JSiSE Research Report
vol.33,no.7(2019-3)

問題文に合わせたプログラムコメントの自動生成

高橋明義[*1], 椎名広光[*2], 小林伸行[*3]
*1 岡山理科大学大学院, *2 岡山理科大学, *3 山陽学園大学

Automatic Generation of Comments for Programs using Question Description

Akiyoshi Takahashi [*1], Hiromitsu Shiina[*2], Nobuyuki Kobayashi[*3]
*1 Graduate School of Informatics, Okayama University of Science,
*2 Faculty of Informatics, Okayama University of Science
*3 Faculty of Regional Management, Sanyo Gakuen University

To aid understanding of programs and understanding of procedures, it aims at automatically generating comments from source code. As a method, we learn the program list and comment pair by Encoder-Decoder model using LSTM, thereby generating comments of the program that was the target of learning. Also, since it is difficult to increase the amount of learning data, we use program problem sentences to generate comments for programs more suitable for users.

キーワード: コメント生成，アルゴリズム理解，ニューラルネット翻訳

1. はじめに

新学習要領が導入され小学校のプログラミング教育が必修化され，2020 年から実施される．その背景から小学校，中学校でプログラミング教育について話題になることが多い．また，新学習要領ではプログラミングそのものではなくプログラミング的思考が重要視されている．プログラミング教育については，いろいろな試みがなされており，Web を利用した支援システム(1)やプログラムの処理をカード型での学習(2)などが提案されている．プログラミング教育のカリキュラムに関しては，ルーブリック(3)が提案されている．また，プログラミング的思考については，プログラムそのものではなく手続きの記述と組み立てが重要になってくると考えられ，新開ら(4)は手作業によるアルゴリズムの学習についての研究を行っている．

本研究では，プログラムの理解や手順の理解を助けるために，ソースコードから手続きを自動生成することでプログラムの理解の補助となることを目的としている．

一方，ニューラルネットワークを利用した手法の発展により，自然言語の翻訳や文生成が行われている．特に LSTM(5)を用いた Encoder-Decoder 翻訳モデル(6)が提案されてきており，翻訳精度が向上している．本研究では，その手法を利用してソースコードとコメントのペアをニューラルネットワークの LSTM を用いた Encoder-Decoder モデルで学習することで，新しいソースコードのコメントや手続きの生成を行っている．

しかしながら，単純に LSTM をソースコードとコメントの学習を適用するだけでは，生成コメントが適切でないことがある．その原因としては，機械学習では学習データの質と量がそのあとの精度に影響を及ぼす．適用範囲の限られた範囲での応用においては，学習データの量を確保するのが難しいが，質については工夫の余地があると考えられる．言い換えると，手軽なデータ量でのニューラルネットワークの利用を想定すると逆に学習データ量をあまり多く取得できなくても補強または関連するデータによって目的達成を目指している．

本研究は，生成したいコメントも学習するソースコードとコメントのうち，特にコメントの記述が重要と考えて，学習データ用のコメント作成には揺らぎを少なくするために大学の情報系学科 1 年生での講義に使用された C 言語の例や課題のソースコード 53 個を利用し，コメント部分は一人の教員のチェックを行っている．加えて，単純にニューラルネットワークの手法では，別な場所にある類似した部分に一致したコメントを部分的にコピーしてくると考えられるが，課題の問題文に少しでもすり合わせをしたコメント生成を行うことを目的としている．これによって，対象に合わせたコメントを生成の精度が上がると考えられる．また，本研究が対象とした，大学の講義ではプログラムの説明を行うが，対象を考慮して説明を行っていると考えている．よって，講義で使用されている説明を学習に利用すれば，それから生成されるコメントは，学習対象者に適合しやすいと想定している．加えて，そこで扱われる課題の問題文や講義資料の記述を生かしたコメント生成も重要であると考えられる．

より学習対象を考慮するという面では，単純に LSMT によって生成されたコードでは不十分で，特に初学者レベルにおいては，変数の変化について知る必要がある．しかしながら，もともと学習用に使われているソースコードだけでは，変数に関する情報が不足気味なるため，ソースコードの変数に関する流れが分かりにくい傾向にある．

これらのことから本研究では，

(1)外部情報の利用として，課題の問題文の重要語を生成結果に影響を与える処理

(2)変数情報をコメント生成に反映させる処理

を追加することで，課題の問題文に近くてより良いコメントの生成を試みている．

2. LSTM によるソースコードからコメント生成

2.1 ソースコードとコメントの学習とコメント生成の概要

コメント生成に対する Encoder-Decoder モデルの学習とコメント生成の処理は，次の 4 つの処理からなる．

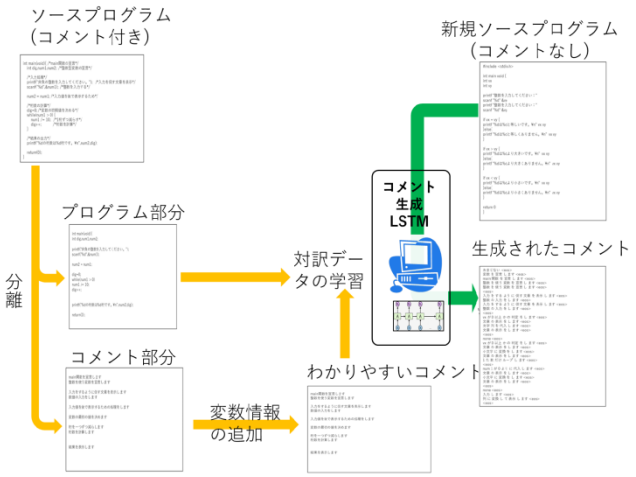

図 1 コメント生成の流

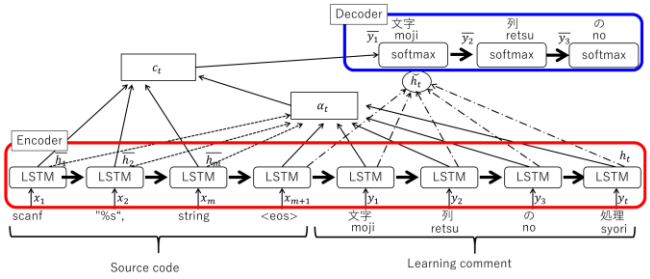

図 2 Encoder-Decoder モデルによるソースとコメントの学習

(1)学習：コメントがついたソースコードと教師データを分離し，ソースコードの変数名の整理を行った後，ソースコードの 1 行とそのコメントのペアを Encoder-Decoder モデルの学習を行う．

(2)コメントの生成確率：コメントの付いていないソースコードに対して Encoder-Decoder モデルを利用してコメント生成での単語連接確率を生成する．

(3)コメントが付されていないソースコードの課題の問題文の重要語に対して，コメント生成の単語連接確率を高くなるように変更し，学習データ以外の情報の影響を大きくする．

(4)コメントの修正：生成したコメントについては，変数情報を補完して最終的な生成コメントとする．

2.2 Encoder-Decoder モデルの学習と生成

本研究で用いた LSTM の学習については，次の 3 つ変換処理を組み合わせている．

(1) ソースコードとコメントをそれぞれ単語単位に分割し，単語のつながりの関係性を学習する．単語に

分割したソースコードを 1 単語ずつ LSTM に入力し，文脈情報を蓄積する．その後コメントを 1 単語ずつ LSTM に入力し，文脈情報を蓄積しながら単語を生成して学習を行う．入力単語x_iに対しては，中間層の出力h_iを保持しておき，出力単語y_tとx_iの類似度を正規化しα_tとする．このα_tとh_iを使い文脈ベクトルc_tを作り，線形作用素Wで重みをつけて活性化関数tanhにより中間層の出力 $\tilde{h_t}$ を作る．$\tilde{h_t}$を入力として softmax 関数による変換y_tを作る．

(2) (1)で行った学習をもとに新たなソースコードに対するコメントを生成する．ソースコードを単語ごとに分割して LSTM に入力していき，最後まで入力を行った後に文脈情報をもとにコメントの 1 つ目の単語を出力し，出力された単語と文脈情報をもとに次の単語を生成する．

2.3 外部情報を利用したコメント生成

前節では，プログラムのソースコードとそのコメントの行単位のペアを学習し，新しいソースコードに対してコメントを生成している．この場合，利用できる情報が，類似するソースコードに対応するコメントの生成を行うと考えられる．しかし，それだけでは課題に対応するソースコードのコメントを生成することが難しい．単純な Encoder-Decoder モデルでは話題が一致しなくても類似したソースコードのコメントをコピーするので，生成したいコメントに関係する話題に少しでも関連するようには，情報が不足している．

本研究では，課題の問題文に関連する情報をコメント生成に生かすように，外部情報の利用として課題の問題文の重要語を生成結果に影響を与えるようにした(図 3)．特に，課題の問題文の中で影響の大きい重要語の抽出には，TF-IDF を利用している．重要語によるコメント生成の手続きを次に述べる．

(1) はじめに元となる問題全体の文書を読み込んでおく．その後入力するプログラムの問題文を入力し，TF--IDF により文書全体に対する問題文中の各単語の重要度を求める．

(2) ソースコードを LSTM に入力し，単語の出力時に問題文中の重要語に一致する単語がある場合，生成確率に対して TF-IDF のα倍を掛け，その中の数値の一番高いものを LSTM の出力とする．学習に利用している課題 53 個の問題文を全体として，問題文「配列 5 個の数値(整数)を，scanf を用いて格納し最大値と最小値を求める．ただし，最小値と最大値の格納されている配列の添字も出力せよ．」の単語 TF-IDF を表 1 に示す．また，次に示す問題文に対するソースコードからコメントを生成する時，TF-IDF 値の高い「配列」をコメント生成に利用している様子を図 4 に示す．

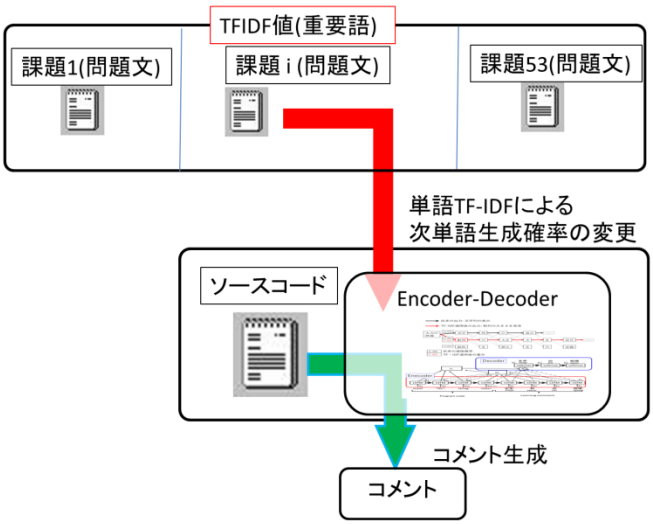

図 3　問題文とコメント生成の関係

表 1　問題文の中の単語の TF-IDF

単語	TF-IDF	単語	TF-IDF
最小	0.45541	ただし	0.22770
最大	0.40027	いる	0.22770
格納	0.38594	用い	0.16841
配列	0.32545	求める	0.16272
数値	0.24959	整数	0.13100
scanf	0.24959	せよ	0.09828
添字	0.24959	出力	0.09667

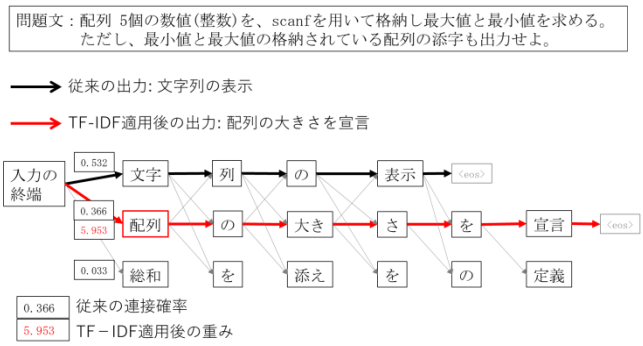

図 4　TF-IDF の適用によるコメント生成の違い

行	プログラム	Encoder–DeCoder のみの単純生成コメント	問題文を利用した生成コメント	変数情報を付加した生成コメント	評価 (6 人の平均)		
					単純生成	問題文利用	変数
1	#include < stdio.h >	入出力を行うための宣言	入出力を行うための宣言	入出力を行うための宣言	5.83	5.83	5.83
2	#define NUMBER 5	文字列を表示する	配列の大きさを宣言	配列の大きさを定義づける	1.50	4.83	5.17
3	int main(void){	main 関数の宣言	main 関数の宣言	main 関数の宣言	5.83	5.83	5.83
4	int i, j;	整数型変数の宣言	整数型変数の宣言	整数型変数 (i)(j) の宣言	4.00	4.00	6.00
5	int min, max ;	整数型変数の宣言	整数型変数の宣言	整数型変数 (min)(max) の宣言	4.00	4.00	6.00
6	int min_loc, max_loc ;	整数型変数の宣言	整数型変数の宣言	整数型変数 (min) の宣言	4.17	4.17	2.67
7	int vx[NUMBER];	整数型の配列の宣言	整数型の配列の宣言	整数型変数 (vx) の宣言	4.50	4.50	3.17
8	for(i=0;i < NUMBER; i++){	定義づけた回数分ループ	配列内の添え字すべてに数値を入力させる	配列内の添え字 (i) すべてに数値を入力	4.67	3.00	3.33
9	printf(”vx[%d]=”,i);	配列のすべての添え字内に数値を入力	配列のすべての添え字内に数値を入力	配列の表示	2.67	2.67	4.33
10	scanf(%d”,&vx[i]);	整数を入力	整数を入力	整数を入力 (i)(vx)	4.17	4.17	4.00
11	}	そうでない場合	そうでない場合	None	1.33	1.33	3.5
12	min=vx[0];	min の初期値を vx[0] とする	最小値の値を記憶	配列の文を表示	5.0	4.17	1.67
13	min_loc=0;	min のあり場所を 0	min のあり場所を 0	最小値の定義づけ	3.67	3.67	2.33
14	max=vx[0];	max の初期値を vx[0] とする	max の初期値を vx[0] とする	x の計算	5.00	5.00	1.67
15	max_loc=0;	max のあり場所を 0	max のあり場所を 0	最大値 (max) の文章を表示	3.83	3.83	1.17
16	for(i=1;i< NUMBER; i++){	定義づけた回数分ループ	定義づけた回数分ループ	配列内の添え字 (i) すべてに数値を入力	4.50	4.50	2.50
17	if(min > vx[i]){	min の方が大きい場合	min の方が大きい場合	min の方が大きい場合	4.17	4.17	1.33
18	min=vx[i];	最小値の値を記憶	最小値の値を記憶	計算を行う (i)(min)	4.83	4.83	1.50
19	min_loc=i;	最小値の場所を記憶	最小値の場所を記憶	最小値の場所 (i) を記憶	4.17	4.17	4.50
20	}	None	None	None	4.00	4.00	3.50
21	if(max < vx[i]){	max の方が小さい場合	max の方が小さい場合	max の方が小さい場合	4.33	4.33	1.33
22	max=vx[i];	max はいつも一番大きく	最大値の値を記憶	計算を行う (i)(max)	1.50	4.50	1.83
23	max_loc = i;	max のあり場所を 0	最大値の場所を記憶	最小値の場所 (i) を記憶	2.00	4.50	2.83
24	}	行内の繰り返し	None	None	1.17	1.17	4.00
25	}	そうでない場合	数値を入力する	None	1.17	1.17	4.00
26	printf(“最小値%d, 添え字%d \n ”, min, min_loc);	最小値と添え字を表示	最小値と添え字を表示	最小値 (min) と添え字を表示	4.00	4.00	4.67
27	printf(“最大値%d, 添え字%d \n ”, max, max_loc);	最大値と添え字を表示	最大値と添え字を表示	最大値 (max) と添え字を表示	4.00	4.00	4.67
28	return(0);	関数を終了させる	関数を終了させる	関数を終了させる	5.50	5.50	3.00
29	}	そうでない場合	数値を入力する	None	1.17	1.17	4.00

図 5 1 行ごとの生成コメントとアンケート評価

2.4 生成コメントへの変数情報の補完

ソースコードからのコメント生成については，学習段階を考慮すべきで，文法を学習している段階では，変数の変化について知る必要がある．しかし，通常のコメントでは，変数の変化まで含めてコメントを記述していない．そのため，変数に関する情報が不足気味なるため，プログラムの変数に関する流れが分かりにくい傾向にある．変数名の補完の処理について，コメント学習時の変数処理と，コメント生成時の処理に分けて述べる．

2.4.1 コメント学習時の変数処理

(1)ソースコードの変数名の変更処理として，ソースコードの変数を同じものに統一する．例えば，変数 min を x，変数 vx を 仮の変数 x に置き換える．

(2)ソースコードに対応するコメントに変数情報として， (x)を付加する．

(3)Encoder-Decoder 翻訳モデルに変数補完処理を行ったデータを学習させる．

2.4.2 コメント生成時の処理

(1)コメントを生成するソースコードに対して，変数名補完処理として，学習データと同じように変数の統一を行う．

(2)コメント生成時に変数情報を適用するために，テストデータから各行の変数情報を取得しておく．

(3)学習済みの Encoder-Decoder 翻訳モデルによって

コメントを生成する.
(4)コメント生成後，コメント内の仮変数(x)をもとのソースコードの変数に戻す.

3. 生成コメントの評価

3.1 1行ごとのコメント生成

LSTM だけで生成したコメント，問題文の単語の TF--IDF を考慮して生成したコメント，変数情報の補完処理をしたコメントの3つを図 5 に示す．また，研究室の学生 6 人による 6 段階評価(（悪い）1～6(良い))の平均ランクについても示す.

- 3 種類のコメントの平均ランクの平均については，Encoder-Decoder のみが 3.68，問題文を加味が 3.99，変数情報の付加が 3.52 となっており，問題文を加味した方が総合すると良いと評価されている.
- 12 行目については，Encoder-Decoder モデルが良い評価を受けているが，著作者たちは問題文を加味した方が良いと考えており，アンケートとの違いが出ている．そのほかの行に土江は，Encoder-Decoder のみより問題文加味が各行でも同等か良い評価となっている.
- 変数情報については，4，5 行目など正しい変数情報が適用されたが，6，7 行目などは変数情報の取り出しがうまくいかず付与された変数情報が間違っている．また，変数を利用している 12～18 行目が良い評価を受けていない．生成のされ方が良くないと考えられる．一方，変数を定義しているところについては，用意評価受けている．変数の情報については，改悪されたものもあり適用個所の工夫の必要があると考えられる.

4. まとめと今後の課題

本研究では，ソースコードとコメントのペアを対訳データとして学習することで，アルゴリズムの手順を理解するために，手順生成を行った．特に，外部情報としての課題の問題文の利用によって，コメントの生成が関係のない内容を取り出してくることが少なくなったと考えられる.

本研究では，外部情報としては課題の問題文を利用しているが，講義資料などの関連する情報を利用することで，より適切なコメントを生成することができると考えられる.

謝辞

本研究は岡山理科大学プロジェクト研究推進事業(OUS-RP-29-2)の助成を受けたものです.

参 考 文 献

(1) 宇野健，畝川みなみ: “C 言語学習支援のための Web 上でのプログラミング環境の開発”, Journal of the Faculty of Management and Information System Prefectural of University of Hiroshima, No. 5, pp. 77-84 (2013).
(2) 松本慎平，林雄介，平島宗: “部分間の関係を考えることに焦点を当てたカード操作によるプログラミング学習システムの開発”，電気学会論文誌 C, Vol.138, No.8, pp.999-1010 (2018)
(3) 松永賢次，萩谷昌己，共通教科情報科ルーブリックにおける思考・判断・表現の位置づけ，第 10 回全国高等学校情報研究会発表会全国大会（東京大会）(2017)
(4) 新開他: “手作業の学習教材を活用したアルゴリズム教育の試み”,日本教育工学会第31回全国大会, pp.407-408 (2015).
(5) Greff, K. et al.: “LSTM: A Search Space Odyssey”, IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, Issue10, pp. 2222-2232 (2017).
(6) Wu, Y. et al.: "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation", arXiv, pp. 1-23(2016) https://arxiv.org/abs/1609.08144 (2019年2月6日確認)