プログラミングコンテスト問題からのプログラム生成用データセットの作成

Panyam Chandrasekarasastry Nagesh* 江原 遥 † 徳永 拓之 ‡ 鶴岡 慶雅 § 高村 大也 †
小田 悠介 ‖ 渡部 有隆 **

1 はじめに

プログラミングは習得に長い時間がかかり，人間にとって負担の大きい作業である．一方，自然言語は人間にとってなじみ深く，母語であれば特段大きな負担なくコミュニケーションをとることが可能である．こうした状況から，自然言語を介してプログラミング言語にアクセスする研究が盛んにおこなわれている．例えば，自然言語からプログラムを生成したり検索したりする研究があげられる．こうした研究を遂行するため，既存では，プログラムコード（以下，単にコード）とテキストの組を集めたデータセットが作成されてきている [7, 6].

しかし，これらの多くのデータセットでは，プログラミング言語を自然言語からより扱いやすくするという初期の目的とは裏腹に，テキスト部分はコードの直接の翻訳になっていることが多い．例えば，表 1 に示す StackOverflow データセット [3] では，テキスト部分の表現が，コードがどのように動作すべきかまで事細かに指示していることが見て取れる．現実的な状況では，計算機がどのような動作をすべきかという手続きが逐一わかっているわけではないことがよくある．もしそうした手続きが仔細に分かるのであれば，その手続きをプログラミング言語で明確に表現することが可能であるので，自然言語でプログラミング言語にアクセスすることの有用性は低い．

本研究では，テキスト部分が翻訳的でも手続き的でもないデータセットを提案する．表 1 の “AIZU” 部分に，提案するデータセット（以下 AIZU）の例を示す．提案するデータセットは，プログラミングコンテスト練習用のジャッジメントシステムに蓄積されたデータを活用している．プログラミングコンテスト用の問題であるため，AIZU はより挑戦的で，プログラミングコードに直接翻訳できない性質に富む．

直接的な翻訳ではないという性質に加えて，AIZU は自然言語によるプログラミング言語への容易なアクセスという当初の目的にふさわしい次の性質を持つ．テキスト部分が人手でチェックされており，プログラミング言語で明確な解答が存在するように作られている点，また，他のデータセットと異なり，1 つのテキストに対して，多くのコードが提出されている点である．

本研究の貢献は以下のとおりである．

- コード生成・コード検索という挑戦的な課題のための，非翻訳的なデータセット AIZU を提案する．
- この挑戦的課題の研究のため，公に入手可能な AIZU データセットを作成したこと[1]．
- コード検索の設定でベースラインとなる実験結果を示したこと．

2 関連研究

テキストとコードの対をなすデータセットの構築時に問題となるのは，コードに自然言語によるアノテーションを行う作業，またその逆にテキストに対応するコードを書く作業の負担が大きいという点である．したがって，既存の研究では，この作業負担の問題をどう解決するかに焦点が当たった．

作業負担を減らす代表的な方法は，何らかの別の目的でテキストとコードの対が蓄積される既存の仕組みを探し，それをデータセット作成に転用しようという試みである．例えば，[7] では IFTTT という，自然言語と，多数のユーザが作成したショートカットキーのようなコマンド列（例：メール送信，ファイルを送る，など）の組を用いた．しかし，この方法では，やはり翻訳に近いデータしか収集できていない．もう一つの典型的な方法は，GitHub（http://github.com/）のような大規模なオープンソースのコード集合を用いて，コードに埋め込まれた自然言語によるコメントを

*メルボルン大学．† 産業技術総合研究所．‡ スマートニュース株式会社．§ 東京大学．¶奈良先端科学技術大学院大学/情報通信研究機構．‖ 会津大学．

[1] https://aistairc.github.io/plu/ にて原則無償で公開予定．

Dataset	Problem	Answer
AIZU	文字列集合を Find palindromes amongst given set of String	filter (lambda x : x == “”.join(reversed(x)) , list_strings)
AIZU	Given the vertices of a triangle and a point, find if it is inside the triangle?	def inside (ax, ay, bx, by, cx, cy, x, y) : . . .
StackOver-flow	What’s the idiomatic syntax for prepending to a short python list?	s.insert(0, x)
StackOver-flow	How can python return the filename currently being processed under Windows	import os filename = os.path.basename(__file__) print (filename)

表 1: 問題-コードペアの例

活用する方法である [1]．しかし，この方法では，文書化の遅れなどによって，自然言語で表現されたコードの機能と，実際のコードの機能に差があることが多く，しかもそれを実際に確認する現実的な方法がないという問題がある．

その他のアプローチとしては，作業負担の問題に人的労力で対応するものである．例えば，[4] は Hearthstone というカードゲームにおけるカードの操作だけにタスクを限定しつつ，人手でデータセットを作成した．また，[6] はクラウドソーシングを用いて，Python の Django という大規模なソフトウェアの各行に自然言語によるアノテーションを施すことで，データセットを作成した．

3 Dataset

提案するデータセットである AIZU は，AIZU Online Judge(AOJ)[2]から取得した．AIZU Online Judge は，**オンラインジャッジサイト**と呼ばれる，プログラミングコンテストの練習用のウェブサイトである．ユーザは，多数の問題からプログラミングの問題を選び，その問題を解くコードを投稿することができる．投稿されたコードが実際に問題に対する回答になっているかの判定（**ジャッジ**）は，あらかじめ問題作成者によって問題とともに用意された**テスト用入出力**を用いて自動的に確認できる．オンラインジャッジサイトは，投稿されたコードをコンパイルしてプログラムを実行し，テスト用の入力を与える．プログラムの出力が実際にテスト出力と同一であったら，妥当，そうでなければ不妥当と判定される．コードは，Python や C/C++など，様々なプログラミング言語で投稿することが可能である．

こうしたオンラインジャッジサイトは AOJ 以外にも多数あり，Peking Online Judge(`http://poj.org/`) などが有名である．AOJ は，日本では長期の運用実績のあるサイトであり，日本語の問題を多く含む．オンラインジャッジサイトでは，プログラミング問題に一律の番号を振って問題を特定しやすくすることが行われており，AOJ でも 4 桁の番号で管理されている．問題文は一部英語に翻訳されているものの，大部分は日本語である．データセット作成の目的のため，公開部分のデータセットについては，すべて人手の翻訳者による問題の英語訳を付けた．

[2]`http://judge.u-aizu.ac.jp/onlinejudge/`

性質	Stack-Overflow	AIZU
コード数	55,882	1,987
問題数	55,882	66
妥当な (問題・コード) 対の数	55,882	1,987
不妥当な (問題・コード) 対の数	55,882	1,987
問題の平均トークン数	9	15
回答の平均トークン数	20	154

表 2: 各データセットの性質．妥当とは，コード部分が実際に問題部分で示された問題を解くことが確認されたことを示す．

表 2 に，提案するデータセットの性質を示す．既存のデータセットで典型的な StackOverflow データセットに比べて，AIZU データセットでは，問題の平均トークン数が大きいことがわかる．これは，AIZU データセットにおいては，より長い自然文の文脈をコード生成のために理解しなければならず，このデータセットにおける問題が挑戦的であることを示唆している．

また，我々は，データセット中の問題のうち 100 問を実際に人手で分析して表 3 に示した．実際に人間が問題を解いた場合でも，正しいコードを提出できるケースが半数を下回ることからも，このデータセット

における問題が挑戦的であることがわかる．また，このデータセットでは，妥当な回答の数以外にも，提出されたコードの数も，問題の難しさを示唆していることも，表 3 からわかる．簡単な区分けの問題に対して提出されたコードの数は，難しい区分けの問題に対して提出されたコードの約 2 倍にのぼる．

4 実験

4.1 比較対象のデータセット

AIZU 問題は AOJ のうち，特に簡単な**パソコン甲子園**の問題を選んで用いた．これは，高校生が解くことを想定したプログラミングの問題である．AOJ では複数のプログラミング言語で投稿することが可能であるが，今回は，Python 言語に限定して実験を行った．これは，Python 言語によるコードが Java 言語や C 言語などと比較して短く，また，コードの構文解析も容易であり，Python 言語の利用者が多く十分な投稿数があること，などが理由としてあげられる．

StackOverflow 比較対象のデータセットとしては，StackOverflow[3] を用いた．これは，プログラミングの質問サイトである StackOverflow から，質問とコードを自動抽出したものである．StackOverflow は質問サイトであり，質問者や他の読者が理解できればよいため，自然言語を用いて回答しても良い点が，AOJ などのオンラインジャッジシステムと大きく異なる．そのため，コード中に自然文が含まれていたり，コード部分が一部の抜粋であるなどの理由で，投稿されたコードの 3 割は実行できなかった．

どちらのデータセットも，テキストとコードの対形式であることは共通しているが，問題の性質が異なる．AIZU コーパスの問題は，論理や推論，アルゴリズムなどの知識を必要とするものがほとんどである．一方，StackOverflow では，そのような問題もないわけではないものの，Python 言語の文法やライブラリの使い方に関する質問が大半を占めており、質問に対応するコードがパターン化しており、推論などの知識を使わないという意味で直訳的である．

4.2 評価

こうした挑戦的な課題におけるコード生成は，推論と翻訳の 2 つの段階に分割して考えることができる．まず，自然言語で与えられている問題に対して，論理的推論を用いて，問題を解く解答を，擬似コードのような自然言語の枠組みに収まる言語で記述する段階である．次に，自然言語とプログラミング言語という異なる言語間で，翻訳を行う段階である．AIZU コーパスの問題は，問題からコードに到達するまで推論と翻訳の 2 段階を要する種類の課題とみなすことができるのに対して，StackOverflow コーパスの問題は，問題からコードに到達するまで翻訳の 1 段階のみを要する種類の課題とみなせる．そこで，これらの課題の性能差をみるため，次の 2 つの設定を用意した．

- 妥当なペアの識別: 与えられた問題-コードのペアに対して，コード部分が問題を解くコードか否かを識別する，二値分類タスクである．このタスクは，コード生成のサブタスクとみなすことができる．コード部分が問題を解くかどうかを正確に判定できたとする．この場合，問題に対して多数の正解コード候補を用意して，コードが問題を解くかどうかを識別する方法で，正しいコードを得ることが期待できる．
- コード検索: 一方で，妥当なペアの識別タスクは，コード検索の観点からも有用である．コード検索の観点からは，まず，GitHub のような大規模なコードベースが与えられた状況を想定する．そうしたコードベースには多数のコードが含まれている．コード検索は，そうした多数のコードの中を，与えられた問題を解いていそうな順にコードを並び変える，というタスクとみなせる．

実験設定 AIZU も StackOverflow も，問題とそれを解くコードの対を含む．これらの対は，前述の識別問題における正例とみなすことができる．負例については，問題に対して，他の問題を解くコードを対にして作成した．正例対と負例対の数が同数になるように設定した．どちらのコーパスでも，訓練/開発セットとテストセットを 60 : 40 の比率で分割した．この分割の際には，訓練セットとテストセットに同じ問題が含まれることのないようにした．

テキストと画像といったような，異なる 2 種のドメインでの対に対する二値分類タスクに有効な手法として，Siamese Neural Network[5] が提案されている．本研究ではこの研究を参考に，図 1 に示す構造のニューラル識別器を作成した．実装は Keras で行い，クロスエントロピー損失を Adam を用いて 20 エポックで減少させた．

コード検索の設定では，[3] と似た評価設定を用いた．ここでは，問題に対して，コード集合から正しいコードを取得する性能を測定したい．そこで，テストセットの各問題に対して，1 個の正しいコードと 49 個の無関係なコードからなる 50 個のコード集合を用意し，上記の識別器を用いたときの検索性能を測定す

難しさ	問題数	総投稿数	妥当な投稿数	例
Hard	19	18,625	42%	Find if a given rectangle is convex ?
Medium	19	22,865	48%	Find the reflection of a point over a line.
Simple	28	32,663	50%	Count palindromes amongst given strings.

表 3: AIZU データセットの複雑さ.

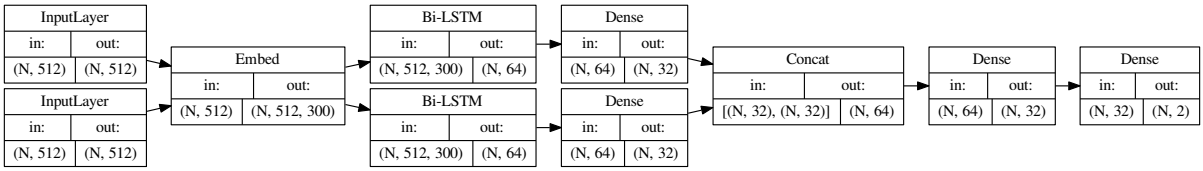

図 1: 作成した二値分類器の構成図.

る．検索性能の指標としては，“Mean Reciprocal Rank (MRR)” を用いた．これは，正例を高く順位付ける性能を表す指標であり，値が高いほどよい．1.0 で，正例より高く順位付けられている負例が存在しないことを表す.

4.3 結果と考察

識別器としては，前述の二値識別器を用いた．Support Vector Machines などの古典的な識別器でも同様の設定で性能を評価したが，前述の二値識別器より低い正解率しか得られなかったので，本稿では割愛する.

表 4 に実験結果を示す．識別設定では正解率 (Accuracy) を，コード検索設定では MRR を性能指標に用いた．正解率については，ベースラインとなる 0.5 から有意に高い正解率であるかを，片側二項検定で検定した．有意水準 0.05 において，どちらのデータセットでも統計的有意な結果を得た.

これらの実験においては，コード中の変数名が性能を向上させている可能性がある．例えば，問題に “ball” という語が含まれていたら，“ball” という変数を含むコードを探すことによって高い性能が得られてしまう懸念がある．この懸念を確認するため，変数名をランダム化した設定でも実験を行ったところ，識別設定では正解率が 53%にまで低下した．しかし，図 1 における “Bi-LSTM” の層の直後に Batch Normalization[2] を入れることにより，正解率は 55% にまで向上した（片側二項検定で有意水準 0.01 で統計的有意）.

Corpus	二値分類正解率	P 値	Code 検索設定での MRR
AIZU	55%	0.017	0.16
StackOverflow	65%	0.010	0.08

表 4: 識別設定，コード検索設定での性能評価.

5 おわりに

本研究では，自然文によるコード生成･コード検索という挑戦的課題の研究を遂行するためのデータセットを提案した．提案するデータセットは，プログラミングコンテストから取得した挑戦的な問題に富み，公開されているため誰でも自由にダウンロードして利用することができる．本稿では，コード検索の設定においてベースラインとなる性能を提示した．より広くデータセットを使っていただき，この分野の研究が進むことを望む.

謝辞

この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構 (ＮＥＤＯ) の委託業務の結果得られたものです.

参考文献

[1] Antonio Valerio Miceli Barone and Rico Sennrich. A parallel corpus of Python functions and documentation strings for automated code documentation and code generation. *arXiv:1707.02275 [cs]*, July 2017.
[2] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.
[3] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *ACL*, 2016.
[4] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. Latent Predictor Networks for Code Generation. In *ACL*, pp. 599–609, 2016.
[5] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pp. 2786–2792, 2016.
[6] 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 *ASE*, pp. 574–584, November 2015.
[7] C. Quirk, R. Mooney, and M. Galley. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *ACL-IJCNLP*, pp. 878–888, 2015.