DEIM2022 E31-4(day3 p24)

励ましながらプログラミング学習を支援する対話システム

富岡　真由† 　小松　 栞†† 　小原百々雅†† 　梶浦　照乃†† 　秋信　有花†
倉光　君郎††

†† 日本女子大学理学部数物科学科　〒 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: †{m1716065tm,m1816038ks,m1816019om,m1816023kt,m1616003ay}@ug.jwu.ac.jp,
††kuramitsuk@fc.jwu.ac.jp

あらまし　近年、プログラミング学習に取り組む人口は増えている。プログラミング学習は一人で行う作業も多く、途中で挫折してしまう人も少なくない。我々は、プログラミング学習者を技術だけでなく、励まし談話による精神面からもサポートできる対話システムの実現を目指している。本研究では、我々が構築を進めてきた自然言語からのコード生成 AI をベースにして、励まし談話システムの統合を行う。具体的には、事前学習済み多言語生成モデル mT5 を用い、コード翻訳コーパスと励まし談話コーパスのマルチタスキング学習を行う。本発表では、試作システムの紹介をしながら、コード翻訳と励まし談話の２つのタスクの正確さを落とすことなく、ひとつの生成モデルとして学習できることを示す。

キーワード　プログラミング支援、対話システム、自然言語処理、機械学習

1　はじめに

近年、社会のデジタル化が進み、プログラミングは現代を生きるための重要なスキルとなっている。義務教育課程においても、小学校のプログラミング教育の必修化や、中高でのプログラミング教育の拡充などこれからの日本人は全てプログラミングを学ぶ状況になっている [11]。

一方、プログラミング学習は、コンピュータを使う点で、従来の学習とは少し異質である。学習者は、コンピュータで動作確認ができるため、より自発的に課題に取り組みやすい反面、タイピングなどのささやかなミスでもエラーが発生し、全く学習が進まなくなることも多い。また、オンライン学習においては、孤独感を感じて、心が折れてしまう学生も増えている [15]。

我々は、モチベーションの維持などの精神的な面から学習支援する必要性を感じ、励まし談話を統合された対話型プログラミング学習支援システム Corgi の開発を進めている。

本論文では、コード翻訳と励まし談話の２種類のコーパスを使用したプログラミング支援システムの提案と、プログラミング支援システムの実現のための予備実験を行なった。本論文の残りの構成は以下の通りである。2 節では、Corgi の構想について述べる。3 節では、Corgi の現在の開発状況について述べる。4 節では、事前学習済み言語モデルによる言語生成モデルの開発について述べる。5 節では、mT5 によるマルチタスク学習について述べる。6 節では、関連研究を外観し、7 節で本論文をまとめる。

2　Corgi の構想

我々が開発している対話型プログラミング学習支援システム Corgi について説明する。

2.1　構　　想

Corgi は、コード生成 AI [9] を応用したプログラミング学習支援 AI システムである。コード生成 AI は、ニューラル機械翻訳 (NMT) に基づいて自然言語で記述されたプログラム意図をコードに変換する言語生成モデルである。ただし、コード生成 AI 単体では、低レベルな変換に過ぎず、意味のあるプログラミング支援をユーザに提供できない。

我々は、ユーザに意味のあるプログラミング支援を提供するため、様々な利用シーンとユーザインターフェースにあわせたシステム化を検討してきた。

- 自然言語プログラミング [7]
- リアルタイムコード翻訳システム [13]
- Visual Studio Code などの IDE への統合型 [8]

Corgi は、より初学者へのプログラミング支援、つまりプログラミング学習やそのつまづきの解消に主眼をおいたシステムを目指している。我々が考えている初学者のつまづきは以下のようなものがある。

- **言語ギャップ**: どう書いたらいいかわからない
- **意味不明なエラー**: 何が間違っているかわからない
- **孤独感**: 心が折れそう

本節の残りでは、これらについて簡単にまとめる。

2.2　言語ギャップ

プログラミングは、簡単なものではなく、多くの学習者が困難に直面している。その理由のひとつは言語ギャップにあると考えられる。プログラミング言語は、我々の日常使っている日本語とは構文や解釈の方法（意味論）が異なる。さらに、最近

日本語で書かれた意図（入力）

'data.tsv' をタブ区切りでデータフレームとして読み込む

コード生成 AI の出力結果

```
pd.read_csv('data.tsv', sep='\t')
```

図 1　コード生成 AI の例

のプログラミングでは API を活用することが求められるが、これらの API やオプションは、英語の語彙をベースにしており、相当な英語力も必要となる。

実際、プログラミング学習者からの感想を抜粋してみると、下記のとおり、言語ギャップに苦しんでいることが伺える。

- メソッド名が覚えきれない
- 多過ぎて何を覚えるべきかわからない
- いちいち調べるのが面倒くさい
- 検索結果の情報量が多過ぎて読解に時間がかかる

コード生成 AI は、このような言語ギャップを解消するため、より直接的な助けになると期待できる。図 1 は、日本語で記述されたプログラム意図を入力にして、コード生成 AI によるコード生成をした例である。我々は、日本語を日常的に使っているため、文法や表現を忘れることはあまりない。そこからヒントになるコードを提示することができれば、学習者の助けになる。

我々は、コード生成 AI による学習効果として、以下のものを期待している。

- 自然言語で正しくプログラミング概念を覚える
- 記法より、プログラムの目的に集中できる
- 検索して意識が中断する時間が減る

2.3　エラー対応

エラーメッセージは、プログラムのエラーが発生した位置を示し、エラーの原因についての簡易な説明が含まれている。プログラミングには、このようなエラーメッセージを読んで、エラーの原因を取り除く作業が必要となる。しかし、初学者はエラーの原因を理解するためのプログラミング知識が不十分なまま取り組まなければならず、初学者のみでエラーを解決するのは困難 [3] と言われている。

図 2 は、教員が示した Python コードに対し、学生がタイピングし、実際に発生させたエラーである。どちらも少しのタイピングミスでエラーが発生しているが、エラーの原因が全くわからず、自分では解決できなかった例である。このようなエラーでつまずいてしまうと、その後の演習についていけなくなり、プログラミング嫌いが増える。

また、エラーメッセージは、英語で出力されることが多い。初学者向け入門言語として多く使われる Python であっても同じである。我々は、このようなエラーメッセージに対し、プログラミング知識の理解を深めるために、解決に向けたヒントを返すことが有用であると考える。

正解コード

```
df = pd.read_csv('data.csv')
```

NameError の例

```
df = pd,read_csv('data.csv')
```

原因: NameError となっているが、`pd.read_csv` とすべき箇所を、`pd,read_csv` と構文的に間違っているため、`read_csv` の名前エラーになっている。

IOError の例

```
df = pd.read_csv('data.csv ')
```

原因: ファイル名に不要なスペースが入っているため、ファイルが読み込めなくなっている。

図 2　初学者の発生させたエラーの例

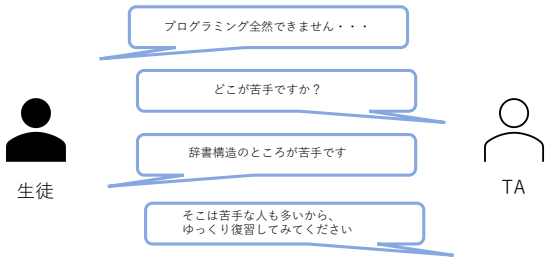

図 3　TA による励まし応答

2.4　励まし応答

プログラミングは、機械と対面した孤独な作業である。プログラミング学習者は、孤独感やエラーなどのつまずきにより、いわゆる「心が折れた」精神状態になる。特に、オンライン学習では孤独感も大きく、プログラミング演習の感想文を見ても、ほとんどの学生が一度は「心が折れそう」という精神面からの不満を報告している。

我々は、プログラミング学習において精神面からのケアも重要ではないかと考えている。この問題は、おそらく様々なアプローチによる解決案が存在すると考えられるが、我々は、特にティーチングアシスタント (TA) の役割に注目している。図 3 は、学習者の弱音に対する TA の励ましの例である。TA は単にプログラミング技術を教えるだけでなく、励ましながら学習者に接している。我々は学習支援システムも TA が励ますような応答を出力することで、少しでも学習者の精神状態を緩和できるのではないかと考えている。

3　対話型 Corgi プロトタイプシステム

3.1　対話型システム

コード生成 AI モデル [1], [9] は、自然言語記述からコードへの変換を実現する Transformer NMT モデルである。ただし、コード生成 AI モデル単体では、低レベルすぎて、意味のあるプログラミング支援をユーザに提供できない。

我々は、ユーザに意味のあるプログラミング支援を提供する

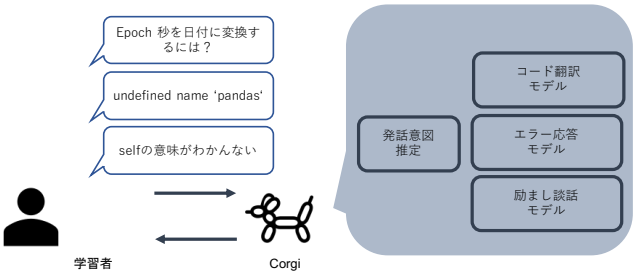

図 4　Corgi のイメージ

ため、過去に様々な利用シーンとユーザインターフェースのシステム化 [12] を検討してきた。本研究で提案するのは、対話システムとしての統合したプログラミング支援である。対話システムは、自然言語によるやりとりをベースにしたコンピュータインターフェースで、同じく自然言語をベースとしたコード生成 AI モデルと相性が良い。また、プログラミング言語より自然言語の方が表現力が高く、コーディング以外の質問応答なども自然に組み合わせることができる。我々が学習支援として重視するエラー対応や励まし談話などの機能も統合しやすい。

3.2　開発環境との統合

対話型 Corgi は、対話システムをベースにしたプログラミング学習支援システムである。本来、対話システムは、テキストベースのやりとりであるため、Slack や LINE などの既存のチャットシステムでも提供できる。しかし、Corgi の試作では、Jupyter Notebook のクラウド版である Google Colabolatory(以下 Colab) 上に独自にチャット風の UI を提供し、プログラミング環境とのシームレスな統合を重視している。

Corgi は、Colab 上の magic キーワード (例えば、`%%corgi`) をリンクされ、起動される。図 5 は、試作された対話システムの Colab 上の動作例である。この例では、ユーザが書いたプログラムにおいて、エラーが発生したとき、Corgi の対話システムは起動され、エラー分析の補助を行っている。

3.3　発話意図の推定

対話システムの利点は、ユーザとインタラクティブに対話することで、問題解決の糸口をより詳細化していくことができる点である。その際、ユーザが何を求めて発話しているのか、発話意図の推定が必要となる。

Corgi 試作システムでは、図 4 に示したとおり、単純に、正規表現によってパターンマッチで発話意図を推定し、その後の対話サブ機能に分岐している。

- **分析パターン**「〜〜は？」
  - **例.** df の値は？
- **質問パターン**「〜〜とは？」
  - **例.** イテラブルとは？
- **翻訳パターン**「〜〜するには？」
  - **例.** グラフの中で日本語を表示できるようにするには？
- **談話パターン** 上記のパターン以外の入力
  - **例.** なぜこんなことしているのかわからないよ

表 1　コード翻訳コーパスの例

日本語	Pythonコード
グラフの中で日本語を表示できるようにする	sns . set (font = 'IPAexGothic' , style = 'white')
グラフ名を <A> にする	plt . title (<A>)
<A> を縦軸、 <B> を横軸として、折れ線グラフをプロットする	plt . plot (<A> , <B>)
CSV ファイル <A> から読み込む	pd . read_csv (<A>)

各対話サブ機能は、それぞれの求められた応答を返すように開発されている。分析パターンは、Colab 上の実行環境にアクセスし、より詳細な情報を得る。一方、翻訳パターンや談話パターンは、学習済みの言語生成モデルによって応答文を得る。言語生成モデルによる応答文の作成については、次節に述べる。

4　言語生成モデルによる応答文

対話型 Corgi システムでは、言語生成モデルによる応答文が鍵となる。本節では、言語生成モデルによるコード翻訳と励まし談話システムについて述べる。

4.1　言語生成モデル

言語生成モデルは、Encoder-Decoder モデルによって、入力文からそれに対応する正解文を予測するモデルのことを指す。まず、Encoder によって入力文からその特徴を捉えた文脈ベクトルを生成し、それを受けた Decoder は入力文に対応する正しい予測文を得られるように学習を行う。2017 年には学習に必要な計算量が少なく、単語間の関係を直接的にモデル化する自己アテンションメカニズムを持つ Transformer が登場したことで、言語生成の精度を飛躍的に向上させた。

本研究では、言語生成モデルとして T5 に着目した。T5 は、分類、翻訳、要約などの様々な自然言語処理タスクを「Text-to-Text」フレームワークで解決する深層学習モデルである [4]。T5 の特徴は、統一されたフレームワークを使用することで、1 つの事前学習済みモデルを様々なタスクに転移学習できる点である。そのため、T5 と複数の種類のコーパスを使用することで、マルチタスク化も容易に行うことができる。

4.2　コード翻訳モデル

コード翻訳モデルは、自然言語で記述されたプログラム意図からコードを生成する言語生成モデルである。現在、表 1 の示す通り、日本語と Python の対訳からコーパスを作成し、日本語を説明変数、Python を目的変数として学習させることで作成する。

なお、コード生成 AI は、コード翻訳モデルに対して、変数やリテラルの特殊トークン化を行い、置き換え可能な位置を学習させている。このように前処理を加えることで、コードが混ざった日本語文であっても、コード部分を抽出し、後処理で生成されたコードと合成が可能になる。Corgi では、このような前処理/後処理を加えたコード翻訳モデルを用いる。

4.3　励まし談話モデル

励まし談話モデルは、初学者の悩みや心配ごとに対し励ます応答を返す言語生成モデルである。表 2 は、作成した励まし談

図 5　Corgi 試作型プログラミング支援システム

表 2　励まし談話コーパスの例

ユーザー発話(感想文)	応答文(感想文に対して手作業で付与)
どのようにアプローチすればよいか思いつきません。	相談してヒントをもらうことも時には大切です。
全然遅れが巻き返せないです。	コツコツ進めていきましょう。
一問あたりにかかる時間が長くなってきました。	大丈夫ですよ、解いていけば早くなっていきますよ。
周りの人が進んでいると思うと焦ってしまいます。	自分のペースで頑張っていきましょう！

話コーパスの例である。コーパスの作成手順は次の通りである。

（1） プログラミング演習の毎回の感想文を収集する

（2） 学生の気持ちが落ちていると思われる文を人手で抽出する

（3） 抽出した文に対して、励ますような応答文を付与する

5　予 備 実 験

対話型プログラミング学習支援において、自然な応答文を実現するためには言語生成モデルの活用が重要である。しかし、Corgi のように複数のタスクが含まれる場合、計算機資源の消費量が大きな問題となる。本節は、コード翻訳と励まし談話をマルチタスキング型学習を行うことで、単一の学習モデルで複数のタスクを行えるか検証する。

5.1　mT5 とマルチタスク化

T5 [4] は、分類、翻訳、要約などの様々な自然言語処理タスクを text-to-text のフレームワークとして解決する Transformer [5] ベースの言語生成モデル [4] である。BERT [2] と同様に学習済み言語モデルが公開され、多くの自然言語処理ベンチマークにおいて、トップレベルのスコアを記録している。本研究では、T5 の多言語版 mT5 [6] を用い、コード翻訳や励まし談話を転移学習する。

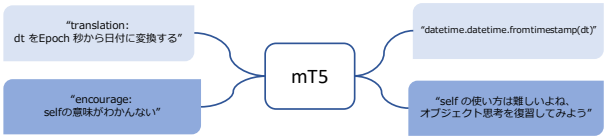

図 6　T5 によるマルチタスク化

図 6 は、学習支援システムを mT5 に適用した場合の構想図である。T5 の一つのモデルで複数のタスクを行うことができる特徴を生かして、複数のタスクを一つのモデルで実現することを考えている。今回は、コード翻訳、励まし談話の 2 つのコーパスを用意し、それぞれの入力文の先頭に `translate:` や `encourage:` などの特殊タグ付けを行う。このように特殊タグで区別することで、ひとつのコーパスで学習したモデルで複数のタスクの処理が予測できるか検証する。

5.2　マルチタスクモデルの作成

我々は、Huggingface から公開された mT5 モデルを用いて、コード翻訳と励まし談話のふたつのコーパスのマルチタスキングモデルの構築を行なった。モデルの学習環境やパラメータは以下に示す通りである。

- 計算リソース：ABCI
  - 資源タイプ：G.small
  - GPU：NVIDIA V100
- 学習パラメータ
  - Epoch:6
  - Bach Size：4
  - 最大トークン数：128

表 3　マルチタスクモデルの生成結果

コーパス		モデル
入力文	出力文	推定文
encourage: お腹が痛い	とりあえずお腹を温めよう!	ゆっくり休んでね
encourage: 犯人を逃した	早く見つけて捕まえよう!	警察に逃しに行こう
encourage: 今日のzoomで周りの生徒の進捗状況が自分よりもだいぶ早そうでとても焦りました。	大丈夫ですよ、焦らず自分のペースで進めていきましょう。	大丈夫ですよ、自分のペースで進めていきましょう。
translation: <A> の元のインデックスをリセットして、置き換える	<A> . reset_index (drop = True , inplace = True)	<A> = <A> . reset_index (drop = True)
translation: エクセルファイル <B> を読み込んで表データに格納し、<A> にする	<A> = pd . read_excel (<B>)	<A> = pd . read_excel (<B>)
translation: パソのマーカーを用いる	marker = 'x'	marker = 'x'

表 4　コード翻訳の比較

学習方法	BLEU 値
オリジナル版	0.78791
マルチタスク版	0.78856

表 5　励まし談話の比較結果

モデル	○	△	×
オリジナル版	0.432	0.270	0.297
マルチタスク版	0.457	0.240	0.304

5.3 実験結果

まず、マルチタスク学習による定性的な言語生成結果を示す。表 3 は、マルチタスクモデルによる入力文に対する言語生成の結果である。コード翻訳と励まし談話がお互いに混ざらずに、生成できていることが確認できた。

次に、マルチタスキングによる正確さへの影響を調べる。これは、マルチタスク処理を行っていない単体のコーパスを学習させたモデル（オリジナル）と正確さを比較することによって行った。

コード翻訳では、オリジナル版とマルチタスク版の正確さを機械翻訳の代表的な自動評価尺度 BLEU を算出し、表 4 の示す通り、その比較を行った。BLEU 値から、マルチタスク化を行っても精度が低下していないということを確認することができた。

励まし談話では、生成された応答を以下の 3 つに分類し、生成された励まし応答文を評価した。

○　：入力文に対応した励ましを返している

△　：会話は破綻していないが、“励まし”として違和感がある

×　：会話が破綻している

表 5 は、オリジナル版とマルチタスク版の比較結果である。比較から、励まし談話の品質も大きく低下していないことが確認できた。

6 関連研究

本節ではプログラミング支援システムについて関連研究を示す。

6.1 プログラミング入門教育対象の授業支援システム

長谷川らはプログラミング授業のリアルタイム支援システム「IDISS」を開発した [14]。これは、授業中にリアルタイムに課題に対する学習者のソースコードに対して評価を送ることができるシステムである。また、課題の提出履歴の分析を行うこともできる。これを使用することにより、アンケートでは得られなかった、学習者の試行錯誤などの詳細や課題に対するエラーのポイントなどを分析することができている。

6.2 思考分析に基づいた初学者支援システム

松井らは擬似的な対話を用いたプログラミング学習支援システムの提案を行なっている [10]。これは、プログラムを入力することでそのプログラムに対してのヒントやアドバイスを受け取ることができるというシステムである。こちらは、対話システムに機械学習モデルなどを使用せず、入力された文字列から特定の文字列を記述すると、その文字列に関連づけられたあらかじめ用意されていた質問が提示されるというものである。

7 むすびに

本論文では、対話型プログラミング学習支援システム Corgi の構想と開発状況について述べた。事前学習済み多言語生成モデル mT5 を用いることで、コード翻訳と励まし談話のマルチタスク化を実現した。また、実験では、コード翻訳と励まし談話の 2 つのタスクの正確さを落とすことなく、一つの生成モデルとして学習可能であることを示した。今後は、エラー対応機能を学習させたマルチタスク対話モデルの作成を目指すとともに、各機能の精度をさらに高め、対話型 Corgi システムの完成を目指す。

文　献

[1] Yuka Akinobu, Momoka Obara, Teruno Kajiura, Shiho Takano, Miyu Tamura, Mayu Tomioka, and Kimio Kuramitsu. Is neural machine translation approach accurate enough for coding assistance? In *Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code*, BCNC 2021, page 23–28, New York, NY, USA, 2021. Association for Computing Machinery.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Jonathan P. Munson and Elizabeth A. Schilling. Analyzing novice programmers' response to compiler error messages. *J. Comput. Sci. Coll.*, 31(3):53–61, jan 2016.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[5] 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.

[6] 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 transf ormer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Com-

putational Linguistics.

[7] 高野 志歩, 田村 みゆ, 富岡真由, 秋信 有花, and 倉光 君郎. 擬似コードから考える自然言語を活かしたプログラミング言語. In 情報処理学会情報教育シンポジウム *(SSS2021)*, 2021.

[8] 高野 志歩, 秋信 有花, and 倉光 君郎. 部分的な日本語記述からコード候補を提示する統合開発環境に向けて. In 第 *24* 回プログラミングおよびプログラミング言語ワークショップ *(PPL2022)*, 2022.

[9] 秋信 有花, 梶浦 照乃, 小原 百々雅, and 倉光 君郎. 自然言語を用いたコーディング支援の実現に向けたニューラル機械翻訳ベースのコード生成. In 情報処理学会第 *136* 回プログラミング研究発表会 *(PRO136)*, 2021.

[10] 松井大成 and 伊藤恵. プログラミング行動中の思考分析に基づいた初学者支援システムの提案. 2018.

[11] 水野修治. ぺた語義：大学入学共通テスト新科目「情報」～これまでの経緯とサンプル問題～. 情報処理, 62(7):326–330, jun 2021.

[12] 小原 百々雅, 秋信 有花, 高野 志歩, and 倉光 君郎. Transformer ベースの nmt が統合されたプログラミング言語処理系. In 情報処理学会第 *136* 回プログラミング研究会, 2021.

[13] 小原 百々雅, 梶浦 照乃, 秋信 有花, and 倉光 君郎. リアルタイムコード翻訳によるプログラミング学習支援 ai に向けて. In 第 *20* 回日本データベース学会年次大会 *(DEIM2022)*, 2022.

[14] 長谷川伸, 松田承一, 高野辰之, 宮川治, et al. プログラミング入門教育を対象としたリアルタイム授業支援システム. 情報処理学会論文誌, 52(12):3135–3149, 2011.

[15] 岡本 雅子. ぺた語義：はじめてのプログラミングとつまずき. 情報処理, 56(6):580–583, may 2015.