DEIM Forum 2023 1b-4-3

文脈を考慮した自然言語からコードへの機械翻訳の実現に向けて

佐藤　美唯† 　木村江梨花† 　小原百々雅†† 　梶浦　照乃†† 　倉光　君郎†

† 日本女子大学理学部数物情報科学科　〒 112–8681 東京都文京区目白台 2-8-1
†† 日本女子大学大学院理学研究科数理・物性構造科学専攻　〒 112–8681 東京都文京区目白台 2-8-1
E-mail: †{m1916038sm,m1916021ke,m1816019om,m1816023kt}@ug.jwu.ac.jp, ††kuramitsuk@fc.jwu.ac.jp

あらまし　近年，大規模言語モデルを用いた機械翻訳の実用性は高まっている．しかし，ユーザは多様な自然言語を入力するため，必ずしもユーザが意図した翻訳結果が得られるとは限らない．文脈に依存したユーザの入力に対して，ユーザの意図と異なる翻訳結果が出力されるという課題がある．この課題に対するアプローチとして，文脈情報の活用が検討されているが，我々が取り組む自然言語からコードへの機械翻訳ではより一層の難しさがある．本研究では，コーディングの文脈に依存した入力から，ユーザの意図したコードを提示することを目指し，Google Colab の内部情報を参照した文脈処理を提案する．本論文では，提案手法を紹介し，実際のユーザの入力に対する翻訳結果を報告する．

キーワード　自然言語，機械翻訳，大規模言語モデル

1　はじめに

近年，深層学習技術の発展は目覚ましく，機械翻訳の実用性は高まっている．ニューラル機械翻訳 [1] の導入や深層学習モデル Transformer [2] の登場により，機械翻訳精度は向上した．また，Open AI/GPT-3 [3] のような，大規模なテキストデータを事前に学習した大規模言語モデルが活用されるなど，機械翻訳精度は向上し続けている．

しかし，依然として課題も残っている．文脈に依存した入力に対して，ユーザの意図と異なる翻訳結果が出力されることが，課題の一つに挙げられる．ユーザの入力を正しく解釈し，ユーザが意図した翻訳を行うために，代名詞，語彙の一貫性と結束性，指示語や省略を正しく扱うことが求められるため，文脈が重要である [4].

我々は，日本語からコードへの機械翻訳の実現を目指している [5]．文脈に依存しない入力に対しては，ユーザの意図した翻訳結果の出力が可能になった．しかし，コーディングの文脈に依存した入力に対しては，ユーザの意図と異なる翻訳結果が出力されてしまう．コーディングの文脈に依存した入力とは，ユーザのコーディング状況に影響されるため、翻訳対象文のみでは，ユーザの入力を正しく解釈できない入力のことである．本研究では，Google Colab の内部情報を参照した文脈処理を提案し、コーディングの文脈に依存した日本語の入力から，ユーザが意図したコードを出力することを目指す．

本論文の残りの構成は以下の通りである．2 節では，日本語からコードへの機械翻訳と課題について述べた後に，我々が開発するプログラミング学習支援 AI KOGI について述べる．3 節では，我々の提案手法である Google Colab の内部情報を参照した文脈処理を紹介する．4 節では，処理の実装とコード翻訳モデルの構築を行った上で，実際のユーザの入力に対する翻訳結果を述べる．5 節では関連研究を概観し，6 節で本論文をまとめる．

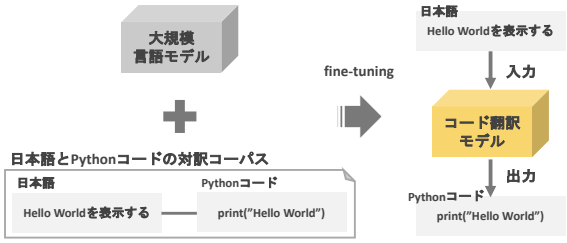

図 1　コード翻訳モデルの構築

2　コード翻訳

本節では，日本語からコードへの機械翻訳と課題について述べた後に、我々が開発するプログラミング学習支援 AI KOGI について述べる．

2.1　日本語からコードへの機械翻訳

我々は，日本語から Python コードへの機械翻訳 (コード翻訳) の実現へ向けて取り組んでいる [5]．構築したコード翻訳モデルが，日本語の入力から Python コードを予測することにより，コード翻訳を行う．

図 1 に，コード翻訳モデルの構築方法を示す．基本的な原理は，ニューラル機械翻訳 [1] と等しく，教師データであるコーパスの出力側を自然言語ではなく，コードにしている点が異なっている．大規模言語モデルに，我々が作成した日本語と Python コードの対訳コーパス約 2 万件を fine-tuning させ，コード翻訳モデルを構築する．テストデータの入力に対して，約 90%の比率で正しいコードの予測が可能である．

2.2　コード翻訳における課題

コーディングの文脈に依存した入力に対しては，ユーザの意図と異なる翻訳結果が予測される課題がある．コーディングの文脈に依存した入力とは，ユーザのコーディング状態に影響され、解釈が変化する入力のことである．

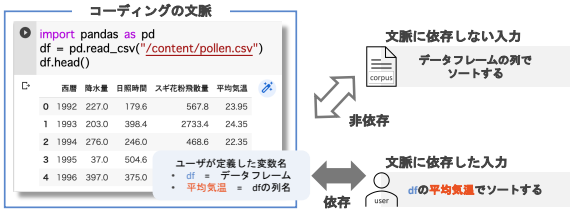

図 2　コーディングの文脈

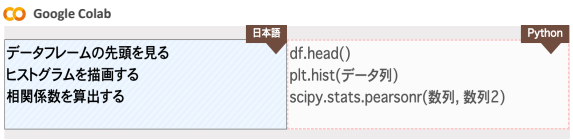

図 3　KOGI のコード翻訳機能

図 2 に，コーディングの文脈と文脈に依存した入力と依存しない入力の例を示す。まず、ユーザは,「csv ファイルをデータフレームとして読み込み，df に代入する」コーディングをする。ユーザが新たに「df」と「平均気温」を定義したコーディングの状態になる．我々は、具体的な変数名が含まれない、コーディングの文脈に依存しない入力を想定してコーパスの作成を行なっていた。しかし、ユーザがコーディングの文脈に依存した入力をした場合は、ユーザの意図したコードの予測が妨げられてしまう.「平均気温」のような、具体的な変数名はコーパスに含まれていないため、誤ったコードを出力する場合がある。

また、コーディングでは、ユーザが自由に変数名を定義出来るため、事前に変数名とコーディング上の意味を想定したコーパスを作成することは難しい。理由の一つに、ユーザが定義した変数名が持つ単語本来の意味とコーディング上の意味が異なることが挙げられる。「平均気温」を例とすると，単語本来の意味は「気温の平均値」であるが、コーディング上の意味は,「dfの列名」　である．このように，単語本来の意味と異なるため，コーディングの文脈に依存した入力を直接モデルに予測させると，誤ったコードが出力される場合がある．

2.3　KOGI

我々は，コーディング環境である Google Colab 上で動作するプログラミング学習支援 AI KOGI の開発を進めている．KOGI は，プログラミング学習者の質問や要求に応えることにより，プログラミング学習者のつまずきを軽減することを目指す [6].

図 3 は，KOGI のコード翻訳機能である．ユーザは，プログラミング学習を進める中で，コードを思い出せない場合に，左側にコード処理内容を入力する．2.1 節で示したコード翻訳モデルを通じて，ユーザの入力に対するコードが予測され，翻訳結果として右側に表示される仕組みになっている．

KOGI のコード翻訳機能は，実際のプログラミング学習者へ提供する実践経験を行なった．その際に，収集したログを解析したところ，ユーザはコーディングの文脈に依存した入力をする傾向があることが明らかになった．しかし，　コーディングの文脈に依存した入力に対しては，2.2 節に示した課題がある．

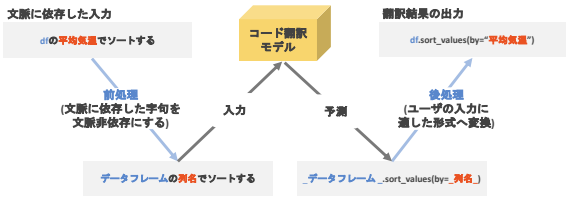

図 4　提　　案

表 1　日本語と Python コードの対訳コーパス

日本語	Python コード
列名のヒストグラムを描きたい	plt.hist(_データ列_)
列名をヒストグラムで図示したい	
列名を柱状図で可視化したい	

3　提　　案

本節では，2.2 節に示した課題に対応するための提案手法について述べる．

3.1　提案手法の概要

我々は、Google Colab の内部情報を参照した文脈処理を提案する．図 4 に，ユーザの入力から，コード翻訳モデルを通じて，翻訳結果が出力されるまでの一連の流れを示す.

（1）　文脈に依存した字句 (変数名) から文脈に依存しない字句 (データ型) に置換する (前処理)

（2）　コード翻訳モデル通じて，前処理後の入力から、コードを予測する

（3）　予測コードに含まれるデータ型を対応する変数名に置換する (後処理)

（4）　後処理後のコードを翻訳結果として出力する

本研究の提案により、期待されることは 2 点ある。1 点目は、コード翻訳モデルが正しいコードを予測しやすい点である。前処理にて、、文脈の依存関係を取り除くことにより、コーパスと同等の入力になるからである。2 点目は、ユーザの意図したコードに近いコードの出力される点である。後処理にて，変数名が穴埋めされたコードが出力されるため、文脈を考慮した翻訳結果の出力される。3.2 節に，文脈処理の実装について，3.3 節に，処理を想定した形式のコーパス作成について述べる．

3.2　文 脈 処 理

まず，前処理について述べる．前処理では、ユーザの入力に含まれる変数名を抽出し、データ型へと置換する。変数名の抽出では、Google Colab の内部情報に含まれる user namespace の変数名とデータ型の記録を使用する．

次に後処理について述べる．後処理では、前処理の内容に基づいて、予測コードのアンダーバー (_) を目印にデータ型を抽出し、変数名へと置換する。文脈に依存した入力に対しては，後処理で書き換えを行うことにより，文脈を考慮した変数名を含むコードを出力する．一方で，文脈に依存しない入力に対しては，後処理で書き換えを行わないコードを出力するため，ユーザが自由に編集することが可能である．

3.3 処理を想定した形式のコーパス

表 1 に，処理を想定した形式の日本語と Python コードの対訳コーパス例を示す．本研究で提案するコーパスには，2 種類の特徴がある．

1 つ目は，1 つの Python コードに対して，複数の日本語の表現が対訳付けられている点である．この特徴は，従来に作成したコーパスの特徴を受け継いでいる [7]．十分なモデルの精度を確保するためには，コーパス量が必要になるため，類義語による置換や語順の入れ替えと行う Data Augmentation を取り入れている．本研究では，1 つのコードに対して，最大 10 種類の日本語を対訳付けたコーパスを作成する．

2 つ目は，Python コード側にデータ型の日本語を含めており，前後にアンダーバーを付与している点である．．データ型の前後のアンダーバー (_) を付与することにより，3.2 節で示した後処理を可能にしている．

4 実　　験

本節では，文脈に依存した入力に対して，提案手法の Google Colab の内部情報を参照した文脈処理を行なった場合の翻訳結果ついて述べる．

4.1 実 験 概 要

我々は，実際にユーザが入力した文脈に依存した入力に対して，従来の文脈処理なしと提案手法である文脈処理のありの翻訳結果を概観し，効果を確認することを目的に実験を行う．実験手順を下記の通りである．

（1） 処理の実装とコード翻訳モデルの構築を行う

（2） 評価用データを作成する

（3） 文脈処理なしと文脈処理のありの翻訳結果を確認する

3 節で示した Google Colab の内部情報を参照した文脈処理により，文脈に依存した入力に対して，ユーザの意図したコードの出力を期待する．

4.2 提案手法の実現

我々は，処理の実装とコード翻訳モデルの構築を行った．コード翻訳翻訳モデルの構築では，大規模言語モデルである google/mT5-small [8] を使用した．mT5(Multilingual T5) は，入出力がテキスト形式で統一されており，日本語・英語を含む 101 言語の大規模テキストデータ mC4 が事前学習されている特徴がある．また，3.3 節で示した形式の日本語と Python コードの対訳コーパスを訓練データに 11993 件，検証データに 2571 件，テストデータに 2570 件使用した．テストデータに対するコード翻訳翻訳モデルの精度は，予測コードと参照コードの完全一致率は 88.8%であった．BLEU スコア (n-gram の適合率に基づく文章の類似度を計る評価尺度) は，93.8 であった．

4.3 評価用データの作成

我々は，事前に KOGI のコード翻訳機能ををプログラミング学習者へ提供する実践経験を行った．その際のログデータに含まれるユーザの入力のうち，文脈に依存した入力をランダムに 50 件抽出し，人手でユーザの意図したコードを付与することにより，評価用データを作成した．

表 2　演 習 内 容

1	'pollen.csv' をデータフレームとして読み込み，最初の 5 行を表示する
2	読み込んだデータフレームのうち，平均気温上位 5 件を表示する
3	統計量を確認する
4	降水量の変化を各年度ごとに折れ線グラフ化する
5	杉花粉の飛散量のヒストグラムを作る
6	降水量とスギ花粉飛散量の散布図を作る
7	降水量とスギ花粉飛散量の相関係数を算出する
8	平均気温とスギ花粉飛散量の単回帰モデルを作成する
9	平均気温 24C のときの花粉飛散量を予測する
10	単回帰モデルの残差の平均と標準偏差を求める

実践経験の概要について，以下に述べる．我々は，日本女子大学の数物情報科学科専門科目である「数値計算法 I」を履修した学生 (大学生) に，KOGI を提供した．実践経験の詳細を下記の通りである．

- 授業内容：データサイエンスと機械学習
- 実施日：2022 年 7 月 11 日
- 対象者：45 名の履修者

なお，授業最終回の総合的な演習を行う際に，図 3 のコード翻訳機能を提供した．表 2 に演習内容を示す．Python によるデータ分析や機械学習の問題が含まれている．

4.4 実 験 結 果

ユーザの入力のうち，文脈に依存した入力をランダムに抽出した 50 件に対して，文脈処理なしと文脈処理のありの場合のコード翻訳を行なった．表 3 は，コード翻訳結果を一部抜粋したものである．「平均気温」,「降水量」,「スギ花粉飛散量」は，ユーザが定義した変数名であり，文脈に依存した字句である．

表 3 の 1 は，文脈処理の前処理により，正しいコードの予測が可能になった例である．文脈処理なしでは完全に誤ったコードを出力しており，文脈処理ありではユーザの意図したコードを出力した．

表 3 の 2 は，文脈処理の後処理により，よりユーザの意図したコードに近いコードの出力が可能になった例である．文脈処理なしでは正しいコードが予測できていたが，ユーザの意図と異なるコードの出力した．文脈処理ありでは「降水量」,「スギ花粉飛散量」の変数名に対応したユーザの意図したコードを出力した．

表 3 の 3 は，文脈処理なしと文脈処理ありともに，ユーザの意図と異なるコードを出力した例である．文脈処理なしと文脈処理ありともに，正しいコードを予測出来なかった．

定量評価では，参照コードと出力コードの完全一致率は，12.0%，BLEU スコアは，40.2 であった．また，コード翻訳モデルは，参照コードとは一致しなくとも，意味は等しいコードを予測していた．そのため，人手評価を行なったところ，54.0%でユーザの意図したコードの出力が可能になった．

4.5 得られた知見

GoogleColab の内部情報を参照した文脈処理を行うことにより，ユーザの意図したコードにより近いコードを出力されることを確認した．文脈処理なしでは、完全に誤ったコードを出力

表 3　ユーザの入力と KOGI の出力結果 (抜粋)

	文脈に依存した入力	文脈処理	翻訳結果の出力	評価
1	df の平均気温を降順に並べ替えたい	なし	scipy.stats.sort values(by=df, ascending=False)	×
		あり	df.sort values(by=”平均気温”, ascending=False)	○
2	降水量とスギ花粉飛散量の散布図を作る	なし	plt.scatter(データ列 x, データ列 y, color=’sagreen’)	△
		あり	plt.scatter(df[”降水量”], df[”スギ花粉飛散量”])	○
3	降水量の変化のグラフを表示する	なし	plt.axhline(y=0, linestyle=’dashed’)	×
		あり	plt.legend()	×

した入力に対しても，文脈処理を行うことにより、ユーザの意図したコードを出力することができるようになった．

しかし，表 3 の 3 のように，文脈処理の有無に関係なく，ユーザの意図したコードと異なるコードを出力した．ユーザの入力には，処理内容の記述が不足した日本語入力も含まれているため，このような入力に対して，文脈処理では対応出来ていないと考えられる．また，本実験に使用した評価データは限られた少量のデータであった．データ量や入力の多様性を増加させた上で，改めて GoogleColab の内部情報を参照した文脈処理の効果を確認することが求められる。

5　関 連 研 究

機械翻訳において，文脈を考慮することはユーザの意図した完全な翻訳を実現するために重要であると考える．文脈情報の活用では，翻訳対象となる入力文 (原文) と翻訳先の出力文 (訳文) の直前の文を文脈として入力する手法 [9] や，原文と訳文の前後の複数文を文脈として入力する手法 [10] が提案されている．また，文脈情報の活用に限らず、コーパスに含まれない入力に対して、事前に対訳辞書を作成し、字句を変数に置換してから翻訳する手法 [11] の研究も行われている。

これらの研究は成果を示しているが、コード翻訳では、2. 2 節に示した難しさに加えて，同じ字句であっても，ユーザのコーディング状況によって，異なる意味を持つ難しさがある．ユーザによって、変数名 x に文字列や整数やリストなど異なるデータ型が代入することが想定される。そのため，事前にコーディングの文脈を追加したコーパス作成や対訳辞書の作成は難しい．

本研究では、コーディングの文脈を含めたコーパスや、変数名とデータ型の対訳辞書は作成せずに、リアルタイムにモデルの外部で処理を加えることを提案した。事前にコーパスや対訳辞書の作成に難しさがあることから、Google Colab の内部情報を参照した文脈処理により、文脈の依存関係を操作している点が異なっている。

6　む す び に

本研究では，コーディングの文脈を依存した入力に対して、ユーザの意図と異なるコードを出力してしまう課題に対して、Google Colab の内部情報を参照した文脈処理を提案した．具体的に、文脈処理の実装と、処理を想定した形式のコーパスの作成を行い、文脈に依存した入力に対して，従来の文脈処理なしと提案の文脈処理ありの翻訳結果の違いを確認した，その結果、一部のコーディングの文脈を依存した入力に対して、正しいコードの予測と、変数名を含む入力に対応しているような、よりユーザの意図したコードに近いコードを出力が可能になった。

今後は、コード翻訳精度の向上に向けて、文脈情報の活用方法について検討を重ねると共に、Google Colab の内部情報を参照した文脈処理による効果を定量的に測りたい。また，KOGI のコード翻訳機能を再度ユーザに提供し、得られたフィードバックを元に、更なる発展と改善に努めたい．

文　　献

[1] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] António V Lopes, M Amin Farajian, Rachel Bawden, Michael Zhang, and André FT Martins. Document-level neural mt: A systematic comparison. In *22nd Annual Conference of the European Association for Machine Translation*, pages 225–234, 2020.

[5] Yuka Akinobu, Teruno Kajiura, Momoka Obara, and Kimio Kuramitsu. Nmt-based code generation for coding assistance with natural language. *Journal of Information Processing*, 2022. to appear.

[6] Miyu Sato Mai Takahashi Teruno Kajiura Mayu Tomioka Yuka Akinobu Momoka Obara, Nao Souma and Kimio Kuramitsu. An ai-based chatbot for programming education on google colab. 2022.

[7] 小原 百々雅, 秋信 有花, and 倉光 君郎. Transformer に適した日本語 data augmentation ツールの実現に向けて. In *NLP* **若手の会第** *16* **回シンポジウム** *(YANS2021)*, 2021.

[8] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[9] Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. Evaluating discourse phenomena in neural machine translation. *arXiv preprint arXiv:1711.00513*, 2017.

[10] Ruchit Rajeshkumar Agrawal, Marco Turchi, and Matteo Negri. Contextual handling in neural machine translation: Look behind, ahead and on both sides. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 11–20, 2018.

[11] Zi Long, Ryuichiro Kimura, Takehito Utsuro, Tomoharu Mitsuhashi, and Mikio Yamamoto. Neural machine translation model with a large vocabulary selected by branching entropy. *arXiv preprint arXiv:1704.04520*, 2017.