデル

提案モデルを図 2 に示す。先行研究 [Jiang 17] で用いられた機械翻訳のモデルが、コード差分に対するエンコーダ 1 つと，コミットメッセージ生成のためのデコーダ 1 つから構成されるモデルであったのに対し、提案モデルでは 2 つの作業文脈からコミットメッセージを生成するため、2 つのエンコーダを持つ点が特徴である。このモデルは bi-LSTM[Schuster 18] から発想を得たもので、エンコーダ部を 2 つにした seqence to seqence（以下 seq2seq）[Sutskever 14] モデルである。bi-LSTM では、同一文の入力順序を逆にして入力していたが、提案モデルでは、問題報告とコード差分の単語列をニューラルネットの階層に逐次的に入力している。それぞれの単語列を入力後、メモリーセルを統合 (concatenate) しデコーダに渡すことで入力と出力が 2 対 1 のコミットメッセージ生成が可能となっている。

連絡先: 岩永 知裕, 福井工業大学, tomo1996722@gmail.com

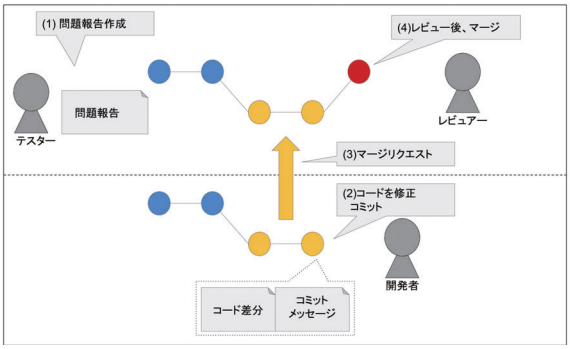

図 2　GitHub における開発の流れ

4.　評価実験

本研究によるメッセージ生成の有効性を評価するため評価実験を行った。

4.1　実験データの収集

本研究で使用したデータは GitHub*1 から取得した。GitHub では、大人数でコードを編集してもバージョンを管理できるよう、図 4.1 に示す作業フローでコード変更がコミットされる。まずユーザやテスターは、バグや不足機能を見つけると問題報告を作成する。この問題報告には問題を要約したタイトルと、その詳細が記載されている。また問題報告にはアサイン機能も存在し、実際にこの問題に対処する開発者の割り当てを行う事が出来る。問題報告に割り当てられた開発者は、この問題報告を参考にソースコードの変更を行い、1 回ないし数回のコミットを行う。このとき他の開発者が平行して開発を進められるよう、履歴を分岐させる。開発者は、修正に必要なコミットが全て完了すると、分岐後の全てのコミットを分岐元にマージするよう要請する。要請されたコミット群をマージしても問題が無いか、レビュアーが確認を行い、問題がなければこれらのコミットを履歴元にマージする。この時、問題報告に対して、マージリクエストの ID が採番される。本研究では、これらマージリクエスト ID を問題報告から抽出し、マージリクエスト内の全コミットと問題報告文を関連づけた。この関連から問題報告、コード差分、コミットメッセージの 3-tuple を作成した。

4.2　実験設定

メッセージ生成精度の定量評価尺度には BLEU を用いる。[Jiang 17] と同様に、生成されたコミットメッセージとオリジナルの文から BLEU スコアを算出し、各手法による精度を比較する。実験では 2 つのデータセットを用いる。これらはいずれも Java 言語を使用する OSS プロジェクトのデータであり、ランキング Top 100 プロジェクトのデータセットと、Top 1000 プロジェクトのデータセットとを用いた。表 1 に両データセットの詳細を示す。ただし問題報告とコード差分はメモリの制約から先頭 200 単語のみを用いた。語彙は頻度上位 4,000 語で生成した。ハイパーパラメータ探索用の開発データと評価用のデータは、それぞれ 1,000 セットずつランダムで抽出した。ただしハイパーパラメータの探索は Top100 のデータセットのみで行い、Top1000 のデータセットでは Top100 と同一のハイパーパラメータを使用した。これらデータセットを用いて複数エンコーダ seq2seq、およびコード差分のみを入力した seq2seq、コード差分と問題報告を連結して入力した seq2seq、の 3 つを比較し、問題報告を利用する効果と、複数エンコーダ seq2seq の有効性を確認する。

*1　https://github.com/

表 1　データセット：diff/issue/コミットメッセージ

–	Top100	Top1000
3-tuple	18,183	46,257
bytes	27M/14M/812K	67M/34M/2.1M
語彙	40000/24543/9704	40000/40000/40000

表 2　各手法によるメッセージ生成精度 (BLEU スコア)

–	Top100	Top1000
複数エンコーダ (提案手法)	0.258	0.179
コード差分のみ	0.313	0.243
コード差分＋問題報告	0.299	0.207

4.3　実験結果

実験結果を表 2 に示す。BLEU スコアはコード差分のみを用いた seq2seq が最も高く、次に問題報告とコード差分を結合した seq2seq、最も低いのが提案モデルという結果になった。またデータセット間においては、すべての手法において、Top100 のデータセットにおいて Top1000 を大きく上回るスコアが得られた。本実験結果からはコミットメッセージ生成に問題報告を用いる有効性は確認できなかった。

5.　考察

今回の実験で提案法により期待する性能が得られなかった要因として、GitHub における開発が、我々の想定する流れと異なっていたことが考えられる。GitHub における開発では、メインの履歴に対して直接コミットされるのではなく、分岐した履歴に対しローカルにコミットしておいて、最後にメインにマージリクエストする形式で開発が進む（図 2）．このような二段階の作業フローでは、マージリクエストのメッセージにおいては元の問題報告を強く意識し、それに対応したことを示すメッセージが作成される傾向があると考えられるが、それ故、前段階のコミットでは問題報告との関係が希薄なメッセージがローカルに作成される可能性がある．今後、GitHub 以外の、チケット駆動開発のプロジェクトデータや、企業など一定の統制の元で作成されたデータにも適用し、提案法の有効性を検証したい。

謝辞

本研究は三菱電機株式会社の協力を得たものです。

参考文献

[Jiang 17] Jiang, D et al.'s: "Automatically Generating Commit Messages from Diffs using Neural Machine Translation", Proc. of ICAE. (2017)

[Sutskever 14] Sutskever, I et al.'s: "Sequence to Sequence Learning with Neural Networks", ANIPS. (2014)

[Schuster 18] Schuster, M et al.'s: "Bidirectional Recurrent Neural Networks", IEEE TSP,(1997)

[Alireza 18] Alireza, A et al.'s: "GENERATING SUMMARIES FOR METHODS OF EVENT-DRIVEN PROGRAMS: AN ANDROID CASE STUDY", arXiv preprint,(2018)

[Pinji 18] Pinji, H et al.'s: "Characterizing the Natural Language Descriptions in Software Logging Statements", ACM/IEEE ICASE(2018)

[Schuster 18] Xing, H et al.'s: "Deep Code Comment Generation", CPC ACM,(2018)