自然言語翻訳を利用したソースコード自動生成に向けて（SES 2019 ポスター発表）

井 刈 是 水
Yoshimi IKARI
情報メディア学専攻修士課程　1 年

1. 概要

本研究は，機械学習分野の翻訳技術を転用し，仕様（機能説明）からのソースコード生成を検討するものである．SES 2019（ソフトウェアエンジニアリングシンポジウム 2019）でポスター発表した．

2. 背景

ソフトウェアの大規模化・複雑化により開発コストは上昇している．開発効率化のため，IDE（統合開発環境）にコード推薦機能などが追加されてきた．

現在のコード推薦は変数名や関数名といったトークン単位であるため，関数やクラス単位での生成ができればより有用である．

2.1 自然言語翻訳

近年，機械学習による自然言語翻訳がよく研究されている．翻訳前後の文章における意味の対応関係のように，仕様とソースコードの間にも意味の対応関係があるため，ソースコードを自然言語のように文章とみなして機械翻訳技術を応用することは適切であると考えられる．

3. 提案手法

自然言語翻訳に用いられる深層学習モデルに，説明コメント（関数・メソッドに対する説明であるコメント，関数コメント）が入力されるとソースコードが出力されるように学習させる．

対象言語は Java とする．オープンソースなどの数が多いため学習データを用意しやすい点と，型を明示的に宣言するため意味を学習させやすいと推測したためである．

3.1 Transformer

ニューラルネットワーク（以下 NN）のモデルの 1 つである Transformer は，非自明な関係を Attention という辞書オブジェクトの Key-Value による変換で表現できる．これにより構文や意味構造を学習できる．

Transformer は Encoder と Decoder によって構成され，どちらも 1 つ以上の Attention と 1 つの順伝播型 NN のセットを幾つか重ねて構成したものである．RNN のような再帰的構造は無いため，Transformer 以前に用いられた RNN より学習が早い．

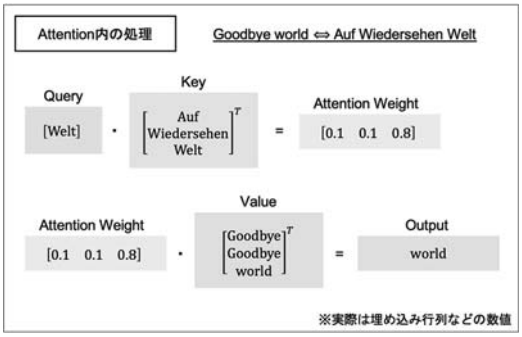

図 1　Attention における処理

3.2 前処理

入力および出力は 1 行である必要があるため，複数行で記述されている説明コメントまたはソースコードは，改行文字をスペースに置き換えて 1 行にする．

また，意味を学習しやすいように以下のように，ソースコードを更に変換する．

［a］フィールドやメソッドの呼び出し（クラス名やメソッド名，クラス名が ”.” で連結されているもの）以外のトークンは，意味が異なる単語としてスペースで区切る

［b］個々のクラスの役割を識別して学習させるため

に，外部クラス名はフルパスのクラス名に置換する［c］プログラマによってコード記述に差の出る変数名，数値，文字，文字列は，VARIABLE，NUMBER，CHARACTER，STRING という固定文字列に置換する

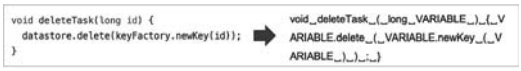

図 2　ソースコードの変換

3.3　学習

Transformer に説明コメントとソースコードのペアを学習させる．

4. 実験結果

実装には，Google 社が提供するオープンソフトウェアライブラリである TensorFlow を用いた．

学習データとして，GitHub から取得した 2 リポジトリ内で，10 文字以上のコメントが付いているメソッドを選び，前処理を行なった．データの総数は説明コメントとメソッドの 92 ペア（コメント 1095 単語，コード 3861 単語，非辞書数）であり，これらを Transformer モデルに学習させた．

評価のために，学習済みモデルに学習に用いたコメントデータを入力し，出力を検証した．

学習データ数	正解数		不正解数	
	完全一致した	正しく機能した	部分的過不足	大きな誤り
92	72	1	6	13
合計	73 (79.35%)		19 (20.65%)	

図 3　学習結果の再現率

5. 考察

検証における再現率は 73% であり，自然言語翻訳の精度が 40% ほどであることを考慮すると，高い値となった．

「正解」のうちの「正しく機能した」結果は，アクセス修飾子 public が出力であるメソッドテキストの先頭に付加されており，学習データ内の対応するメソッドには記述されていなかったが学習データ側の機能として出力結果も動作するため，「正解」と分類した．

「不正解」のうち「部分的過不足」は，必要な処理の半分程度しか記述されていない，または，宣言などの式の記述が途中で終わってしまっていてコンパイルエラーを起こす，といったものであった．「大きな誤り」は，類似コメント A と B に対し，コメント A を入力するとコメント B に対応するメソッドが出力された．

実験結果には記述してないが，検証として，学習データ中でコメントに使用された単語を用い，意味のあるコメントを作成して入力した．学習データ中に完全一致するコメントが無いように作成したが，出力は類似したコメントのペアであるメソッドであった．これは，学習データが乏しいため，2・3 単語の差異に対応せず，類似コメントのペアを出力するように学習したと考えられる．

現在，コメントはソースコード中から人手により取得しているが，今後は Java に自動的に説明コメントをつける研究を利用して正確で法則性のあるコメント生成し，データ数不足の解決や正解率の向上を図る．