競技プログラミングにおける記号と制約を考慮したモデルによる着眼点の分類

泰山　幸大†　　山本　岳洋††

† 兵庫県立大学 社会情報科学部　〒 651–2197 兵庫県神戸市西区学園西町 8-2-1
†† 兵庫県立大学 大学院情報科学研究科　〒 651–2197 兵庫県神戸市西区学園西町 8-2-1
E-mail: †fa19p039@stsis.u-hyogo.ac.jp, ††t.yamamoto@sis.u-hyogo.ac.jp

あらまし　本論文では，競技プログラミングのコンテストに出題される問題に対して，着眼点と呼ぶラベルを予測する問題について取り組む．本研究で扱う着眼点とは，“シミュレーションが可能か考える” や “全探索が可能か考える” といった問題を解くための方針や “動的計画法” といった問題で用いるアルゴリズムを表したラベルである．問題から自動的に着眼点を予測しユーザに提示することができれば，ユーザが問題を解く支援になったり，ユーザが適切な着眼点を自分で考えながら問題を解く力の養成につながると考えられる．本研究では入力を問題文，記号および制約のデータとし，入力の問題に対して一つ以上の適切な着眼点を予測するマルチラベル分類問題として着眼点の予測問題を扱う．着眼点の分類モデルとして本研究では記号と制約の情報を特殊トークンとして追加した BERT モデルを提案する．AtCoder から独自でタグ付けを行った着眼点データと AtCoder Tags から収集したデータの二つのデータセットを用いて本研究で提案するモデルの性能を検証する．

キーワード　競技プログラミング，BERT，マルチラベル分類

1　はじめに

近年，世界各国でプログラミング教育が推し進められており [13]，日本でも小学校でのプログラミング教育が義務化された．IT 産業の拡大に伴い，IT 人材の需要は今後も増大していくと予想されている [7]．過去にプログラミング教育を支援するための研究 [6] [9] [8] も数多く行われており，そのなかでも Codeforces[1] という競技プログラミングサイトを用いたプログラミング学習法についての研究 [5] が行われており，先端技術に興味のある学生に大きな需要があることが示されている．

競技プログラミングとは，数学パズルやアルゴリズムの問題が与えられ，参加者がそれを解決するコードを提出するという形をとった競技である．日本最大級の競技プログラミングサイトである AtCoder[2] では毎週 5,000 人以上のユーザが参加するコンテストが開かれており，ウェブサイトで提供されている自動採点システムを通してプログラミングやアルゴリズムの学習をゲーム感覚で行うことができる．競技プログラミングには出題される問題の性質上，情報科学に関する問題が多く出題されるため，情報科学の基礎を身につけることができる．また，考えたアルゴリズムをプログラムとして実現することが要求されるため研究開発に必要な論理的思考能力やプログラムの実装力を鍛えることができる．

競技プログラミングには問題を解くための典型的なテクニックが数多く存在し，競技プログラミングを始めた初心者や数学が苦手な人にとってこのような問題を解くことは難しい．例えば，ある条件を満たす i と j の組み合わせの個数を求める必要がある問題の場合「i を固定したときの条件を満たす j の組み合わせを高速に求める」というテクニックを考えると問題の見通しがよくなることがある．また,「問題で与えられた数式を変形する」ことによって考えやすくなる問題も存在する．このように，競技プログラミングの問題を解くためには，使うアルゴリズムだけでなくテクニックや考え方を工夫しなければならず，これらは問題を解くときに注意すべきポイントである．上記のような問題を解くときに考えると良いアルゴリズムや考える方針のことを，本研究では着眼点と呼ぶ．

本研究では，問題文および制約と記号を入力とした着眼点を自動でタグ付けするためのモデルを提案する．使用するアルゴリズムやテクニックを考えるとき，制約によって使えるアルゴリズムが異なることもあるため，制約を考えることは重要である．また，記号は問題文の大まかな特徴を捉えるのに役に立つと考えられる．例えば与えられたクエリを処理していく形の問題であれば Q という記号が使用される傾向にあり，グラフに関する問題であれば u や v などの記号が出現する傾向がある．このような理由から，本研究では問題文だけでなく問題に割り当てられている制約や記号なども入力として加えたモデルの精度を検証する．

本研究では BERT 事前学習済みモデルを利用して 2 つの分類タスクに取り組む．1 つ目はユーザが投票によって問題を分類することができる AtCoder Tags [1] というサイトでタグ付けされているデータを用いた分類タスクであり，2 つ目は，AtCoder の着眼点を分類するタスクである．2 つのデータセットを用いて提案するモデルの精度を検証する．

AtCoder における問題に着眼点タグを付与することができれ

1：https://codeforces.com/
2：https://atcoder.jp

ば，ユーザが過去問演習を行う際のヒントを提示することができ，ユーザは問題をヒントを元に直接答えを見ることなく問題を考えることができる．またコンテスト中に使用することで問題を解くまでのスピードが要求される状況での問題を解くことの補助にもなる．

本論文の構成を次の通りである．第 2 節では競技プログラミングに関する用語と本研究で使用するデータセットの内容を解説した上で問題定義を行う．第 3 節では競技プログラミングや BERT に関する関連研究について述べる．第 4 節では本論文で使用するモデルについて述べる．第 5 節では実験結果について述べる．第 6 節では実験結果に対する考察を行う．第 7 節では本研究の結論について述べる．

2 競技プログラミングと問題定義

本章ではまず主な競技プログラミングサイトや，競技プログラミングの関連サイトについて解説をする．その後，問題定義を行う．

2.1 AtCoder

AtCoder は，日本最大級のプログラミングコンテストである．AtCoder で開催されているコンテストには AtCoder Beginner Contest(以下 ABC) や ABC よりも難易度の高い問題が多く出題される AtCoder Regular Contest(以下 ARC) などがある．ABC では一回のコンテストに A から Ex の難易度が低い順に 8 つの問題が出題され，参加者は問題を解くことで得た合計得点と提出速度によって順位付けされる．出題される問題 [3]の例を示す．

問題文

AtCoder 国には 1 から N の番号がついた N 個の都市と，1 から M の番号がついた M 個の道路があります．
道路 i を通ると都市 A_i から B_i へ移動することができます．都市 B_i から都市 A_i への通行はできません．
ピューマは，どこかの都市からスタートし，0 本以上の道路を使い移動して，どこかの都市をゴールとするような旅行の計画を立てようとしています．
スタート地点とゴール地点の都市の組として考えられるものは何通りありますか?

制約

- $2 \leqq N \leqq 2000$
- $0 \leqq M \leqq min(2000, N(N - 1))$
- $1 \leqq A_i, B_i \leqq N$
- $A_i \neq B_i$
- (A_i, B_i) は相異なる
- 入力に含まれる値は全て整数である

AtCoder で出題されている問題は，その問題内容が記述されている問題文と，入力が満たすべき条件を示している制約から構成されている．

3：https://atcoder.jp/contests/abc204/tasks/abc204_c

表 1　着眼点データセット 200 問に含まれる着眼点のラベル名と各ラベルに該当する問題の数.

ラベル名	問題数
動的計画法	52
全探索ができそうか考える	90
シミュレーションができそうか考える	82

2.2 着　眼　点

着眼点とは，競技プログラミングを解く上で必要な方針であり，正しい解法に使用されるアルゴリズムやデータ構造の種類に加え競技プログラミング特有のテクニックも着眼点に含まれている．着眼点は大きく分けて 3 つの種類に分けられる．まず，問題理解のための着眼点というものがある．例えば “全探索を考える” という着眼点がある．これは主に組み合わせの問題にタグ付けされている．問題で与えられたある条件を満たすような i と j を求める問題など，まず全ての組み合わせの通り数がどれくらいあるのかを把握するべきである．また，“シミュレーションが可能か考える” という着眼点がある．これはある操作が与えられる問題にタグ付けされている．例えば,「箱に i にボールを入れる」,「箱 i にあるボールを全て箱 j に移し変える」というクエリが与えられる問題などは，まず与えられた操作を実現するようなプログラムを考えるべきである．次に，問題考察のための着眼点である．問題考察とは問題で求めなければならないものを理解し，問題の性質や考えるべき条件などなどをもとに，AtCoder において出題されるテクニックが存在する．“桁ごとに考える” という着眼点が存在する．これは与えられる入力を整数などの数値型として保持するのではなく，文字列として保持し，各桁ごとに処理を行なうというテクニックである．桁ごとに処理を行わないと実行時間制限を超過してしまうように作られている問題は，明らかに与えられる数値が大きくなっている場合が多い．最後にアルゴリズムやデータ構造に関する着眼点である．これは問題を解くために用いるアルゴリズム名を着眼点としている．例えば “ダイクストラ法” や “動的計画法” などがある．競技プログラミングの問題を解く流れとして，問題で求めらているものが何かを把握し，問題特有の性質を考察テクニックから考え，使用するアルゴリズムを決定していくことが一般的である．競技プログラミングの問題を解くことに慣れていない人がこれらの着眼点をもとに考えを進めていくことで，コンテスト本番で素早く正確に問題考察を進める力を身につけていくことができると考えている．

2.3 本研究で用意するデータセット

2.3.1 着眼点データセット

本研究で定義した着眼点を分類するモデルの性能を検証するため，ABC の過去問のうち 200 問に対し着眼点を手作業でタグ付けを行い，データセットを作成した．本研究では 3 つの着眼点を扱う．着眼点の例と各着眼点が付与されている問題の数を表 1 に示す．これらの着眼点と問題のペアを用いて BERT モデルをファインチューニングする．

表 2　AtCoder Tags データセット内 778 問における，各大分類タグに該当する問題の数.

タグ名	問題数
Ad-Hoc	47
Data-Structure	76
Dynami-Programming	121
Graph	53
Greedy-Methods	47
Mathematics	143
Searching	148
String	62
Technique	81

2.3.2　AtCoder Tags データセット

本研究で提案するモデルの性能を検証するために，着眼点データセットとは別に AtCoder Tags という Web サイトをもとにデータセットを作成した．AtCoder Tags [1] とはユーザの投票によって AtCoder に出題されている問題のタグ付けを行っているウェブサイトであり，ユーザはカテゴリ別に分類された問題を検索することができる．タグには大分類と小分類という 2 つの 2 つの種類がある．大分類には “Dynamic-Programming” や “Searching” というタグが存在し，1 つの問題に対してただ 1 つ決定される．小分類には “深さ優先探索” や “最小全域木” などのより詳しい問題の性質や使用するアルゴリズムに関するタグが存在する．これらのタグは 1 つの問題に対して複数付与されている．AtCoder Tags [1] から 2023 年 1 月 8 日時点でのデータをスクレイピングによって収集した．大分類のタグの中から，該当する問題が 50 未満のタグと “Easy” のタグを取り除き，それぞれのタグに該当する問題を抽出した．大分類における各タグが付与されている問題の数を表 2 示す．抽出したタグが付与されている問題の数は全部で 778 問であり，これらの問題を用いて事前学習済み BERT モデルをファインチューニングする．

2.4　問 題 定 義

本研究では 1 つの問題に対して複数の着眼点を予測するマルチラベル分類タスクを解く．ここで，予測に使用する問題の 1 つを S_i とおく．予測する問題の集合を S とする．

$$S = \{S_1, S_2, ..., S_N\} \tag{1}$$

ここで N はデータセットにおける予測する問題の数を表す．そして，$S_i(1 \leqq i \leqq N)$ に対しての予測を P_i と置く．いま，予測したいラベルの数を M とするとき，P_i は以下の様に表すことができる．

$$P_i = \{p_{i,1}, p_{i,2}, ..., p_{i,M} \ | p_{i,j} \in \{0,1\} (1 \leqq j \leqq M)\} \tag{2}$$

ここで，$p_{i,j}$ は問題 S_i がラベル j に該当するかどうかを表す二値変数であり，該当する場合 1，そうでない場合 0 をとる．

また，マルチラベル分類問題だけでなく，マルチクラス分類問題とした際の提案手法の有効性を検証するため，AtCoder Tags におけるデータを用いて 1 つの問題に対して 1 つのタグを予測するマルチクラス分類タスクについても取り組む．いま，予測したいラベルの数を M とするとき，問題 S_i に対する予測 Q_i は以下の様に表すことができる．

$$Q_i = \{j | j \in \{1, ..., M\}\} \tag{3}$$

たとえば j 番目のラベルが該当すると予測した場合，$Q_i = j$ である．

3　関 連 研 究

本章では競技プログラミングや学習支援，BERT に関する関連研究について述べる．

3.1　競技プログラミング

競技プログラミングの問題を特定の種類に分類する研究が複数行われており，Ianc らの研究 [4] や Athavale らの研究 [2] では Codeforces や Topcoder といった海外の競技プログラミングサイトの問題文からタグを予測する研究が行われている．ここでのタグとは各問題のおおまかな種類を表すもので，“Brute Force” や “Graph”，“Dynamic Programming” などのタグ付けがなされている．それら問題文を用いて分類を行なっている．また，Codeforces において問題文の類似度を調査している研究も行われている [10]．AtCoder における問題のタグ付けを対象としている研究として，川渕らの研究がある [11]．川渕らの研究におけるタグとは，AtCoder Tags [1] で設定されているタグである．解答ソースコードから抽象構文木を作成し，Doc2Vec を用いてタグの分散表現を獲得し，同じように作成された問題の分散表現とタグの分散表現比較することでタグ付けを行なっている．

3.2　学 習 支 援

田口らは，過去の問題演習の履歴を利用し，個々の学習者の理解状況と学習意欲に合わせたプログラミング教育を行うための手法を提案している [8]．新開らは，プログラムを作成するために必要な力を問題解決のプロセスを通して身につけることを目的とした学習支援システムを開発している [9]．学習支援システムの中にヒントを提示するような機能を持たせるような研究も行われており，鈴木らは個人の学習進度に適応したヒントを学習者に提示する機能を提案している [14]．また，中村らの研究 [12] ではコード設計のヒントを提示するような機能を実装し，その有効性を明らかにした．競技プログラミングに関する学習支援の研究も過去に行われており，Mike らは競技プログラミングのウェブサイトである Codeforces をプログラミング学習に取り入た学習について調査を行い，先端技術に興味を示している学生らに需要があることを明らかにした [5].

4　問題文および制約と記号を入力とした分類モデル

本研究では，BERT を用いて着眼点をラベルとしたマルチラベル分類を行う．

表 3　トークンとして追加した記号の種類.

記号	記号の主な用途に関する説明
N	入力の数
M	グラフの辺の本数などの N とは異なる入力の数
K	操作の回数
A	配列の要素の値
Q	クエリの個数
W	グリッドの横幅
H	グリッドの縦幅
S	文字列の長さ
T	時間や文字列の長さ
x	座標の位置
y	座標の位置
u	グラフの頂点の値
v	グラフの頂点の値

4.1　BERT

本研究では，自然言語処理モデルである BERT [3] を用いて分類タスクを行う．事前学習済みモデルは東北大学の乾研究室が提供している日本語 BERT モデルを使用する．

4.2　制約・記号のトークンの追加

本研究では BERT モデルに対し記号トークンおよび制約トークンを新しい単語として追加し，AtCoder の問題文において非常に重要な情報を持っている記号と制約のデータを入力に明示的に追加することで分類精度が向上するのかを調査する．記号とは問題文や制約中に登場する記号のことであり，N や S などのさまざまな種類がある．記号は種類によって同じような用途で使用される傾向があり，本研究では問題文中によく使われている N，A，Q，W，H，s，t，u，v の 9 種類の記号を新しい特殊トークンとしてトークナイザに追加する．追加した記号の種類と問題での主な用途を表 3 に示す．また，問題に登場する制約に関しては 8 種類の分類を作成し，各記号に対して与えられている制約における数値の最大値を抽出する．そうして得られた数値が表 4 における定義域のどの範囲に属しているかを調べることによって制約の種類を決定する．問題に与えられる制約によって考えるべきアルゴリズムや考察テクニックが変わる．制約が小さい場合は愚直なアルゴリズムを実装すれば解ける問題である場合や，制約が小さいときに適用できるアルゴリズムを使う問題である場合がある．逆に，与えられた制約が明らかに大きい場合は効率的なアルゴリズムを実装しなければならないような問題もある．このように，着眼点を分類する上で制約は重要な情報であるため，各制約に対しても新しい特殊トークンとしてトークナイザに追加する．追加したそれぞれの制約に対しての定義域を表 4 に示す．上記の新しく追加したトークンについて，初期化された重みを BERT モデルの embedding 層に追加した．

4.3　入力データの前処理

ここでは本研究で提案する記号制約付き BERT モデルや比較実験で用いるモデルに入力するためのデータの加工方法について説明する．モデルに入力する情報として，AtCoder の問題文のテキスト，制約，記号があり，それらの加工の流れの詳細について述べる．

AtCoder の問題文中には $A = (A_1, A_2, ..., A_N)$ のような数式が記述されていることが多く，そのような数式をそのまま入力として利用すると，入力トークン数が非常に大きくなり一度に入力できるトークン数の上限値である 512 を超えてしまうことがある．そのため，トークン数の増加を防ぐために問題文のテキストから数式や記号を削除する．AtCoder の問題ページから本文のテキストを抽出するのに，Python ライブラリの BeautifulSoup を用いた．HTML データでは記号や数式は var タグで囲まれており，var タグ内のテキストを削除することで本文から記号や数式を除去できる．

次に新しいトークンとして追加した記号に対応する制約を AtCoder の問題ページから抽出し，制約の情報をもとにトークンを作成する．例えば，$1 \leqq N \leqq 10^5$ のような制約が与えられた場合，その記号のとりうる数値の最大値である 10^5 を抽出し，その数値が表 4 で示したもののどの制約に該当するかを調べる．その後，数式と記号を除去したテキストデータの末尾に `[N][const6]` のように記号トークンと制約トークンを並べて追加する．BERT への入力データと本研究で提案する分類モデルの概要を図 1 に示す．

表 4　トークンとして追加した制約の種類.

トークン	変数の定義域
`[const1]`	$0 \leqq x < 10$
`[const2]`	$10 \leqq x < 20$
`[const3]`	$20 \leqq x < 10^3$
`[const4]`	$10^3 \leqq x < 10^4$
`[const5]`	$10^4 \leqq x < 10^5$
`[const6]`	$10^5 \leqq x < 10^6$
`[const7]`	$10^6 \leqq x$

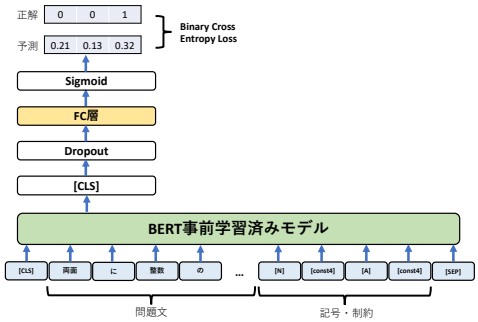

図 1　BERT を用いた着眼点の分類モデル.

4.4　損 失 関 数

本研究ではマルチラベル問題を解くため，損失関数としてバイナリ交差エントロピー損失関数を用いる．いま，ある問題 S_i（$1 \leqq i \leqq N$）に対するラベル m（$1 \leqq m \leqq M$）に対してモデルが正例であると予測出力した確率を $p_{i,m}$ とする．また，S_i のラベル m に対する正解ラベルを $y_{i,m}$ とする．$y_{i,m}$ は問題 S_i がラベル m を持つとき 1，そうでないとき 0 となる 2 値変数である．このとき，S_i に対するバイナリ交差エントロピー損失

関数 L は以下の式で表される．

$$L = \sum_{m=1}^{M} \left(-w_m y_{i,m} \log p_{i,m} - (1 - y_{i,m}) \log(1 - p_{i,m})\right) \tag{4}$$

ここで，w_m は正例に対する重みである．本研究では各ラベルの正例に比べて負例が多い不均衡データとなっている．w_m に対して高い重みを与えることで，モデルが負例と予測しやすくなることを抑制できる．

マルチクラス分類においては，損失関数として交差エントロピー損失関数を用いる．マルチラベル分類問題と異なり，マルチクラス分類問題では，正解となるラベルは 1 つのみである．いま，問題 S_i がラベル k ($1 \leqq k \leqq M$) に属すとき，正解ラベル $y_{i,m}$ は $m = k$ のとき 1，そうでないとき 0 となる．このとき，交差エントロピー損失関数 L は以下の式で表される．

$$L = -\sum_{m=1}^{M} w_m y_{i,m} \log p_{i,m} \tag{5}$$

これらの損失関数を用いて，BERT のパラメータを更新する．

5 評価実験

本節では提案手法を有効性を検証するために行った実験とその結果について述べる．まず，評価指標について説明する．その後，AtCoder Tags データセットに対する予測と着眼点データセットに対する予測の 2 つの実験におけるそれぞれの実験条件について述べる．

5.1 実験に用いた手法

5.1.1 BERT(記号制約あり)

4.2 節で新しいトークンを認識できるようにした BERT に対し，記号と制約の情報を追加したテキストデータを入力としてファインチューニングを行う．ファインチューニングに用いたハイパーパラメータの詳細は以下の通りである．

- **maxlength：512**
- **batch size：16**
- **学習率：** 2×10^{-5}
- **Dropout：0.1**
- **early stopping：3 patience**
- **Optimizer: Adam**

5.1.2 BERT(記号制約なし)

乾研究室が公開している事前学習済み BERT に対し，記号や数式を取り除いたテキストデータを入力としてファインチューニングを行う．ファインチューニングに用いたハイパーパラメータは，BERT(記号制約あり) と同様である．

5.1.3 ロジスティック回帰

ロジスティック回帰はデータが各クラスに所属する確率を求めることで分類を行うアルゴリズムである．ABC001 から ABC286 までの 1616 問の問題を文書集合として作成した TFIDF ベクトルを入力とする．記号と制約の情報を追加する場合，その問題に含まれている記号と制約をそれぞれ特徴量とし，入力とする問題に含まれる数を要素とするベクトルを TF-IDF ベクトルに結合する．本研究では scikitlearn の sklearn.linear_model.LogisticRegression を用いて分類を行う．

5.1.4 SVM(サポートベクターマシン)

分類や回帰に用いられる機械学習アルゴリズムであり，データを分割する最適な超平面を求めることで分類を行う．本研究では scikitlearn の sklearn.svm.SVC を用いて分類を行う．ロジスティック回帰の場合と同様に，TFIDF ベクトルを入力とし，記号と制約の情報を追加する場合はあらかじめ作成したベクトルを結合したものを入力とする．

5.1.5 Random Forest

機械学習アルゴリズムの一種であり，複数の決定木を組み合わせて作成されるアンサンブル学習の一種である．分類や回帰に使用されるモデルである．本研究では scikitlearn の sklearn.ensemble.RandomForestClassifier を用いて分類を行う．ロジスティック回帰の場合と同様に，TFIDF ベクトルを入力とし，記号と制約の情報を追加する場合はあらかじめ作成したベクトルを結合したものを入力とする．

5.2 評価指標

マルチラベル分類モデルの精度を比較するための評価指標として，Macro-Precision，Macro-Recall，Macro-F_1 を用いる．TP，FP，TN，FN を以下のように定義する．

- TP：正例と予測されたもののうち実際に正例であったものの数
- FP：正例と予測されたもののうち実際は負例であったものの数
- TN：負例と予測されたもののうち実際に負例であったものの数
- FN：負例と予測されたもののうち実際は正例であったものの数

予測するラベルの数を N とすると，Macro-F_1 スコアは以下のように定義される．

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}$$

$$\text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}$$

$$\text{Macro-Recall} = \frac{1}{N}\sum_{k=1}^{N} \text{Recall}_\text{k}$$

$$\text{Macro-Precision} = \frac{1}{N}\sum_{k=1}^{N} \text{Precision}_\text{k}$$

$$\text{Macro-}F_1 = \frac{1}{N}\sum_{k=1}^{N} \frac{2}{\frac{1}{\text{Precision}_k} + \frac{1}{\text{Recall}_k}}$$

表 5　AtCoder Tags データの大分類タスクにおける各ラベルごとのクラスウェイトの値.

ラベル名	重み
Ad-Hoc	1.189
Data-Structure	1.133
Dynamic-Programming	0.712
Graph	1.646
Greedy-Methods	1.819
Mathematics	0.606
Searching	0.586
String	1.410
Technique	1.063

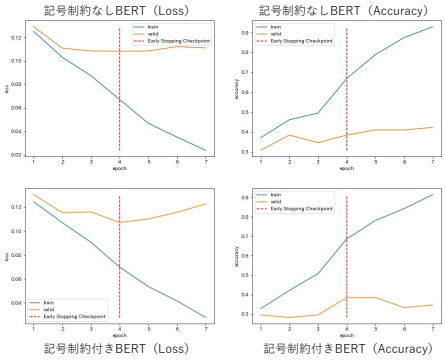

図 2　AtCoder Tags データの大分類タスクにおける BERT の学習曲線.

5.3　AtCoder Tags データの大分類における分類タスク

AtCoder Tags データから大分類とそれぞれのタグに該当する問題によってファインチューニングを行った．また，分類するクラスごとに正例の数の偏りがあるため，訓練データ中の各ラベルの分布に基づき，numpy の numpy.compute_class_weight を用いてクラスウェイトを設定した．パラメータの詳細を表 5 に示す．問題データとラベルのペアデータは 778 件あり，そのうち 8 割を学習データ，1 割を検証データ，1 割をテストデータとした．また，Early Stopping を適用し，記号と制約を考慮していない BERT では 2epoch 目，記号制約を考慮した BERT では 3epoch 目で早期終了した．学習時の学習曲線を図 2 に示す．また，各評価指標の値を表 6 に示す．また，BERT(記号制約あり) による予測を行なった結果の混同行列を図 3 に，BERT(記号制約なし) による予測を行なった結果の混同行列を図 4 に示す．

学習を行った結果，記号と制約を新しいトークンとして追加した BERT モデルの方が追加しなかった BERT モデルよりも Macro-F_1 スコアが高くなった．また Random Forest とロジスティック回帰での分類精度は向上したものの，SVM の分類精度は向上しなかった．これらの結果から，大まかな問題の特徴を捉える場合記号と制約は重要な特徴量となりうることがわかった．

5.4　着眼点データにおける分類タスク

着眼点タグとそれぞれのタグに該当する問題によってファインチューニングを行った．

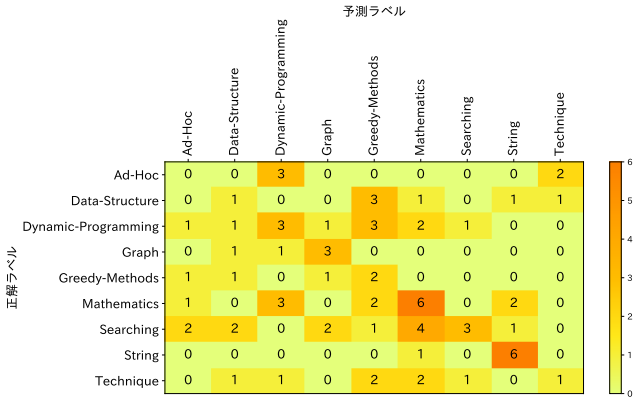

図 3　AtCoder Tags データの大分類タスクにおける BERT(記号制約あり) での検証結果の混同行列.

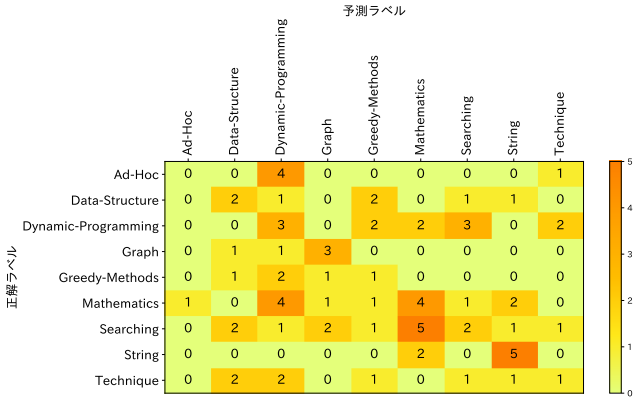

図 4　AtCoder Tags データの大分類タスクにおける BERT(記号制約なし) での検証結果の混同行列.

問題データとタグのペアデータは 200 件あり，そのうち 8 割を学習データ，1 割を検証データ，1 割をテストデータとした．また，Early Stopping を適用し，記号と制約を考慮していない BERT では 4epoch 目，記号制約を考慮した BERT では 1epoch 目で早期終了した．学習時の学習曲線を図 5 に示す．また，各評価指標の値を表 7 に示す．

学習を行った結果，記号と制約を新しいトークンとして追加した BERT モデルの方が追加しなかった BERT モデルよりもわずかに Macro-F_1 が低くなった．一方で，Random Forest と SVM に関しては Macro-F_1 スコアが大きく下がった．記号と制約を考慮した BERT モデルを用いたときの着眼点ごとの各評価指標の値を示す表 8 を見てみると，“全探索が可能か考える” と “動的計画法” に関してうまく分類できていないため，記号と制約についての影響を上手く検証できていない可能性がある．

6　考　　察

本研究では AtCoder Tags の大分類におけるマルチクラス分類と着眼点のマルチラベル分類の 2 つのタスクに取り組んだ．学習を行った結果，記号と制約を新しいトークンとして追加した BERT モデルの方が追加しなかった BERT モデルよりも

表 6　AtCoder Tags データセットに対するモデルごとの評価指標の値.

	記号制約あり				記号制約なし		
	Macro-Precision	Macro-Recall	Macro-F_1		Macro-Precision	Macro-Recall	Macro-F_1
BERT	0.314	0.334	0.300	BERT	0.248	0.288	0.259
Random Forest	0.284	0.282	0.277	Random Forest	0.280	0.253	0.255
SVM	0.174	0.239	0.187	SVM	0.174	0.239	0.187
ロジスティック回帰	0.318	0.272	0.277	ロジスティック回帰	0.270	0.257	0.255

表 7　着眼点データセットに対するモデルごとの評価指標の値.

	記号制約あり				記号制約なし		
	Macro-Precision	Macro-Recall	Macro-F_1		Macro-Precision	Macro-Recall	Macro-F_1
BERT	0.750	0.467	0.531	BERT	0.611	0.575	0.571
Random Forest	0.269	0.350	0.259	Random Forest	0.537	0.517	0.492
SVM	0.258	0.283	0.236	SVM	0.424	0.367	0.347
ロジスティック回帰	0.508	0.625	0.508	ロジスティック回帰	0.439	0.437	0.375

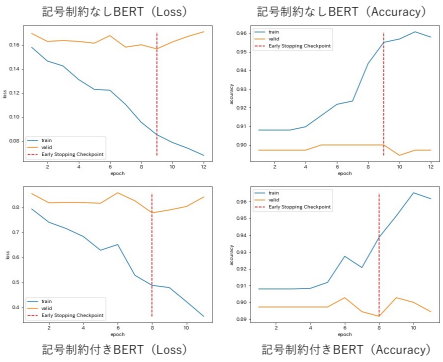

図 5　着眼点データの分類タスクにおける学習曲線.

表 8　BERT(記号制約あり) での予測結果における各着眼点タグに対する評価指標の値.

ラベル名	Precision	Recall	F_1
全探索が可能か考える	0.500	0.400	0.444
シミュレーションが可能か考える	0.750	0.750	0.750
動的計画法	1.000	0.250	0.400

Macro-F_1 スコアが高くなった．タグごとの分類結果を見てみると，特に “String” のタグの分類精度が他のタグと比較して非常によく，BERT(記号制約あり) では 7 個の正例のうち 6 個を正例であると分類できている．これは “String” とタグ付けされている問題文に「文字列」という単語が含まれていることが多いためであると考えられる．正解タグが “String” である ABC135-F の問題について考えると，BERT(記号制約なし) のときは “Mathematics” であると分類されていたが，BERT(記号制約あり) では “String” タグと正解タグを出力できていた．問題文が短いことや「非負整数」という単語が出現しているため “Mathematics” と分類されていたが，S という記号があることによって正解タグを出力できた可能性があると考えられる．しかし，今回の実験ではテストデータの数が十分ではないため，記号制約の有無によって予測精度が向上したかどうかについては引き続き検証が必要である．

着眼点データにおける分類タスクでは，記号と制約を用いたデータで分類を行なった場合の分類精度がそうでない場合よりも精度が低下した．今回分類に用いたラベルである “全探索が可能か考える”，“シミュレーションが可能か考える” の 2 つのラベルはグラフの問題や文字列の問題などおおまかな問題の分類ではなく，組み合わせの問題であるか，操作が与えられるような問題かどうか，という観点でラベルを付与したため，記号と制約を用いたとしても分類精度が向上する可能性は低いと考えられる．

BERT に新しいトークンを追加した場合，基本的に大量のデータで追加学習を行う必要があり，今回の訓練データの数は 200 と比較的少ないデータ数であった．そのため，新しく追加したトークンに当たるパラメータの更新が上手く行われなかったことが精度の低下の原因であると考えられる．

今後精度の検証を行うには，より多くのデータセットで実験する必要がある．今回対象とした AtCoder Beginner Contest では今回扱った着眼点を必要としないような容易な問題も多い．今後は，AtCoder Beginner Contest だけでなく，yukicoder [4] や Aizu Online Judge [5]のようなさまざまな競技プログラミングコンテストの問題を収集し実験する必要がある．

7　結　　論

本研究では AtCoder の問題に出現する記号と制約の情報を明示的に入力に組み込んだ BERT モデルを提案し，AtCoder Tags データセットと着眼点データセットの 2 つのデータセットにおいて効果を検証した．AtCoder Tags データセットでの実験の結果，記号と制約の情報を入力に加えたほうが，わずかに分類精度が高くなることが分かった．しかし，今回の実験ではテストデータの数が十分ではないため，記号制約の有無によって予測精度が向上したかどうかについては引き続き検証が必要である．着眼点データセットでの実験の結果，記号と制約の情報を考慮したモデルの分類精度が低くなった．BERT に新

4：https://yukicoder.me/
5：https://judge.u-aizu.ac.jp/onlinejudge/

しいトークンを追加した場合，基本的に大量のデータで追加学習を行う必要があり，今回の訓練データの数は 200 と比較的少ないデータ数であった．そのため，新しく追加したトークンに当たるパラメータの更新が上手く行われなかったことが精度の低下の原因であると考えられる．今後精度の検証を行うには，より多くのデータセットで実験する必要がある．今回対象とした AtCoder Beginner Contest では今回扱った着眼点を必要としないような容易な問題も多い．今後は，AtCoder Beginner Contest だけでなく，yukicoder や Aizu Online Judge のようなさまざまな競技プログラミングコンテストの問題を収集し実験する必要がある．

謝辞　本研究は JSPS 科学研究費助成事業 JP21H03904, JP22H03905，による助成を受けたものです．ここに記して謝意を表します．

文　　献

[1] AtCoder Tags. https://atcoder-tags.herokuapp.com/. 2022 年 2 月 1 日閲覧.

[2] Vinayak Athavale, Aayush Naik, Rajas Vanjape, and Manish Shrivastava. Predicting algorithm classes for programming word problems. In *Proceedings of the 5th Workshop on Noisy User-generated Text*, pp. 84–93, 2019.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

[4] Bianca Iancu, Gabriele Mazzola, Kyriakos Psarakis, and Panagiotis Soilis. Multi-label classification for automatic tag prediction in the context of programming challenges. *CoRR*, Vol. abs/1911.12224, , 2019.

[5] Mike Mirzayanov, Oksana Pavlova, Pavel Mavrin, Roman A. Melnikov, A. S. Plotnikov, Vladimir Parfenov, and Andrew Stankevich. Codeforces as an educational platform for learning programming in digitalization. In *Olympiads in Informatics*, Vol. 14, pp. 133–142, 2020.

[6] Kissinger Sunday, Patrick Ocheja, Sadiq Hussain, Solomon Oyelere, Balogun Samson, and Friday Agbo. Analyzing student performance in programming education using classification techniques. *International Journal of Emerging Technologies in Learning (iJET)*, Vol. 15, No. 2, pp. 127–144, 2020.

[7] 経済産業省. 第 1 回「第 4 次産業革命スキル習得講座認定制度（仮称）」に関する検討会. `https://www.meti.go.jp/shingikai/economy/daiyoji_sangyo_skill/pdf/001_s03_00.pdf`, 2017. 2022 年 2 月 1 日閲覧.

[8] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, 島川博光. 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援. 情報処理学会論文誌, Vol. 48, No. 2, pp. 958–968, 2007.

[9] 新開純子, 炭谷真也. プロセスを重視したプログラミング教育支援システムの開発. 日本教育工学会論文誌, Vol. 31, No. Suppl., pp. 45–48, 2008.

[10] 新濱遼大, 槇原絵里奈, 小野景子, 幾島直哉, 山川蒼平. オンラインジャッジシステムにおける問題文の類似度調査. 研究報告ソフトウェア工学（SE）, Vol. 2021-SE-209, No. 9, pp. 1–7, 2021.

[11] 川渕皓太, 松下誠, 吉田則裕, 井上克朗. 解答ソースコードを用いたプログラミング演習問題に対するタグ付け手法の提案. 研究報告ソフトウェア工学（SE）, Vol. 2022-SE-211, No. 14, pp. 1–8, 2022.

[12] 中村拓哉, 舩曳信生, 中西透, 天野憲樹. Java プログラミング学習支援システムのコード作成問題における javadoc を用いたヒント機能. Vol. 113, No. 377, pp. 115–120, 2014.

[13] 文部科学省. 諸外国におけるプログラミング教育に関する調査研究. `https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/__icsFiles/afieldfile/2018/08/10/programming_syogaikoku_houkokusyo.pdf`, 2015. 2022 年 10 月 7 日閲覧.

[14] 鈴木孝幸, 納富一宏. 初学者向けプログラミング演習支援システムにおける学習進度適応型ヒント提示機能の実装. 情報科学技術フォーラム講演論文集, pp. 313–314, 2019.