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

小原百々雅† 秋信 有花†† 梶浦 照乃† 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: †{m1816019om,m1616003ay,m1816023kt}@ug.jwu.ac.jp, ††kuramitsuk@fc.jwu.ac.jp

あらまし プログラミング教育は広がりを見せているが，プログラミング言語は海外で開発され，英語ベースの言語設計になっている．我々は，ニューラル機械翻訳技術を活用し，日本語で入力した意図を Python コードに翻訳する AI システムの構築を行っている．本発表では，リアルタイムコード翻訳としてプログラミング学習支援に用いた経験を報告する．

キーワード 機械翻訳，自然言語処理，機械学習，プログラミング教育

1 はじめに

今日，社会のデジタル化が進み，プログラミング教育が盛んになってきている．しかし，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．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．

我々は，このような言語ギャップを解消するため，日本語で記述したプログラムの意図から Python コードを生成する機械翻訳モデルの開発を行っている．一般の機械翻訳モデルと区別するため，本論文では「プログラミング学習支援 AI モデル」と呼ぶ．本研究では，プログラミング学習支援 AI モデルを実際の学習者に提供するため，リアルタイム翻訳システムを開発した．

リアルタイム翻訳システムは，Google Translator や DeepL などの自然言語間の機械翻訳において，採用される機械翻訳システムの UI/UX である．特徴は，ユーザが入力の途中であっても，順次，機械翻訳の結果が提示される点にある．もし，意図にあわない翻訳が提示されたとき，入力を修正しなおすことで，翻訳結果を調整することができるため，コード翻訳のような，常に正解が表示されない用途では有用である．

本研究では，プログラミング学習支援 AI を用いたリアルタイムコード翻訳システム Corgi を開発した．また，日本女子大学の全学向けのデータサイエンスの入門講義の演習時間において課題を解くのに活用してもらい，実践形式での評価を行った．なお，プログラミング学習支援 AI モデルの構築やその応用であるリアルタイム翻訳システムは，まだ研究の初期段階と言える．本実践の結果は，予備的なものといえるが，学習者からのフィードバックを得ることで，今後のプログラミング学習支援 AI への知見が得られた．

本論文の残りは，以下の通りである．第 2 節では，プログラミング学習支援 AI モデルを紹介する．第 3 節では，リアルタイムコード翻訳システムについて述べる．第 4 節では，教育実践について報告する．第 5 節では，関連研究を外観し，第 6 節で本論文をまとめる．

2 コード生成 AI

コード生成 AI [1] は，「自然言語で書かれたプログラム意図からコードを生成する」機械翻訳である．Transformer をベースとした言語生成モデルに対して，自然言語記述とコードをペアにした教師データを学習して構築する．本論文の執筆時点で，ホールドアウト法によるテスト検証によって，90%程度の正解率が得られる．

2.1 原 理

コード生成 AI は，Transformer をベースとした言語生成モデル (Encoder-Decoder アーキテクチャ) を用いたニューラル機械翻訳である．特徴は，自然言語とコードを対訳ペアとした教師データを用いる点で，翻訳された自然言語の代わりに，図 1 のように意図にあったコードが生成される．

このようなコード生成は，ニューラル機械翻訳，つまり Transformer を用いた機械学習によって実現されている．図 2 は，Transformer をベースにしたコード生成の概要を示している．入力された自然言語文は字句解析をされ，多次元の系列ベクトル $x = (x_1, ..., x_n)$ になる．Encoder は，この入力を文脈ベクトル $z = (z_1, ..., z_n)$ に変換し，Decoder では z から出力系列 $y = (y_1, ..., y_n)$ を生成する．

Transformer の優れている点は，文脈ベクトルに入力と出力間

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

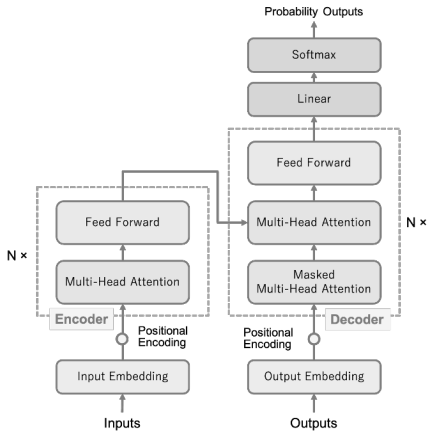

図 2　Transformer のアーキテクチャ

プログラム意図	コード
A が偶数であるかどうか	`A % 2 == 0`
CSV ファイル A を読み込む	`pd.read_csv(A)`
カンマ区切りで A を読む	`pd.read_csv(A, sep=',')`

図 3　コード・コーパスの例

の確率的な重みを加えることで，字句間の複雑な関係性をより高い精度で捉えている点である．しかし，ソースコードや抽象構文木のような深い木構造の関係性まで，正しく捉えられないと言われている．コード生成 AI は，この辺りの特徴を留意し，生成モデルの学習を行う．

2.2　コード・コーパスと機械学習

コード生成 AI では，自然言語記述とコードをペアにして集めたデータをコード・コーパスと呼ぶ．コーパスは，言語生成モデルの機械学習を行うとき，教師データ，検証データに使われる．

図 3 は，自然言語部を日本語，コード部を Python としたコード・コーパスの例である．コード・コーパスのコードは，原則，関数/メソッド適用や演算などの式 (Expression) に相当するコード片である．そのコードの意図を日本語の記述で与えることでペアを構成する．プログラム意図の記述は，直訳的な記述でも，意訳的な記述でも，あるいは省略的な記述でも構わない．コードは，`n % 2 == 0` のように 2 つ以上の式が複合されても構わない．（自然言語として，自然に意図を記述できる単位で記述してもよい．）

一般に，言語生成モデルの精度をあげるには，教師データの量は重要である．我々は，データ拡張法 [2] を応用した手法を導入し，コード・コーパスを合成的に増やすことで十分な精度が得られる学習を達成している．

2.3　コード生成モデルと正確さ

コード生成 AI は，言語生成モデルのアーキテクチャ，コード・コーパスの量によって，コード生成の正確さは大きく変わる．本研究では，以下の通りに学習し得られた言語生成モデルを用いる．

- **言語生成モデルのアーキテクチャ**:
  - T5（Text-to-Text Transfer Transformer）[3]
- **事前学習**
  - 多言語版 T5(google/mt5-small) [4]
  - mC4（multilingual Colossal Clean Crawled Corpus)
  - 日本語を含む 101 の言語を含むデータセット
- **コード・コーパス**:
  - データサイエンスの教科書から作成 [5]
  - 対訳ペア: 約 21,000 件

最終的に，上記の組み合わせで構築されたコード生成モデルは，ホールドアウト法によるテスト検証によって，90%程度の正解率が得られた．ここで正解率については注意が少し必要である．正解率とは実行可能かつ入力された意図にマッチしたコードが得られた頻度で，いわゆる直感的な正解率に反しない．しかし，コーパス作成の元になったデータサイエンスの教科書に掲載されていないコードは生成されない．つまり，コード・コーパスに依存した正解率と言える．

3　Corgi: リアルタイムコード翻訳システム

本節では，我々が提案するリアルタイムコード翻訳システムについて説明する．

3.1　構　　想

Corgi は，コード生成 AI [5] 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 単体では，低レベルな変換に過ぎない．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．

- リアルタイムコード翻訳
- 自然言語プログラミング [6]
- Visual Studio Code などの IDE への統合型 [7]
- 対話システムによるプログラミング学習支援 [8]

リアルタイムコード翻訳は，Google 翻訳や DeepL などのリアルタイム翻訳に着想を得て，もっとも初期段階に構想された支援システムである．

3.2　リアルタイムコード翻訳

リアルタイムコード翻訳はリアルタイム翻訳のコード版である．図 4 に見られるとおり，日本語で書きたいプログラムの意図を入力すると，その左側にコード生成 AI で生成された Python コードが表示される．入力テキストボックスのキーイベントを得ることで，日本語文を書いている途中から逐次的に Python コードが表示される．

なお，初期のコード生成 AI は，50%弱程度の正解率であった．我々がリアルタイムコード翻訳に注目した理由として，コード生成の誤りに対してロバストな UI/UX な点があげられる．実際，もし間違ったコード生成が得られても，ユーザは入力し直すことで適切なコードが得やすい．

3.3　プログラミング演習環境

Corgi は，プログラミング演習とシームレスに利用できるよ

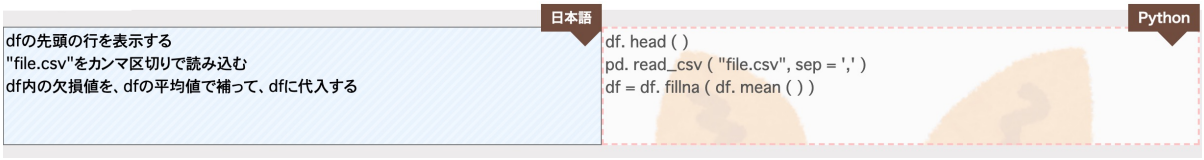

図 4　リアルタイムコード翻訳のインターフェース

図 5　Corgi システムの全体像

表 1　演習課題の一覧

基本問題	1	'movie2021.csv' を読み込む
	2	data の件数を調べる
	3	data の' 地雷フラグ' 列が 1 の値のデータ数
	4	data の' 友人におすすめ' 列の平均値を得る
	5	data の' 友人におすすめ' 列を順に並べ変える
	6	data の' ワクワク' 列のヒストグラムを表示する
	7	data の' 友人におすすめ' 列と最も相関が高い属性を調べる
	8	data の' ワクワク' 列を x 軸,' 社会派' 列を y 軸とした散布図を書く
応用問題	1	KMean 法で 4 クラスターに分類し，主成分分析で 2 次元に圧縮して分布図を表示する
	2	回帰アルゴリズムを用いて，' 友人におすすめ' 列の点数を予測する
	3	2 クラス分類（ロジスティック回帰，決定木，サポートベクターマシンなど）を用いて ' 地雷フラグ' 列を予測する

うに，Google Colaboratory(Colab) 上で動作するようになっている．図 5 は，我々が開発した Corgi システムの全体像を示している．Corgi は，Github 経由でユーザに配信され，Google アカウントを所持していれば誰でも Corgi を使うことができる．また，Python の実行セルにおいて，corgi() コマンドを実行するだけで，いつでもリアルタイムコード翻訳の UI を Colab 上の画面に表示できる．

コード生成 AI は，T5 を使用しているため，コード生成時に，少なくない計算機資源が必要となる．Corgi のコード生成は，同じく Colab 上で実行でき，必要であれば Colab の GPU も利用できる．そのため，人数の多い教室で計算機リソースの不足を心配することなく利用できる．

また，Corgi では，ユーザの入力やコード生成の誤りを把握できるように，Slack ベースのログ記録を用意している．これにより，授業中であっても簡単に Corgi の動作を確認することができ，学生への対応に活かすことができる．

図 6　想定する Colab 上での回答例（抜粋）

表 2　アンケート回答者に関する情報

学科	学年	授業以外でのプログラミング経験	プログラミングに対する苦手意識
教育学科	2	なし	ある
心理学科	2	3 ヶ月以内	ある
心理学科	2	3 ヶ月以内	どちらでもない
化学生命科学科	3	3 ヶ月 ～ 半年	ある
家政経済学科	3	半年 ～ 1 年	どちらでもない
現代社会学科	3	3 ヶ月以内	ある
住居学科	3	なし	ある
住居学科	3	なし	どちらでもない
心理学科	3	なし	ある

4　実 践 報 告

我々は，第 3 節で述べた Corgi を実際に学生に利用してもらい，実践形式で評価を行った．

4.1　概　　要

我々は，日本女子大学の全学向けの情報科目である「AI 入門」において，受講生に Corgi を提供し，その利用経験のフィードバックを得た．授業概要は，以下の通りである．

- 授業内容：Python によるデータサイエンス入門
- 実施日：2021 年 12 月 22 日
- 人数： 33 名 (対面 5 名, 残りはオンライン)
- 授業形態：総合演習（1 人 1 台 PC を使用）

なお，Corgi を活用したのは，最終回の総合的な演習課題を解く回だけとなった．これは，我々が意図して選んだわけでなく，Corgi の開発進捗からきた日程による．ただし，具体的な演習課題に取り組むことで，Corgi の利用範囲を予想しやすくなった．

演習課題は，表 1 に示す通り，基本問題と応用問題からなる．基本問題は，Python によるデータ分析の課題で，pandas や matplotlib の API を正しく使えれば，1 ステートメントで答えが得られる課題である．応用問題は，クラスタリングや回帰分析など機械学習が含まれた内容であり，複数のステートメントを組み合わせて解く必要がある．図 6 は，Colab 上による想定する回答例（抜粋）である．

なお，演習課題を解くため，Corgi 以外に講義資料やネット上の記事，書籍等の参考資料の閲覧は自由とした．

4.2　アンケート調査

我々は，Corgi の利用体験に関して，フィードバックを得るためアンケートを行った．アンケートの主な項目は，図 7 に示す通りである．有効回答数は，9 件であった．以下，アンケート結果について述べる．

4.2.1　学 生 属 性

まず，今回の実践に参加した学生の属性から見ていきたい．

学生属性の調査

A： アンケート回答者の属性

B： プログラミングに対する苦手意識

C： 授業以外でのプログラミング経験

Corgi へのフィードバック

D： Corgi の使用状況に関する調査（5 段階評価）

（1） 何も見ずにプログラムを書くことはできたか

（2） Corgi を見ることで助けやヒントは得られたか

E： Corgi のシステム評価（複数選択可）

（1） システムの良かった点

（2） システムの改善すべき点

（3） 追加で欲しい機能

F： Corgi への期待度（5 段階評価）

（1） Corgi があるとプログラミング学習が続けられると思うか

（2） どのようにプログラミングしたら良いかわからないとき，Corgi は役立つと思うか

（3） Corgi があると，細かいコードの書き方にとらわれず内容に集中できると思うか

（4） 今後も Corgi を利用したいと思うか

G： Corgi に対する意見・感想（自由記述）

図 7　アンケート項目

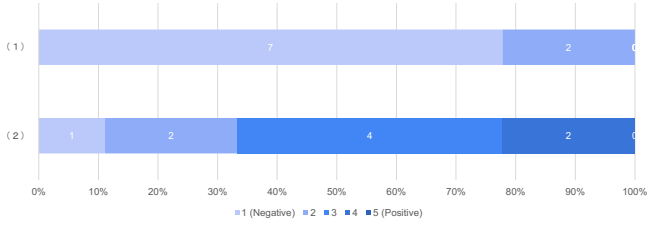

図 8　D：Corgi の使用状況に関する調査（基本問題）

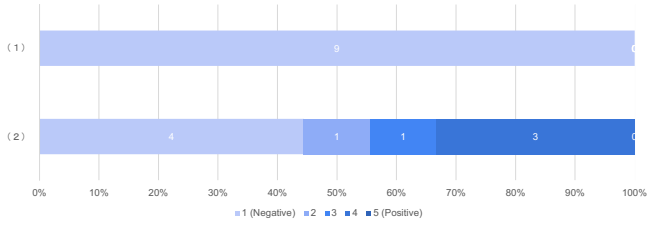

図 9　D：Corgi の使用状況に関する調査（応用問題）

表 2 は，『A : アンケート回答者の属性』と『B : プログラミングに対する苦手意識』，『C : 授業以外でのプログラミング経験』をまとめたものである．全員が情報科学を専門としておらず，プログラミング経験は授業外でほとんどなかった．また，プログラミングに苦手意識がないと回答した学生もいなかった．

4.2.2　Corgi へのフィードバック

『D : Corgi の使用状況に関する調査』は基本問題と応用問題，それぞれの回答時について行った．結果を基本問題は図 8，応用問題は図 9 に示す．「(1) 何も見ずにプログラムを書くことはできたか」という問いに対しては，基本問題・応用問題共に 9 名（100%）の学生が「できなかった」あるいは「ややできなかった」を選択した．「(2) Corgi を見ることで助けやヒントは得られたか」では，基本問題では「得られなかった」あるいは「やや得られなかった」を選択した学生は 3 名（33.3%）であったのに対し，応用問題で「得られなかった」あるいは「やや得られなかった」を選択した学生は 5 名（55.5%）と過半数であった．

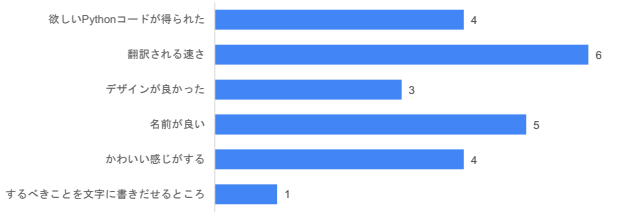

図 10　E - (1)：良かった点

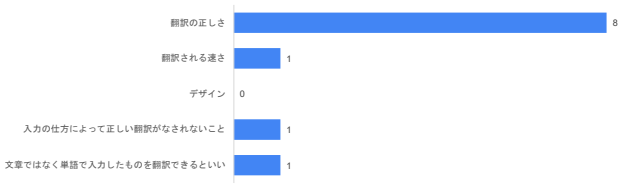

図 11　E - (2)：システムの改善すべき点

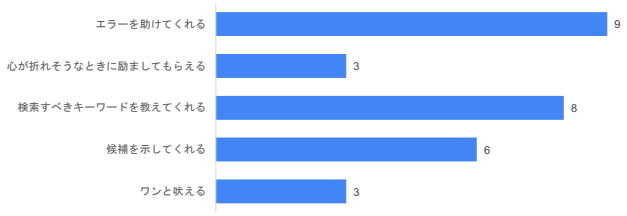

図 12　E - (3)：追加で欲しい機能

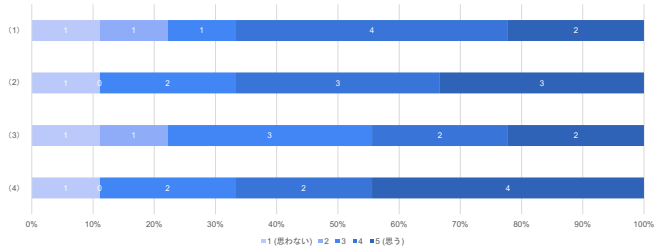

図 13　F：Corgi への期待度（5 段階評価）

表 3　Corgi に対する意見・感想（自由記述）

1	シンプルで使いやすく，講義資料と合わせて使うととても便利だった
2	最初誤翻訳が出ても入力文の書き方を変えると正解が出る物も多く，わからなくても教えてもらえるという安心感があった
3	Corgi に助けられてレポート課題に取り組むことができそう
4	思っていたよりも誤翻訳が多かった
5	ある程度プログラミングの知識がある人には大いに役立つのだと思う
6	ラバーダックのように使うこともできるのかなと思った

図 10〜12 は『E:Corgi のシステム評価（複数選択可）』の結果である．「(1) 良かった点」では「翻訳される速さ」，「(2) 改善すべき点」では「翻訳の正しさ」がそれぞれが 6 名（66.7%），8 名（88.9%）と一番多く選択された．「(3) 追加で欲しい機能」

表 4　自然言語プログラミング

目的	ソース	生成物	想定利用者	実現手法
プログラミング支援 [9]	英語	形式言語	エンドユーザ	構文木マッピング
Excel アドイン [10]	英語	表計算プログラム	エンドユーザ	意味解析
コーディング支援 [11]	英語	Python	全てのプログラマー	NN モデル
コード理解支援 [12]	Python	擬似コード (英/日)	プログラミング初学者	SMT
コード理解支援 [13]	Java	コメント (英)	開発者	NMT(seq2seq)
プログラミング体験の向上 [14]	英語	LOGO プログラム	エンドユーザ	ルールベース
例外指示 [15]	日本語	SDC(意味構造)	自動運転システム	SDC/CCG
本研究: プログラミング支援	日本語	Python	プログラミング初学者	NMT(mT5)

では，「エラーを助けてくれる」，「検索すべきキーワードを教えてくれる」，「候補を示してくれる」の 3 つがそれぞれ 9 名（100%），8 名（88.9%），6 名（66.7%）と多くの回答を得た．

図 13 は『F :Corgi への期待度（5 段階評価)』の結果をまとめたものである．「(1) Corgi があるとプログラミング学習が続けられると思うか」に対しては，6 名（66.7%）の学生が「思う」あるいは「やや思う」を選択した．「(2) どのようにプログラミングしたら良いかわからないとき，Corgi は役立つと思うか」に関しても，6 名（66.7%）の学生が「思う」あるいは「やや思う」を選択した．「(3) Corgi があると，細かいコードの書き方にとらわれず内容に集中できると思うか」では，「どちらでもない」を選択した学生が 3 名（33.3%）と一番多い結果となり，「思う」あるいは「やや思う」を選択した学生は合わせて 4 名（44.4%）であった．「(4) 今後も Corgi を利用したいと思うか」では，6 名（66.7%）の学生が「思う」あるいは「やや思う」を選択した．

表 3 は，『G：Corgi に対する意見・感想（自由記述)』として得られた回答を示したものである．この設問では，6 名の学生から回答が得られた．「わからなくても教えてもらえるという安心感があった」や，「Corgi に助けられてレポート課題に取り組むことができそう」などポジティブな意見が大半であったが，「思っていたよりも誤翻訳が多かった」という改善すべき点の指摘も得られた．

4.3　得られた知見

本実践とアンケートの結果から，多くのプログラミング学習者が Corgi によってプログラミング時の助けを得たことが明らかとなった．また，多くの受講生が「Corgi を使うことでその後のプログラミング学習が続けられそう」と感じていたことから，Corgi は長期的な目線で見ても，プログラミング学習者を支援可能なシステムであると考えられる．しかし，細かいコードの書き方にとらわれず内容に集中できると思うと回答した受講生は過半数に満たなかった．これは，コード生成 AI モデルの精度が原因だと考えられる．今後は，回収したログの分析を進めていき，コード生成 AI の翻訳精度の向上に取り組んでいきたい．

5　関 連 研 究

自然言語によるプログラミングは，コンピュータサイエンスの黎明期から多くの研究者を惹きつけてきた．表 4 は，近年の研究発表を目的，ソース，生成物，想定利用者，手法の観点からまとめたものである．研究目的や想定利用者は多岐にわたり，自動運転システムにおける音声例外指示のような音声処理技術との統合を目指した研究も報告されている [14], [15].

実現手法は，構文木へのマッピングや意味解析などのルールベースの手法 [9], [10] から，統計的機械翻訳 (SMT) やニューラル機械翻訳 (NMT) を用いたモデルによる生成手法まで多数報告されている [11]．近年では，機械翻訳の発展に伴い，機械翻訳技術を手法とした研究が増えてきている．自然言語間 (e.g.: 日本語 → 英語) の翻訳タスクを応用することで，自然言語からコードへの変換も可能となる．また，機械翻訳を用いた手法は，単純に自然言語からコードだけではなく，コードから自然言語という逆変換も実現可能である [12], [13].

6　む す び に

本研究では，日本語で記述したプログラムの意図から Python コードを生成するプログラミング学習支援 AI を用いて，リアルタイムコード翻訳システム Corgi を開発した．また，Corgi を実際に学生に利用してもらい，実践を報告した．実践結果から，どのようにプログラミングしたら良いかわからないとき，Corgi は役立つと感じた学生が多く見られたことから，Corgi をプログラミング学習支援に活用していくことは有用と考えられる．今後は，翻訳の正確さを向上させ，より的確な学習支援を目指していきたい．

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] 秋信 有花, 小原 百々雅, 梶浦 照乃, and 倉光 君郎. 自然言語とソースコード間の対訳コーパス向け data augmentation 手法の提案. In 言語処理学会第 *28* 回年次大会予稿集, 2022.

[3] 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. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[4] 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. 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.

[5] 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. 2021.

[6] 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.

[7] 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.

[8] 富岡 真由, 小松 栞, 小原 百々雅, 梶浦 照乃, 秋信 有花, and 倉光 君郎. 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. In 第 *20* 回日本データベース学会年次大会 *(DEIM2022)*, 2022.

[9] Greg Little and Robert C. Miller. Translating keyword commands into executable code. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, page 135–144, New York, NY, USA, 2006. Association for Computing Machinery.

[10] Sumit Gulwani and Mark Marron. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *SIGMOD '14 Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 803–814. Association for Computing Machinery, June 2014.

[11] H. Fudaba, Y. Oda, K. Akabe, G. Neubig, H. Hata, S. Sakti, T. Toda, and S. Nakamura. Pseudogen: A tool to automatically generate pseudo-code from source code. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 824–829, 2015.

[12] Y. Oda, H. Fudaba, G. Neubig, H. Hata, S. Sakti, T. Toda, and S. Nakamura. Learning to generate pseudo-code from source code using statistical machine translation (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 574–584, 2015.

[13] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In *Proceedings of the 26th Conference on Program Comprehension*, ICPC '18, page 200–210, New York, NY, USA, 2018. Association for Computing Machinery.

[14] Xiao Liu and Dinghao Wu. PiE: Programming in eliza. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 695–700, New York, NY, USA, 2014. ACM.

[15] Akari Inago, Hiroshi Tsukahara, and Ichiro Kobayashi. Parsing parking instructions for self-driving cars into spatial semantic descriptions. *JCP*, 14:328–338, 2019.