LSTMによる外部情報を利用したプログラムコメント自動生成

高橋 明義[1]　椎名 広光[2]　小林 伸行[3]
岡山理科大学大学院 総合情報研究科 情報科学専攻[1]
岡山理科大学 総合情報学部 情報科学科[2]
山陽学園大学 地域マネジメント学部 地域マネジメント学科[3]

i18im03ta@ous.jp[1], shiina@mis.ous.ac.jp[2], kob_nob@sguc.ac.jp[3],

1　はじめに

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の重要性が言われており，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が重要になってくると考えられる．

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 を用いた Encoder-Decoder 翻訳モデル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，大学の情報系学科 1 年生での講義に使われた C 言語の例や課題のプログラム 53 個を利用して，プログラムリストの 1 行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，新しいプログラムのコメントや手続き (コメント) 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，利用者の立場からは生成されたコメントの評価が良くない場合がある．その原因として，もともと学習用に使われているソースコードだけでは，コメントを生成するには，学習量とともに関連する情報が不足していることが問題と考えられる．加えて，プログラムリスト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，大学生レベルにおいては，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，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．そこで本研究では，(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ような処理，(2) 変数情報をコメント生成に反映させる処理のを新たに追加することで，課題の問題文に近くてより良いソースコードのコメントを生成することを目標としている．

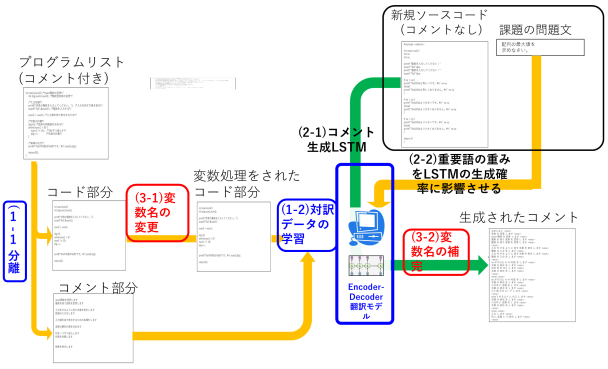

図 1: Encoder-Decoder 翻訳モデルによる学習とコメント生成の概要

2　LSTMによるソースコードからコメント生成

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 3 つの処理からなる (図 1)．

(1) 学習：コメントがついたプログラムリストを教師データを分離し，プログラムリストの変数名の整理を行った後プログラムリストの 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プログラムに対して Encoder-Decoder モデルを利用して

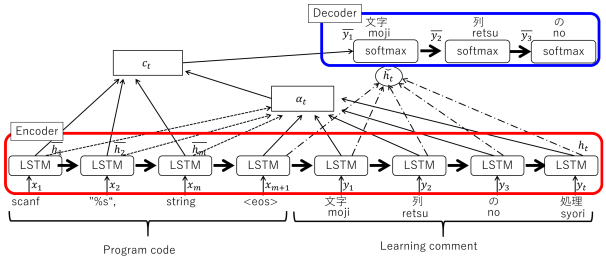

図 2: LSTM の構成

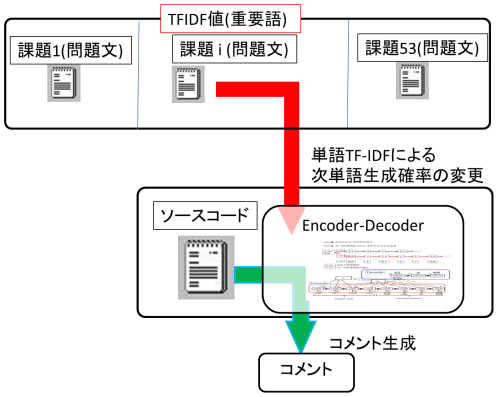

図 3: 問題文を加味したコメント生成処理の概要

コメント生成での単語連接確率を生成し，コメントの付いていないプログラムの課題の問題文の重要語を，コメント生成での単語連接確率に影響を与えるようにする．

(3) 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．

2.1 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，つぎの 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，1 行ごとに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プログラムを 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 $\check{h}_t$ を作る．$\check{h}_t$ を入力として softmax 関数による変換 y_t を作る．

表 1: 課題の問題文における単語の TF–IDF

単語	TF–IDF	単語	TF–IDF
最小	0.45541	ただし	0.2277
最大	0.40027	いる	0.2277
格納	0.38594	用い	0.16841
配列	0.32545	求める	0.16272
数値	0.24959	整数	0.131
scanf	0.24959	せよ	0.09828
添字	0.24959	出力	0.09667

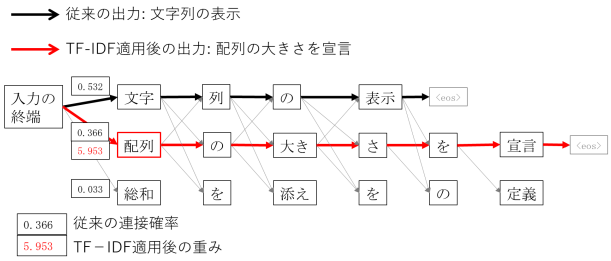

図 4: TF–IDF による問題文の影響によるコメント生成

(2) (1) で行った学習をもとに新たなプログラムに対するコメントを生成する．プログラム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．

2.2 外部情報を利用したコメント生成

これまで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には利用する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)．課題の問題文の中から重要となる単語を抽出するのには，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．

表 2: 生成コメントに対する TF–IDF と変数情報の適用

行	プログラム	TF–IDF 適用前の生成コメント	TF–IDF 適用後の生成コメント	TF–IDF 適用＋変数情報を適用	評価		
					適用前	TF–IDF 後	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	4	4	4
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	4	4
3	int main(void){	main 関数の宣言	main 関数の宣言	main 関数の宣言	4	4	4
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	4	5
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	4	5
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	4	2
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	4	3
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	3	3
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	4	4	2
10	scanf(%d”,&vx[i]);	整数を入力	整数を入力	整数を入力 (i)(vx)	4	4	3
11	}	そうでない場合	そうでない場合	None	1	1	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	3	4	1
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	3	1
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	3	3	1
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	3	1
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	3	3	2
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	3	3	3
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	3	1
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	3	3	4
20	}	None	None	None	5	5	5
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	3	3	3
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	4	2
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	1	4	2
24	}	行内の繰り返し	None	None	1	5	5
25	}	そうでない場合	数値を入力する	None	1	1	5
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	4	5
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	4	5
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	4	4	4
29	}	そうでない場合	数値を入力する	None	1	1	5

(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 に入力していき，単語の出力時に出力確率上位 5 単語を取り出す．その 5 単語のうち問題文中の重要語にマッチする単語があった場合，生成確率に対して TF–IDF の $\alpha(= 10)$ 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，次の問題文の単語 TF–IDF を表 1 に示す．

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．

また，次に示す問題文に対するプログラムリスト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．

コメントに変換するプログラムリスト

#define NUMBER 5

2.3 生成コメントへの変数情報の補完

プログラムリスト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，初心者段階では変数を意識

しなくてもよいと考えられる．しかし，次の段階を想定すると変数の変化について知る必要がある．また，もともと学習用に使われているプログラムリストだけでは，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．次に変数名の処理と補完の処理について述べる．

(1) プログラムリストの変数名の変更処理

まず学習用ソースプログラムの変数を同じものに統一する (例:min → x , vx → x)．次に学習用コメント内の変数情報を付与する部分を仮に (x) と置く．そして変数情報の処理をした学習データを繰り返し学習する．

(2)Encoder-Decoder 翻訳モデルによる変数名補完処理

学習データと同じようにテストデータも変数の統一を行う．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．(1) の学習データをもとにテストデータに対するコメントを生成する．このコメントの変数情報は統一されたものが出力されるので，各行の変数情報を適用する．

3　生成コメントの評価

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 つを表 2 に示す。また，生成されたコメントを 1〜5 で評価した結果についても表 2 に示す。

- 表 2 の 2 行目は TF–IDF の適用により「文字列」→「配列」となり適用前より正しいコメントが生成された．12，13 行目などは中途半端に単語が変換されたことにより統一性のないコメントになっている．全体を見ると TF–IDF 適用前に比べて 4 つのコメントの評価が高くなっており，評価の平均が 0.45 高くなっている．問題文の情報に関連させることで，生成したコメントの評価が改善されたと考えられる．

- 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，変数を統一した影響で 12〜15 行目のような元の変数情報の影響を強く受けたコメントなどが悪くなっている．変数情報の補完は，TF–IDF 適用のコメントと比べて評価が高くなったコメントが 8 つだが，評価の低くなったコメントが 12 と多く，評価平均が 0.7 低くなっている．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，変数情報の追加，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．

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，林，平島, “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，萩谷, “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] Greff, K. et al., “LSTM: A Search Space Odyssey,” IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, Issue10, pp. 2222-2232, 2017.