修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工研究科　　情報・ネットワーク専攻　博士前期課程		
氏　　名	門脇瑞穂	学籍番号	2431040
論 文 題 目	問題難易度予測のための LLM による問題-スキル間構造を組み込んだ Graph Neural Network		

要　旨

アダプティブラーニングの普及に伴い，学習者にとって最適かつ公平な教育の基盤として，「問題難易度」の正確な推定が不可欠となっている．従来，難易度は事前テストや専門家の判断に依存して決定されてきたが，これらの手法は多大なコストと時間を要する上に主観的であるという課題があった．特に，論理構造やアルゴリズムが複雑なプログラミング問題において，高精度な自動推定は困難とされてきた． また既存の自動推定手法は，問題文やコードの表層的な特徴に依存しており，学習者が問題を解く際に行う試行錯誤や解法選択といった潜在的な思考プロセスを十分に考慮できていない．

そこで本研究では，大規模言語モデル（LLM）とグラフニューラルネットワーク（GNN）を融合させ，解答過程の潜在的な情報を活用する新たな難易度予測手法を提案する． 具体的には，まず LLM にコード実行・検証が可能な環境を与え，自律的に問題を解かせることで，「解法説明」，「必要なスキル」，「難所」といった潜在的な難易度要因を抽出する．これらの抽出には，LLM が正解に至った推論過程のみを使用することで，ハルシネーションの影響を抑制し，特徴量の信頼性を担保する． 次に，これらの情報を基に，問題とスキルをノードとする 2 部グラフを構築する．このグラフ上で GNN によるメッセージパッシングを行うことで，問題とスキルの明示的な依存関係を学習し，共通スキルを介した問題間の情報伝播を実現する． さらに，得られた問題表現間の関係性から，ベイズ推定を用いて問題間のグラフ構造を適応的に学習する．これにより，問題間の潜在的な依存関係を補完し，予測精度のさらなる向上を図る．

評価実験の結果，本手法は LLM のパラメータを凍結することで，学習コストを大幅に抑制した構成を実現しつつ，Full Fine-tuning を行う既存手法を上回る予測精度を達成した． この結果は，グラフ構造による制約とスキル情報の活用により，データ効率の良い学習が実現されていることを示唆している． 以上より，提案手法は計算資源や利用可能なデータが限られる現実的な運用環境においても，有効な選択肢となり得ることが示された．

令和７年度　修士論文

問題難易度予測のためのLLMによる問題-スキル間構造を組み込んだGraph Neural Network

電気通信大学大学院情報理工学研究科

情報・ネットワーク工学専攻 情報数理工学プログラム

2431040

門脇瑞穂

主任指導教員:　植野 真臣

副指導教員:　宇都 雅輝

2026年2月26日

目 次

1　まえがき

近年，教育現場におけるデータサイエンスの活用は急速に進展しており，特に学習者個人の知識レベルに合わせて最適な問題を提示するアダプティブラーニングが注目されている．アダプティブラーニングは学習者のスキルを客観的かつ効果的に測定する技術に支えられている．この教育測定の質と妥当性は，テストを構成する個々の問題の質に大きく依存しており，中でも「難易度」の概念は，テスト全体の質を決定づける最も重要な要素の一つである [1]．問題の難易度情報を用いることで，知識習熟度の異なる学生に適切な難易度の演習を提案したり [2]，難易度の異なる問題を組み合わせてテスト内容を自動的に構成したり [3]，そのような構成問題の異なるテストの公平性の保証を可能にしたりすることができる [4]

問題の難易度は，学習者の実際の解答反応データに基づき，項目反応理論（IRT: Item Response Theory）や古典的テスト理論（CTT: Classical Test Theory）といった心理測定的モデルを用いて事後的に推定される [5, 6]．しかし，多くの教育機関や資格試験団体では，試験の公平性・比較可能性を担保するためにも，毎年新規問題を作成し続けなければならない [1, 7]．ここで，新規の問題に対し適切な難易度が設定されていないと学習効果や公平性が損なわれしまう危険性があるが，問題の難易度はテストを実施する前に直接観察することはできない．この課題に対処するため，従来は事前に仮のテストを行ったり，専門家の勘と経験に基づく手動の難易度ラベル付けが依頼されてきた [8, 9]．しかし，これらの手法は時間的・金銭的コストを要するため大規模な運用が困難であると頻繁に批判されている [10–12]．また，専門家の判断のようなヒューリスティックなアプローチは，難易度推定の手段として一貫性を欠く場合があることが実証研究によって指摘されている [13, 14]．加えて，近年普及が著しいコンピュータ適応型テスト（CAT: Computerized Adaptive Testing）においては，事前の難易度推定精度はテストの測定精度に直結する重要な要素である．したがって，機械学習や自然言語処理を応用した，客観的かつ高精度な自動難易度予測技術の構築は喫緊の課題とされている [12].

そこでこれまで，問題文のテキストに内在する言語的特徴から難易度を予測する研究が進められてきた [15]．これらの難易度予測研究の対象は，自然言語の文章読解中心のものが殆どである．一方で，近年コンピュータサイエンス分野の重要性が急速に高まる中，プログラミング教育に関する研究は，語学学習ドメインに次いで活発な研究分野となってい

る [12]. プログラミング学習は，単なる文法学習・コーディングスキル習熟にとどまらず，離散数学，グラフ理論，計算幾何学といった高度なアルゴリズムの学習に通づる．なおかつ実用的な身の回りの課題に対し，時間複雑度や空間複雑度を考慮し効率的な解法を導き出すような，論理的思考力を鍛える側面を持つため教育分野としても重要性が高い．しかし，プログラミング問題の難易度予測に特化した手法は，未だ非常に少ないという課題がある．

プログラミング問題においては，問題文の言語的特徴に加え，解答コードの構造やアルゴリズムの複雑さが難易度に寄与するため，特有の分析が必要とされる [16]. 既存研究は大きく三つに別れる．一つは，問題文と難易度のセットで事前学習済みの小規模言語モデルを Full Fine-tuning して予測を行う研究 [17, 18] である．二つ目は，問題文と解答コードに対して個別に専用の特徴量抽出を行ってから連結する手法だ. 特に解答コード情報に注目し, コード情報を可能な限り抽出するために抽象構文木 (AST: Abstract Syntax Tree) と畳み込みニューラルネットワーク (CNN: Convolutional Neural Network) を用いる [16]. 三つ目は大規模言語モデル (LLM: Large Language Model) を活用する手法である．LLM に指定された範囲の数値のみを出力とするよう指示し Fine-tuning する手法 [19] や，LLM を個別の学習者として振る舞うよう強化学習し，LLM に問題を解かせた時の正誤反応を用いる手法がある [20]．しかし，これらの先行研究は数億以上のパラメータを持つ言語モデルを Fine-tuning するため学習コストが高い．またプログラミング言語の種類が異なるとうまく学習できないことがわかっており同一言語の十分な学習データは収集が困難なため予測精度が低くなってしまうという課題がある．

また，先行研究は主に問題文と解答コードのみから特徴量を抽出しており，学習者が解答コードを作成するまでに行う「中間作業」——すなわち，問題の理解，試行錯誤，解法の選択といった思考プロセス——を十分に捉えられていない．プログラミング問題においては，暗記してきた知識を瞬時に回答するよりも，適切な解法を選択し適用するまでのプロセスが難易度に強く寄与する．しかし，多くの教育現場において，こうした学習者の内部的な処理過程をデータとして取得・蓄積することは困難であった．

そこで本研究では，この中間作業を反映する情報として「解法の説明」,「問題解答に必要なスキル」,「解答過程における難所」に着目する．提案手法では，LLM に実際に問題を解答させることで，これらの潜在的な難易度要因を顕在化させ，特徴量として活用する．さらに，問題とスキルの関係性をグラフ構造としてモデル化し，グラフニューラルネットワーク（GNN）を導入することで予測精度の向上を図る． また，テキスト及びコードの表現獲得には高度な知識および，推論能力を持つ LLM を基盤とした埋め込みモデルを，パラメータを固定して使用する．これにより，言語モデル自体の Fine-tuning を回避し，学習対象パラメータを大幅に削減した低コストなモデル構築を実現する．

2　先行研究

2.1　難易度予測タスクの特徴

難易度予測は，類似タスクである解答時間の予測とは異なる特性を持つことが報告されている．例えば，解答時間の予測においては問題タイプや選択肢数といったメタデータが支配的な要因となるのに対し，難易度予測ではそれらの寄与は限定的であり，テキスト長や意味内容といった言語的特徴がより重視される傾向にある [21]．また，一般に難易度予測は解答時間の予測よりも困難なタスクとされており，どのような特徴量を用いても大幅な精度向上は達成しにくく，ベースラインに近い性能に留まる傾向があることが指摘されている [21]．

2.2　問題文の言語的特徴量を活用する研究

問題文を用いた難易度予測に関する初期の研究では，表層的な指標を用いた自動抽出，あるいは人手による特徴量抽出に依存していた．例えば，Aryadoust らは，統語的特徴の人手による抽出を行っている [22]．一方，2017 年以降のニューラルネットワークに基づくアプローチでは，モデルの特徴量抽出器として単語埋め込み等のニューラル言語モデルが導入され始めた．例えば，Zhou と Tao (2020) はプログラミング問題の難易度予測を Encoder 型の言語モデル (BERT) の Full Fine-tuning によって行った [17]．このような言語モデルは，Vaswani らによって提案された Transformer アーキテクチャ[23] を基盤としている．Transformer は，入力系列内の各要素間の関係性を重み付けする注意機構を備えており，系列内の長距離の依存関係を効率的に捉えることが可能である．特に BERT 等のモデルは，この Transformer の Encoder ブロックを多層に積み重ねた構造を持ち，双方向からの文脈理解を実現している．BERT 等の言語モデルを用いた Full Fine-tuning では，入力テキスト x に対するモデルの出力 $f(x;\theta)$ と，正解ラベル y との間の損失関数 $\mathcal{L}$ を最小化するように，モデルの全パラメータ θ が更新される．

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(f(x_i;\theta), y_i) \tag{2.1}$$

ここで，N は学習データのサンプル数である．さらに近年の手法では，BigBird や Longformer といった，より長いコンテキストを扱える大規模モデルが難易度予測研究に適用されている [18]．これらは Transformer の注意機構を疎行列化するなどの工夫により，プログラミング問題のように長文になりがちなテキストに対しても効率的な学習を可能にしている．

2.3 コード特有の特徴を活用する研究

プログラミング言語は，自然言語と比較して厳密な構文規則や論理構造といった豊富な情報を含有している．その構造の特殊性ゆえに，コード特有の性質に着目した分析が行われてきた．初期の研究では，Grivokostopoulou らによる，探索アルゴリズム演習における難しさの原因調査が行われた [24]．この研究では演習固有の特徴，例えばノード数，ノードが持つ平均的な子ノードの数，ツリーの深さ，解の長さなどが難易度に影響することが明らかにされた．

更に最近の研究では，SQL 問題の難易度予測モデル（SQL-DP）が提案されている [16]．SQL-DP では，解答 SQL コードを AST に変換し，Tree-Based Convolutional Neural Network（TBCNN）によって木構造から固定長のコード構造特徴量を抽出し回帰モデルを用いて難易度を推定する．

TBCNN による AST 上の畳み込み　AST 上の各ノード i を埋め込み $\mathbf{x}_i \in \mathbb{R}^{N_f}$（$N_f$ はノード埋め込み次元）で表す．あるスライディングウィンドウ内のノード数を n_w とし，そのノード埋め込みを $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n_w}\}$ とすると，畳み込み出力 $\mathbf{y} \in \mathbb{R}^{N_c}$（$N_c$ は feature detector 数＝畳み込み特徴次元）は次式で与えられる：

$$\mathbf{y} = \tanh\left(\sum_{u=1}^{n_w} \mathbf{W}_{\mathrm{conv},u} \mathbf{x}_u + \mathbf{b}_{\mathrm{conv}} \right). \tag{2.2}$$

ここで $\mathbf{W}_{\mathrm{conv},i} \in \mathbb{R}^{N_c \times N_f}$ はウィンドウ内位置（役割）i に対応する重み，$\mathbf{b}_{\mathrm{conv}} \in \mathbb{R}^{N_c}$ はバイアスである．

Continuous Binary Tree(CBT) による重みの定義　子ノード数が可変な AST に対応するため，$\mathbf{W}_{\mathrm{conv},i}$ は 3 つの基底行列の線形結合として定義される：

$$\mathbf{W}_{\mathrm{conv},u} = \eta_u^t \mathbf{W}_{\mathrm{conv}}^t + \eta_u^l \mathbf{W}_{\mathrm{conv}}^l + \eta_u^r \mathbf{W}_{\mathrm{conv}}^r. \tag{2.3}$$

ここで $\mathbf{W}_{\mathrm{conv}}^t, \mathbf{W}_{\mathrm{conv}}^l, \mathbf{W}_{\mathrm{conv}}^r \in \mathbb{R}^{N_c \times N_f}$ はそれぞれ top/left/right の基底重みである．係数 $\eta_i^t, \eta_i^l, \eta_i^r$ はウィンドウ深さから計算され，ウィンドウ最大深さを $d_{\max}$，ノード i の深さを

d_i として

$$\eta_u^t = \frac{d_{\max} - \mathrm{dep}_u}{d_{\max}}. \tag{2.4}$$

また，同一深さ内で左から数えたインデックスを i（$1 \le i \le n_w$）とし，葉（leaf）か否か（non-leaf）で

$$\eta_u^l = \begin{cases} \frac{p_u - 1}{n_{\mathrm{dep}_u} - 1}\,(1 - \eta_u^t) & \text{(non-leaf)}, \\ 0.5\,(1 - \eta_u^t) & \text{(leaf)}, \end{cases} \tag{2.5}$$

残りを

$$\eta_u^r = (1 - \eta_u^t) - \eta_u^l. \tag{2.6}$$

とすることで，可変分岐の部分木に対しても一貫した重み付けで畳み込みが可能になる．

動的プーリングと回帰損失　AST 全体で得られる多数の $\mathbf{y}$ を dynamic pooling で集約し，固定長のコード構造特徴 $\mathbf{h}_{\mathrm{code}} \in \mathbb{R}^{N_c}$ を得る．問題 p_j の真の難易度を d_j，予測値を $\hat{d}_j$，データ数を N とすると，回帰学習は平均二乗誤差

$$\mathcal{L} = \frac{1}{N} \sum_{j=1}^{N} \left(\hat{d}_j - d_j \right)^2 \tag{2.7}$$

を最小化することで行われる．Xu らの実験設定において，SQL-DP は最も関連性の高いフレームワークと比較して，難易度予測の RMSE を最大 7.23% 低減した [16].

2.4　LLM を活用する研究

Yoshida らは，プログラミング問題の難易度推定に対し，GPT-3.5 Turbo を OpenAI API で Fine-tuning する手法を提案した [19]．この研究では，元の 28 段階（800–3500）の難易度を対応する $y \in \{0, \ldots, 27\}$ の整数ラベルとして扱い，LLM に「数値のみを出力」させることで回帰として直接推定する枠組みを採用している．Fine-tuning におけるモデルへの入力は，以下の 3 ロール（システム/ユーザ/アシスタント）からなるメッセージとして構成される．

システム ‘‘`Answer with a number on a scale of 28 from 0 to 27. You must output only a number.`’’

ユーザ ‘‘`Please tell me the difficulty of this competitive programming problem.:` ⟨`Problem description`⟩’’

アシスタント 正解の難易度ラベル

結果として，この Fine-tuning により先行研究よりも優れたパフォーマンスを発揮するモデルが得られることがわかった.

その他，複数の LLM に問題を解答させ，その正誤反応を事前テストの結果として代用する手法 [25] や，LLM を指定された能力の学習者として振る舞うよう強化学習し学習者集団を擬似的に生成して事前テストを行う手法が提案されている [20]，しかしいずれの手法も LLM に何度も推論を繰り返させる必要があるため計算コストが極めて高くなるという共通の問題点がある．また前者は LLM の正誤傾向と実際の学習者集団の正誤傾向は異なるという課題がある．そして後者の強化学習を用いる手法は安定した難易度推定を実現させるためのモデル構築自体に学習コストがかかるという問題点がある．

3　提案手法

3.1　準備; LLM による Embedding モデル

難易度予測において，自然言語の問題文とプログラミング言語の解答コードは，互いに密接な関係を持つ．これらを組み合わせて活用することは予測の有効性を高めるが，言語モダリティの違いにより，両者を深く比較しながら難易度予測に寄与する特徴を抽出することは困難である．先行研究では，特徴量抽出および予測において主に言語モデルを学習させる手法が取られてきたが，本研究では関係性をより深く捉えつつ計算コストを抑えるため，以下の 2 段階で構成される予測モデルを提案する．

1. **特徴量抽出**：Decoder 型 LLM をベースとし，埋め込みタスクに特化して学習されたモデルを「特徴量抽出器」として使用する．ここで埋め込みとは，テキストやコードの意味情報を固定長のベクトル表現へ変換する処理を指す．
2. **難易度予測**：抽出された特徴量を入力とし，軽量な多層パーセプトロン（MLP）を用いて難易度回帰を行う．

本構成の最大の利点は学習コストの削減にある．パラメータ数が膨大な LLM 部分を固定し，後段の軽量な MLP のみを学習対象とすることで，計算リソースと学習時間を大幅に削減しつつ，実用的な予測モデルの構築を可能とする．

3.1.1　特徴量抽出器

特徴量抽出器として，Decoder 型 LLM をベースとした埋め込みモデル（Qwen3 Embedding 等）を採用する．本モデルでの抽出プロセスには以下の 2 つの重要な特性がある．

因果的注意機構による意味理解　従来の埋め込みモデルの多くは Encoder 型（BERT 等）であったが，本モデルは Decoder 型であり，因果的注意機構（Causal Attention）を有する．入力シーケンスの末尾にある [EOS] トークンの隠れ状態を最終的な埋め込み表現として抽出することで，LLM が持つ文脈理解能力を転用する．ベースとなる LLM は大規模データセットで事前学習されており，高い推論能力と知識を有する．この LLM をベー

スに学習された埋め込みモデルを用いることで，問題文と解答コードの論理的なつながりや依存関係を深く捉えることが可能となる．実際，コード理解のベンチマーク（MTEB Code）においても，同モデルは Gemini-Embedding 等の商用モデルを凌駕するスコアを記録しており [26]，この能力は難易度予測に必要な特徴量の取得に効果的であると考えられる．

指示追従能力の活用　ベースとなる LLM の指示追従能力を活用できる点も大きな特徴である．「プログラミング問題の難易度予測タスクのために埋め込みを行ってください」といった自然言語の指示を入力に含めることで，タスクの文脈に即した最適な特徴空間へのマッピングが可能となる．これにより，追加学習を行うことなく難易度予測という特定の目的に特化した特徴抽出を実現する．

なお，使用するモデルは InfoNCE フレームワークに基づく対照学習（Contrastive Learning）によって事前学習段階で最適化されている．バッチサイズ N における損失関数 $\mathcal{L}_{embedding}$ は以下のように定義され，正例 d^+ との類似度を高め，負例との類似度を下げるよう学習されている．

$$\mathcal{L}_{embedding} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(q_i,d_i^+)/\tau}}{Z_i} \tag{3.1}$$

ここで，$s(\cdot,\cdot)$ はコサイン類似度関数，τ は温度パラメータ，Z_i は正規化項である．本研究ではこの学習済みモデルのパラメータを固定し，特徴量抽出器としてのみ利用する．

3.1.2　難易度予測

前段で抽出されたベクトル $\mathbf{x} \in \mathbb{R}^{768}$ を入力とし，MLP により難易度 y を回帰予測する．まず

$$\mathbf{h}^{(0)} = \mathrm{LayerNorm}(\mathbf{x}) \tag{3.2}$$

とし，隠れ次元 128 の特徴変換ブロックを L 回適用する：

$$\mathbf{h}^{(l+1)} = \mathrm{Dropout}\Big(\phi\Big(\mathbf{W}_2^{(l)}\,\mathrm{Dropout}\Big(\phi\Big(\mathbf{W}_1^{(l)}\mathbf{h}^{(l)} + \mathbf{b}_1^{(l)}\Big)\Big) + \mathbf{b}_2^{(l)}\Big)\Big), \quad l = 0, \ldots, L-1, \tag{3.3}$$

ここで $\phi(\cdot) = \mathrm{GELU}(\cdot)$ である．最終表現 $\mathbf{h}^{(L)} \in \mathbb{R}^{128}$ から回帰ヘッドにより予測値を得る：

$$\hat{y} = \mathbf{w}_{\mathrm{out}}^{\top}\mathbf{h}^{(L)} + b_{\mathrm{out}}. \tag{3.4}$$

学習には平均二乗誤差（MSE）

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3.5}$$

を用い，勾配降下法により最適化する．

3.2 LLMによる特徴量抽出

本節では先行研究から更に予測精度を高めるために，LLMを活用した問題解答方法の解説，及び問題解答に必要なスキル抽出を提案する．

3.2.1 Agentic Workflowによる解答過程の信頼性担保

LLMは，Transformerアーキテクチャ[23]を基盤として構築されたニューラルネットワークモデルであり，自己教師あり学習を通じて文脈内の次単語予測（Next Token Prediction）を行う．その過程で膨大な計算資源と広範なコーパスを用いて訓練することで，単なる統計的な語彙の並び替えに留まらず文脈に応じた対話や要約，知識抽出，数学的演算などの高度な推論能力を獲得した点が技術的特異点とされる [27].

そのため，LLMに対して適切な指示と共に問題文と解答コードを与えることで，解法や使用スキル，難易度の所感といった自然言語による説明文を生成すること自体は可能である．しかし，LLMには事実に基づかないもっともらしい嘘を出力するハルシネーションという課題が知られている [28]．従来のChain-of-Thought（CoT）プロンプティング(推論過程の明示生成)は推論能力を引き出す上で有効だが，モデル内部の知識のみに依存するため，外部情報による検証が行われずハルシネーションを避けづらいという欠点がある [29].

そこで近年，LLMを単なる知識ベースとしてではなく，自律的に行動するエージェントとして扱うAgentic Workflowが注目されている．ReAct [29]やReflexion [30]といった研究では，LLMが人間のように外部ツールを使用し，その実行結果を観測して自身の出力を段階的に修正することが，高度な推論において極めて効果的であることが示されている．特に，外部ツールを用いた検証の重要性は高く，CRITIC [31]などの研究において，コード実行結果によるフィードバックがLLMの論理的整合性を担保する上で有効であることが報告されている．逆に，外部フィードバック無しでモデルが自身の出力を修正しようとするIntrinsic Self-Correctionは，推論を要するタスクにおいては改善が見られない，あるいは悪化する場合があることが指摘されている [32].

以上の知見に基づき，本研究ではAgentic Workflowを通じて「実際に問題を解き，正解に辿り着く」というプロセスを構築する．これによって得られる検証された推論過程（Verified CoT）を活用することで，信頼性の高い難易度情報を抽出する手法を提案する．

3.2.2　検証済み CoT に基づく特徴量抽出プロセス

提案手法は大きく「LLM による自律的な問題解答」と「LLM による情報の抽出」の 2 段階から構成される．全体の処理フローを図 3.1 に示す．

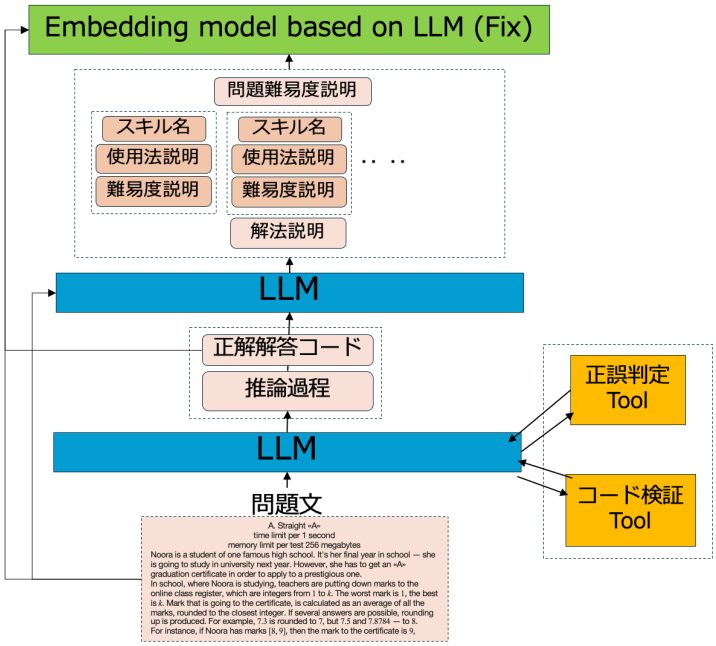

図 3.1: 全体の処理フロー図

LLM による自律的な問題解答

本研究では，ReAct [29] の枠組みを拡張し，LLM のアクション（Act）として以下の 2 つのツールを定義する．

- `exec_cpp`: C++コードをコンパイル・実行し，標準出力およびエラーメッセージをフィードバックするツール．
- `submit_solution`: 解答コードを提出し，テストケースの通過率，実行時エラー（Runtime Error），誤答時の期待される出力（Expected Output）との差分等をフィードバックするツール．

LLM に対して問題文（実行時間とメモリ制約、および入出力例を含む）とこれらのツールを与え，自律的に問題解答に取り組ませる．これにより，LLM は試行錯誤を通じて正解に至るまでのプロセスを探索する．この過程で得られる検証された推論過程には，問題文の深い理解（制約条件の確認），複数の方針の立案，採用したアルゴリズムの選択理由，テストによる検証，そしてエラー出力を受けての方針転換などが記録される．これらは，

単に正解コードを与えるだけでは得られない問題を解くための「中間作業」であり，難易度を構成する潜在的な要因を顕在化させたものである．

解答過程に基づく潜在的な難易度要因の抽出

次に，問題文，正解コード，および前段で得られた「正解に至った推論過程」を LLM に入力し，本研究で定義した潜在的な難易度要因を抽出する．具体的には,「解法の説明」,「難易度の説明（難所）」，および「解答に必要なスキル」の情報を生成させる．スキルの抽出においては，問題 p に対して必要スキル集合 $S_p = \{s_1, s_2, \ldots\}$ を推定し，その「名称」,「具体的な使用法」,「当該スキルに起因する難易度説明」を個別に記述させる．

この時，各説明文は問題固有の言葉の使用を避け，プログラミングや数学の一般的な言葉で表現するよう，強く指示する．

1. **解法説明文**: 問題解法はどのようなロジックか説明をする．
2. **難易度説明文**: 試行錯誤の過程でどこに躓いたか，どの制約が厳しかったかなどを踏まえた問題の全体的な難易度を記述する．
3. **スキル名**: 問題を解くために必要であると推定されたスキル種別の名前
4. **問題におけるスキルの使用法説明文**: 当該スキルを，本問題のどの部分でどのように用いたかを具体的に説明する．
5. **問題におけるスキルの難易度説明文**: 当該スキルを適用するうえで要求された難しい点を述べる（例：適用可否の判断根拠，計算量制約下での設計，実装上の落とし穴，反例・境界条件への対応）．

この手法により，幻覚を含む可能性のあるゼロショット生成ではなく，検証済みの解答プロセスに根拠づけられる信頼性の高い特徴量を得ることが可能となる．

3.2.3 抽出情報の定義と特徴量化

本手法により抽出されるテキスト特徴量，およびその埋め込み表現を以下のように定義する．

テキスト特徴量の定義

ある問題 p に対し，問題文 stmt_p および解答コード code_p は与えられているものとする．これに加え，前述のプロセスにより抽出されたテキストを以下のように定義する．

$$\begin{aligned}
\mathrm{sol}_p &: \text{問題 } p \text{ の解法説明文} \\
\mathrm{diff}_p &: \text{問題 } p \text{ の難易度説明文} \\
\mathrm{name}_{p,s} &: \text{問題 } p \text{ のスキル } i \text{ の名前} \\
\mathrm{usage}_{p,s} &: \text{問題 } p \text{ のスキル } i \text{ の使用法説明文} \\
\mathrm{diff}_{p,s} &: \text{問題 } p \text{ のスキル } i \text{ の難易度説明文}
\end{aligned} \tag{3.6}$$

埋め込み（テキスト → ベクトル）の定義

抽出された各テキストは，後段の予測モデルに入力するためにベクトル化を行う．ここで f はテキストを d 次元ベクトルに写像する埋め込み関数であり，本研究では事前学習済み言語モデル（Embedding Model）により実装する．各テキスト特徴量の埋め込み表現 $\mathbf{x}$ を以下のように定義する．

$$\mathbf{x}_p^{\mathrm{stmt}} := f(\mathrm{stmt}_p), \qquad \mathbf{x}_p^{\mathrm{code}} := f(\mathrm{code}_p), \tag{3.7}$$

$$\mathbf{x}_p^{\mathrm{sol}} := f(\mathrm{sol}_p), \qquad \mathbf{x}_p^{\mathrm{diff}} := f(\mathrm{diff}_p). \tag{3.8}$$

また，各スキル $s \in \mathcal{S}(p)$ に関する特徴量は以下のように定義される．

$$\begin{aligned}
\mathbf{x}_{p,s}^{\mathrm{name}} &:= f(\mathrm{name}_{p,s}), \\
\mathbf{x}_{p,s}^{\mathrm{usage}} &:= f(\mathrm{usage}_{p,s}), \\
\mathbf{x}_{p,s}^{\mathrm{diff}} &:= f(\mathrm{diff}_{p,s}).
\end{aligned} \tag{3.9}$$

3.3 問題-スキル間構造を組み込んだ Graph Neural Networks

本節では，問題とスキルの関係を 2 部グラフとしてモデル化し，グラフニューラルネットワーク (GNN: Graph Neural Networks) [33] を用いて難易度予測を行う．本手法では，

メッセージパッシングを通じて「共通スキルを介した問題間の情報共有」と「エッジ特徴を用いた情報の動的な選別」を同時に実現する．前者は類似した問題表現の一貫性を担保し，後者は難易度の根拠が関係性 (p, s) に依存するという性質を反映して，問題文脈に即した情報のみを選択的に伝播させる．

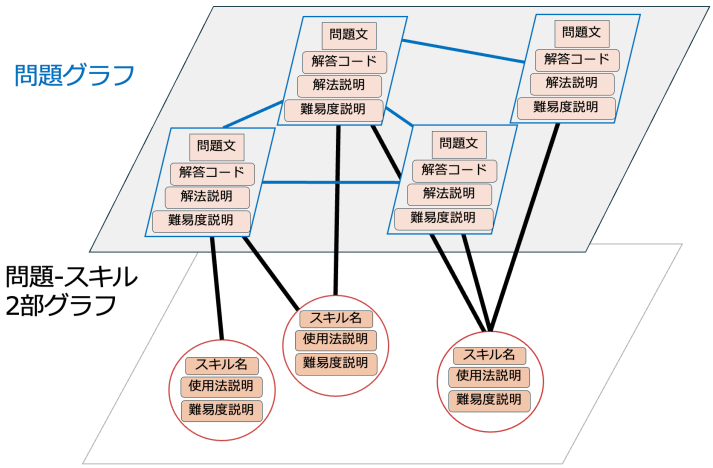

図 3.2: 問題とスキルのグラフ構造

3.3.1 GNN の基礎

GNN は，グラフ構造データに対する深層学習手法であり，各ノードが近傍ノードから情報を集約して自身の表現を更新するメッセージパッシングを基本操作とする．ノード v の第 l 層における表現 $\mathbf{h}_v^{(l)} \in \mathbb{R}^{d_h}$ は以下の式で更新される：

$$\mathbf{h}_v^{(l+1)} = \mathrm{UPDATE}\left(\mathbf{h}_v^{(l)},\ \mathrm{AGGREGATE}\left(\left\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\right\}\right)\right), \tag{3.10}$$

ここで $\mathcal{N}(v)$ はノード v の近傍ノード集合，AGGREGATE は近傍ノードの表現を集約する関数，UPDATE は自身の表現と集約結果を組み合わせる関数，d_h は隠れ層の次元である．

3.3.2 Pooling by Multihead Attention（PMA）

提案手法は可変長集合を固定長（あるいは k 個）の表現へ集約する手法として，Set Transformer [34] で提案された Pooling by Multihead Attention（PMA）を用いる．集合（インスタンス集合）を $Z = [\mathbf{z}_1; \ldots; \mathbf{z}_M] \in \mathbb{R}^{M \times d}$ （M はインスタンス数，d は埋め込み次元）とし，学習可能な k 本のシードベクトル集合を $S = [\mathbf{s}_1; \ldots; \mathbf{s}_k]^\top \in \mathbb{R}^{k \times d}$ とする．

Attention と Multi-head Attention

d 次元ベクトルからなる集合を行列として表し，行数を要素数とする．クエリ集合を $Q \in \mathbb{R}^{n\times d}$，キー集合を $K \in \mathbb{R}^{n_v\times d}$，バリュー集合を $V \in \mathbb{R}^{n_v\times d}$ とする．ここで n はクエリの要素数，n_v はキー/バリューの要素数，d は各要素の埋め込み次元である．attention は

$$\mathrm{Att}(Q,K,V;\omega) = \omega(QK^\top)\,V, \tag{3.11}$$

で定義される．ここで $\omega(\cdot)$ は行ごとに作用する重み関数である．Multi-head attention は h 個のヘッドに分けて線形射影を行う．d が h で割り切れるとし，各ヘッド次元を $d_k := d/h$ とおく．scaled softmax

$$\omega(A) = \mathrm{softmax}\left(\frac{A}{\sqrt{d_k}}\right) \tag{3.12}$$

を用いる．ここで $A := QK^\top \in \mathbb{R}^{n\times n_v}$ であり，softmax は各行に対して適用する（行和が 1 となる）．このとき

$$\mathrm{Multihead}(Q,K,V) = \mathrm{concat}(O_1,\ldots,O_h)\,W^O, \tag{3.13}$$

$$O_j = \mathrm{Att}(QW_j^Q,\ KW_j^K,\ VW_j^V;\omega), \tag{3.14}$$

と定義される．ここで $\mathrm{concat}(\cdot)$ は特徴次元方向の結合である．各ヘッドの射影行列は $W_j^Q, W_j^K, W_j^V \in \mathbb{R}^{d\times d_k}$，出力射影は $W^O \in \mathbb{R}^{hd_k\times d}(=\mathbb{R}^{d\times d})$ であり，$O_j \in \mathbb{R}^{n\times d_k}$，$\mathrm{Multihead}(Q,K,V) \in \mathbb{R}^{n\times d}$ となる．

Multihead Attention Block（MAB）

LayerNorm（LN）と残差結合を用いた Multihead Attention Block（MAB）は，2 つの集合 $X \in \mathbb{R}^{n\times d}$，$Y \in \mathbb{R}^{n_v\times d}$ に対して

$$H = \mathrm{LN}(X + \mathrm{Multihead}(X,Y,Y))\,, \tag{3.15}$$

$$\mathrm{MAB}(X,Y) = \mathrm{LN}(H + \mathrm{rFF}(H))\,, \tag{3.16}$$

で定義される [34]．ここで LN は各行（特徴次元）に対する LayerNorm である．rFF : $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ は row-wise feed-forward（各行に同一の MLP を独立に適用）であり，層数と中間次元はハイパーパラメータとする．本研究では活性化関数として GELU を用いる．なお，以下で用いる $\mathrm{rFF}(Z)$ も同様に row-wise feed-forward とし，$\mathrm{rFF} : \mathbb{R}^{M\times d} \to \mathbb{R}^{M\times d}$ として形状を保存する．

PMA の定義

PMA は，k 本のシード集合 S をクエリとして，エンコーダ出力 Z（の行ごとの変換）に注意を向けることで集約する：

$$\mathrm{PMA}_k(Z) = \mathrm{MAB}(S,\ \mathrm{rFF}(Z)) \in \mathbb{R}^{k\times d}. \tag{3.17}$$

特に $k = 1$ の場合，$\mathrm{PMA}_1(Z) \in \mathbb{R}^{1\times d}$ は集合を単一ベクトルへ集約する操作となる．本研究では S をゼロ初期化する．

3.3.3 2 部グラフの構築

問題集合 $\mathcal{P} = \{p_1, \ldots, p_N\}$ と，**??**節で定義したスキルトピック集合 $\mathcal{S}$ からなる 2 部グラフ $\mathcal{G} = (\mathcal{P}, \mathcal{S}, \mathcal{E})$ を構築する．エッジ集合は $\mathcal{E} = \{(p, s) \mid p \in \mathcal{P},\ s \in \mathcal{S}(p)\}$ と定義される．同一トピックに属するスキルは同一のスキルノードとして扱われるため，共通のスキルトピックを持つ問題間でメッセージパッシングによる情報共有が可能となる．

エンコーダの構造

入力埋め込みを隠れ次元に射影するエンコーダ $\phi : \mathbb{R}^d \to \mathbb{R}^{d_h}$ は，以下の 2 層 MLP で実装される：

$$\phi(\mathbf{x}) = \mathrm{GELU}\left(\mathbf{W}_2 \cdot \mathrm{Dropout}\left(\mathrm{GELU}\left(\mathbf{W}_1 \cdot \mathrm{LN}(\mathbf{x})\right)\right)\right), \tag{3.18}$$

ここで $\mathbf{W}_1 \in \mathbb{R}^{d_h\times d}$，$\mathbf{W}_2 \in \mathbb{R}^{d_h\times d_h}$ は学習可能な重み行列，LN は LayerNorm である．問題ノード用，スキルノード用，エッジ用にそれぞれ独立したエンコーダを使用する．

固定種類集合の融合（Multi-head mean–attn + gate）

固定長（固定本数）集合の融合には，multi-head attention による重み付きプーリングと，Highway [35] に代表されるゲート付き補間，および複数の集約表現を学習的に混合するゲート付きプーリング [36] の考え方に基づき，用途 u ごとに独立なパラメータを持つ融合演算子 $\mathrm{Fuse}_u(\cdot)$ を定義する．入力は T 個のベクトルからなる行列 $X = [\mathbf{x}_1; \ldots; \mathbf{x}_T] \in \mathbb{R}^{T\times d_h}$

として表す.

$$\bar{X}_u = \mathrm{LN}_u^{\mathrm{in}}(X), \tag{3.19}$$

$$\mathbf{m}_u(X) = \frac{1}{T}\sum_{t=1}^{T} \bar{\mathbf{x}}_{u,t} \in \mathbb{R}^{d_h}, \tag{3.20}$$

$$\mathbf{a}_u(X) = \mathrm{Multihead}_u\big(Q = \mathbf{q}_u^\top,\ K = \bar{X}_u,\ V = \bar{X}_u\big) \in \mathbb{R}^{d_h}, \tag{3.21}$$

$$\mathrm{Fuse}_u(X) = \mathrm{LN}_u^{\mathrm{out}}\Big((1-g_u)\,\mathbf{m}_u(X) + g_u\,\mathbf{a}_u(X)\Big) \in \mathbb{R}^{d_h}, \tag{3.22}$$

ここで $\mathrm{LN}_u^{\mathrm{in}}$ および $\mathrm{LN}_u^{\mathrm{out}}$ は用途 u に固有の学習可能パラメータを持つ LayerNorm である．$\bar{X}_u = [\bar{\mathbf{x}}_{u,1}; \ldots; \bar{\mathbf{x}}_{u,T}] \in \mathbb{R}^{T\times d_h}$ は入力行列 X に $\mathrm{LN}_u^{\mathrm{in}}$ を行方向に適用した正規化行列である．$\mathbf{q}_u \in \mathbb{R}^{d_h}$ は用途 u に固有の学習可能クエリベクトルであり，$\mathbf{q}_u^\top \in \mathbb{R}^{1\times d_h}$ として単一クエリのアテンションを計算する．また $g_u = \sigma(\gamma_u) \in (0,1)$ は用途 u に固有の学習可能ゲートであり，$\gamma_u \in \mathbb{R}$ は学習可能なスカラーパラメータ，σ はシグモイド関数である．

問題ノードの初期特徴量

問題 p の 4 種類の特徴ベクトル（問題文，コード，解答説明，難易度根拠）を

$$X_p^{(0)} = \big[\phi_P(\mathbf{x}_p^{\mathrm{stmt}}),\ \phi_P(\mathbf{x}_p^{\mathrm{code}}),\ \phi_P(\mathbf{x}_p^{\mathrm{sol}}),\ \phi_P(\mathbf{x}_p^{\mathrm{diff}})\big]^\top \in \mathbb{R}^{4\times d_h}. \tag{3.23}$$

とし，用途 $u = \mathrm{init}$ に対応する融合演算子を用いて

$$\mathbf{h}_p^{(0)} = \mathrm{Fuse}_{\mathrm{init}}\big(X_p^{(0)}\big) \tag{3.24}$$

と定義する．

スキルノードの初期特徴量

スキルノード $s \in \mathcal{S}$ の初期特徴量は，そのスキルトピックが出現する全問題における特徴量の平均として定義する．まず，スキル名と使用法説明文の連結埋め込みを以下のように定義する：

$$\mathbf{x}_{p,s}^{\mathrm{name}\|\mathrm{usage}} := f\left(\mathrm{concat}(\mathrm{name}_{p,s},\ \mathrm{usage}_{p,s})\right). \tag{3.25}$$

スキルトピック s が出現する問題集合を $\mathcal{P}(s) := \{p \mid s \in \mathcal{S}(p)\}$ とすると，スキルノード s の特徴量は以下のように定義される：

$$\mathbf{h}_s^{(0)} = \phi\left(\frac{1}{|\mathcal{P}(s)|}\sum_{p\in\mathcal{P}(s)} \frac{\mathbf{x}_{p,s}^{\mathrm{name}} + \mathbf{x}_{p,s}^{\mathrm{name}\|\mathrm{usage}}}{2}\right). \tag{3.26}$$

エッジ特徴量

問題 p とスキルトピック s を結ぶエッジの特徴量は，スキル名と難易度説明文の連結埋め込みを用いる：

$$\mathbf{x}_{p,s}^{\text{name}\|\text{diff}} := f\left(\text{concat}(\text{name}_{p,s},\, \text{diff}_{p,s})\right), \tag{3.27}$$

$$\mathbf{e}_{p,s} = \phi_e\left(\mathbf{x}_{p,s}^{\text{name}\|\text{diff}}\right), \tag{3.28}$$

3.3.4 メッセージパッシング

本研究のメッセージパッシングは，(i) エッジ特徴により情報伝播を条件付ける MPNN の一般枠組み [37]，(ii) 可変長近傍からの重要度付き集約を可能にする Attention 機構および Graph Transformer の設計 [**?**, 38], [**?**, 23]，(iii) エッジ属性に基づくゲーティングおよび残差更新による安定化 [**?**] に基づいて設計する．特に，問題 p におけるスキルトピック s の難易度寄与はノードではなくエッジ (p, s) に付随する情報であるため，メッセージの生成・重み付けにエッジ特徴 $\mathbf{e}_{p,s}$ を直接組み込む（edge-conditioned）[**?**]．さらに，ゲート $\mathbf{g}_{u,v}$ とゲート付き残差 g_{res} により，情報伝播の強さを学習的に制御し，過度な更新による性能悪化を抑制する [**?**].

2 部グラフ上でのメッセージパッシングを行う．本研究では，1 層モデルと 2 層モデルを比較検討する．1 層モデルでは，問題ノードとスキルノードを同一ラウンドで同時に更新する：

$$\mathbf{h}_p^{(1)} = \text{MP}_{s\to p}\left(\mathbf{h}_p^{(0)},\, \{\mathbf{h}_s^{(0)}\}_{s\in\mathcal{S}(p)},\, \{\mathbf{e}_{p,s}\}_{s\in\mathcal{S}(p)}\right), \tag{3.29}$$

$$\mathbf{h}_s^{(1)} = \text{MP}_{p\to s}\left(\mathbf{h}_s^{(0)},\, \{\mathbf{h}_p^{(0)}\}_{p\in\mathcal{P}(s)},\, \{\mathbf{e}_{p,s}\}_{p\in\mathcal{P}(s)}\right). \tag{3.30}$$

2 層モデルでは，同期更新を 2 回繰り返すことで 2-hop（$p \to s \to p$ および $s \to p \to s$）の情報伝播を実現する：

$$\mathbf{h}_p^{(1)} = \text{MP}_{s\to p}\left(\mathbf{h}_p^{(0)},\, \{\mathbf{h}_s^{(0)}\}_{s\in\mathcal{S}(p)},\, \{\mathbf{e}_{p,s}\}_{s\in\mathcal{S}(p)}\right), \tag{3.31}$$

$$\mathbf{h}_s^{(1)} = \text{MP}_{p\to s}\left(\mathbf{h}_s^{(0)},\, \{\mathbf{h}_p^{(0)}\}_{p\in\mathcal{P}(s)},\, \{\mathbf{e}_{p,s}\}_{p\in\mathcal{P}(s)}\right), \tag{3.32}$$

$$\mathbf{h}_p^{(2)} = \text{MP}_{s\to p}\left(\mathbf{h}_p^{(1)},\, \{\mathbf{h}_s^{(1)}\}_{s\in\mathcal{S}(p)},\, \{\mathbf{e}_{p,s}\}_{s\in\mathcal{S}(p)}\right), \tag{3.33}$$

$$\mathbf{h}_s^{(2)} = \text{MP}_{p\to s}\left(\mathbf{h}_s^{(1)},\, \{\mathbf{h}_p^{(1)}\}_{p\in\mathcal{P}(s)},\, \{\mathbf{e}_{p,s}\}_{p\in\mathcal{P}(s)}\right). \tag{3.34}$$

メッセージパッシング関数 MP は，エッジ特徴量を考慮したマルチヘッドアテンション機構を用いる．まず，全てのノード・エッジ特徴量に各別の Pre-LayerNorm を適用する：

$$\hat{\mathbf{h}}_u^{(l)} = \text{LN}_{\text{src}}(\mathbf{h}_u^{(l)}), \quad \hat{\mathbf{h}}_v^{(l)} = \text{LN}_{\text{tgt}}(\mathbf{h}_v^{(l)}), \quad \hat{\mathbf{e}}_{u,v} = \text{LN}_{\text{edge}}(\mathbf{e}_{u,v}). \tag{3.35}$$

エッジゲートを用いたソース特徴量の変調：

$$\mathbf{g}_{u,v} = \sigma\left(\mathbf{W}_g \hat{\mathbf{e}}_{u,v}\right), \tag{3.36}$$

$$\tilde{\mathbf{h}}_{u,v} = \left[\mathbf{g}_{u,v} \odot \hat{\mathbf{h}}_u^{(l)};\, \hat{\mathbf{e}}_{u,v}\right], \tag{3.37}$$

ここで $\mathbf{W}_g \in \mathbb{R}^{d_h \times d_h}$，$\mathbf{g}_{u,v}$ はエッジゲート，σ はシグモイド関数，$\odot$ は要素積，$[\cdot;\cdot]$ はベクトルの連結である．マルチヘッドアテンションによる重み付け（H ヘッド，各ヘッド次元 $d_k = d_h/H$）：

$$\mathbf{q}_v = \mathbf{W}_q \hat{\mathbf{h}}_v^{(l)}, \quad \mathbf{k}_{u,v} = \mathbf{W}_k \tilde{\mathbf{h}}_{u,v}, \quad \mathbf{v}_{u,v} = \mathbf{W}_v \tilde{\mathbf{h}}_{u,v}, \tag{3.38}$$

ここで $\mathbf{W}_q \in \mathbb{R}^{d_h \times d_h}$，$\mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_h \times 2d_h}$ である．$\mathbf{q}_v, \mathbf{k}_{u,v}, \mathbf{v}_{u,v}$ を H ヘッドに分割し $\mathbf{q}_v^{(h)}, \mathbf{k}_{u,v}^{(h)}, \mathbf{v}_{u,v}^{(h)} \in \mathbb{R}^{d_k}$ を得る．アテンション係数と集約は：

$$\alpha_{u \to v}^{(h)} = \frac{\exp\left(\mathbf{q}_v^{(h)\top} \mathbf{k}_{u,v}^{(h)} / \sqrt{d_k}\right)}{\sum_{u' \in \mathcal{N}(v)} \exp\left(\mathbf{q}_v^{(h)\top} \mathbf{k}_{u',v}^{(h)} / \sqrt{d_k}\right)}, \tag{3.39}$$

$$\mathbf{m}_v = \mathbf{W}_o \left[\mathbf{m}_v^{(1)}; \cdots; \mathbf{m}_v^{(H)}\right], \quad \mathbf{m}_v^{(h)} = \sum_{u \in \mathcal{N}(v)} \alpha_{u \to v}^{(h)} \mathbf{v}_{u,v}^{(h)}, \tag{3.40}$$

ここで $\mathbf{W}_o \in \mathbb{R}^{d_h \times d_h}$ は出力射影行列，$\mathcal{N}(v)$ は v の近傍集合であり，$s \to p$ 方向では $\mathcal{S}(p)$，$p \to s$ 方向では $\mathcal{P}(s)$ となる．ターゲットノードの更新はゲート付き残差接続と rFF により行う：

$$\mathbf{h}'_v = \mathbf{h}_v^{(l)} + g_{\text{res}} \cdot \text{Dropout}(\mathbf{m}_v), \tag{3.41}$$

$$\mathbf{h}_v^{(l+1)} = \mathbf{h}'_v + g_{\text{ffn}} \cdot \text{rFF}\left(\text{LN}(\mathbf{h}'_v)\right), \tag{3.42}$$

ここで $g_{\text{res}} = \sigma(\gamma_{\text{res}})$，$g_{\text{ffn}} = \sigma(\gamma_{\text{ffn}})$ は学習可能なゲートパラメータである．全てのゲートパラメータは負の値で初期化され，学習初期は残差接続が支配的となり勾配の安定化に寄与する．

3.3.5 特徴量の統合と予測

GNN による更新後，問題 p に出現する各スキルトピック $s \in \mathcal{S}(p)$ について，GNN 更新後のスキルノード特徴量と問題固有のスキル特徴量を統合する．

スキル単位での特徴量結合

問題 p におけるスキルトピック s について，GNN 更新後のスキルノード表現と問題固有特徴（使用法・難易度）を

$$X_{p,s} = \left[\mathbf{h}_s^{(L)},\, \phi_{\text{sem}}(\mathbf{x}_{p,s}^{\text{name}\|\text{usage}}),\, \phi_{\text{diff}}(\mathbf{x}_{p,s}^{\text{name}\|\text{diff}})\right]^\top \in \mathbb{R}^{3 \times d_h}, \tag{3.43}$$

とし，用途 $u = \text{skill}$ に対応する融合演算子を用いて

$$\mathbf{h}_{p,s}^{\text{comb}} = \text{Fuse}_{\text{skill}}(X_{p,s}) \tag{3.44}$$

で統合する．

問題単位でのスキル集約

統合されたスキル特徴量を PMA で集約し，問題ごとのスキル表現を得る：

$$\mathbf{z}_p^{\text{skill}} = \text{PMA}_1\left(\left\{\mathbf{h}_{p,s}^{\text{comb}}\right\}_{s\in\mathcal{S}(p)}\right). \tag{3.45}$$

問題・スキル特徴量の連結

問題ノード特徴量とスキル集約特徴量を融合し，GNN 適用前の特徴量との残差接続を行う：

$$\mathbf{z}'_p = \text{GELU}\left(\mathbf{W}_{\text{fuse}}\text{LN}\left(\left[\mathbf{h}_p^{(L)};\, \mathbf{z}_p^{\text{skill}}\right]\right)\right), \tag{3.46}$$

$$\mathbf{z}_p = g_{\text{fuse}} \cdot \mathbf{z}'_p + (1 - g_{\text{fuse}}) \cdot \mathbf{W}_{\text{pre}}\mathbf{h}_p^{(0)}, \tag{3.47}$$

ここで $g_{\text{fuse}} = \sigma(\gamma_{\text{fuse}})$ は学習可能なゲートパラメータであり，GNN 更新前の問題特徴量との内挿を制御する．

難易度の予測は，2 層のフィードフォワードネットワークからなる回帰ヘッドにより行う：

$$\hat{y}_p = \mathbf{w}_2^\top\left(\text{Dropout}\left(\text{GELU}\left(\mathbf{W}_1\mathbf{z}_p\right)\right)\right), \tag{3.48}$$

ここで $\mathbf{W}_1 \in \mathbb{R}^{d_h/2\times d_h}$，$\mathbf{w}_2 \in \mathbb{R}^{d_h/2}$ は学習可能なパラメータである．
学習は平均二乗誤差を損失関数として行う：

$$\mathcal{L} = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\left(\hat{y}_p - y_p\right)^2, \tag{3.49}$$

ここで y_p は問題 p の真の難易度，$\mathcal{P}$ は問題集合である．

3.3.6 問題グラフの適応的推定による予測精度の向上

プログラミング問題には，類似したアルゴリズムを必要とする問題や，近い難易度を持つ問題間に潜在的な依存関係が存在すると考えられる．このような関係をグラフとして明示的にモデル化し，近傍問題からの情報伝播を活用することで，より正確な難易度予測が

期待できる．しかし，問題難易度予測に関する問題間の真の関係構造は事前に与えられておらず，データから推定する必要がある．

この課題に対し，本研究ではPal et al. [39]が提案した非パラメトリックなベイズグラフニューラルネットワーク（Bayesian GNN）の枠組みを適用する．この手法は，本研究のように一つのデータ (Bag) に可変個の特徴量 (Instance) が存在するタスクにおいて、あるモデルの学習したBag表現を基にBag Graphを非パラメトリックなモデルとしてMAP推定し、推定グラフ上でグラフ畳み込みネットワーク (GCN: Graph Convolution Networks) を適用することにより予測精度を向上させるものである．本研究では，この枠組みにおけるベースモデルとして前節の2部GNNを使用する．これにより，問題とスキルの関係から獲得した表現を活用しつつ，難易度予測における問題間の潜在的な依存関係も学習に組み込むことが可能となる．

第1段階：ベースモデルによる問題表現の獲得

Pal et al. [39]の枠組みでは，まずベースモデルによりノード（バッグ）表現を獲得する．本研究では，前節の2部GNNをベースモデルとして用い，その出力である問題表現$\mathbf{z}_p$（問題ノード特徴量とスキル集約特徴量の融合により得られる）をそのまま使用する．全問題の表現を行列としてまとめると，$\hat{\mathbf{Z}} = [\mathbf{z}_{p_1}; \ldots; \mathbf{z}_{p_N}]^\top \in \mathbb{R}^{N \times d_h}$（$N = |\mathcal{P}|$ は問題数）となる．

第2段階：グラフ推定とGCN

Pal et al. [39]の枠組みの第2段階では，ベースモデルの出力から距離行列を構築し，ベイズ推定によりグラフを推定した後，推定グラフ上でGCNを適用する．

距離行列の構築　グラフ推定には，ノード間の非類似度を測る距離行列 $\mathbf{D} \in \mathbb{R}_+^{N \times N}$ が必要である．Pal et al. [39]と同様に，本研究ではベースモデル（2部GNN）により獲得した問題表現 $\mathbf{z}_p$ 間の距離を用いる．

具体的には，問題表現間のユークリッド距離を計算する：

$$D_{ij} = \|\mathbf{z}_{p_i} - \mathbf{z}_{p_j}\|_2^2. \tag{3.50}$$

この距離行列は，2部GNNが問題とスキルの関係から学習した表現に基づいており，類似したスキル構成や解法パターンを持つ問題間の距離が小さくなることが期待される．

ベイズグラフ推定　構築した距離行列 $\mathbf{D}$ を観測データとみなし，Pal et al. [39, 40] の非パラメトリックモデルを用いて真のグラフ構造 $\mathcal{G}$（隣接行列 $\mathbf{A} \in \mathbb{R}_+^{N\times N}$）を推定する．

まず，グラフの事前分布 $p(\mathcal{G})$ は，孤立したノードが存在しないこと、および極端に密にならないことを保証する. ギブス分布を用いて以下のように定式化する：

$$p(\mathcal{G}) \propto \exp\left(\alpha \mathbf{1}^\top \log(\mathbf{A}\mathbf{1}) - \beta \|\mathbf{A}\|_F^2\right), \tag{3.51}$$

ここで $\mathbf{1}$ は全要素が 1 のベクトル，$\|\cdot\|_F$ はフロベニウスノルム, $\mathbf{A}\mathbf{1}$ は各ノードの次数ベクトルである．第 1 項は孤立ノードの発生を防ぎ，第 2 項はエッジ重みの二乗和を最小化することでグラフの疎性を促進する．$\alpha, \beta > 0$ はそれぞれの寄与を制御するハイパーパラメータである．

次に，観測データ（距離行列 $\mathbf{D}$）に対する尤度 $p(\mathbf{D}|\mathcal{G})$ を，グラフ信号の滑らかさ（smoothness）の仮定 [41] に基づいて定義する．すなわち，特徴空間上で距離が近い（D_{ij} が小さい）ノード間には強いエッジ（A_{ij} が大きい）が張られるよう促す：

$$p(\mathbf{D}|\mathcal{G}) \propto \exp\left(-\|\mathbf{A} \circ \mathbf{D}\|_{1,1}\right), \tag{3.52}$$

ここで $\circ$ はアダマール積，$\|\cdot\|_{1,1}$ は要素ごとの ℓ_1 ノルムである．

これらの事前分布と尤度に基づき，最大事後確率（MAP）推定を行う．事後分布 $p(\mathcal{G}|\mathbf{D}) \propto p(\mathbf{D}|\mathcal{G})p(\mathcal{G})$ の最大化は，負の対数事後確率の最小化と等価であるため，推定される隣接行列 $\mathbf{A}_{\hat{\mathcal{G}}}$ は以下の最適化問題の解として得られる：

$$\mathbf{A}_{\hat{\mathcal{G}}} = \arg\min_{\substack{\mathbf{A}\in\mathbb{R}_+^{N\times N} \\ \mathbf{A}=\mathbf{A}^\top}} \|\mathbf{A} \circ \mathbf{D}\|_{1,1} - \alpha \mathbf{1}^\top \log(\mathbf{A}\mathbf{1}) + \beta \|\mathbf{A}\|_F^2. \tag{3.53}$$

この凸最適化問題は，Kalofolias [41] が提案した Primal-Dual アルゴリズムにより効率的に大域的最適解を求めることが可能である．本研究では，計算効率化のため k-近傍グラフに基づくマスク処理を適用した上で最適化を行う．

GCN による最終予測　推定されたグラフ $\hat{\mathcal{G}}$ 上で，グラフ畳み込みネットワーク（GCN）[?] を適用する：

$$\mathbf{H}^{(1)} = \sigma\left(\tilde{\mathbf{A}}_{\hat{\mathcal{G}}} \hat{\mathbf{Z}} \mathbf{W}^{(0)}\right), \tag{3.54}$$

ここで $\tilde{\mathbf{A}}_{\hat{\mathcal{G}}} = \mathbf{\Delta}^{-1/2} \mathbf{A}_{\hat{\mathcal{G}}} \mathbf{\Delta}^{-1/2}$ は推定グラフの正規化隣接行列，$\mathbf{\Delta} = \mathrm{diag}(\mathbf{A}_{\hat{\mathcal{G}}}\mathbf{1})$ は次数行列，$\mathbf{W}^{(0)} \in \mathbb{R}^{d_h \times d_{\mathrm{gcn}}}$ は学習可能な重み行列，σ は活性化関数（GELU）である．

GCN 出力 $\mathbf{h}_p^{(1)} \in \mathbb{R}^{d_{\mathrm{gcn}}}$ に対し，回帰ヘッドを適用して難易度を予測する：

$$\hat{y}_p = \mathbf{w}_2^\top \sigma(\mathbf{W}_1 \mathbf{h}_p^{(1)} + \mathbf{b}_1) + b_2, \tag{3.55}$$

ここで $\mathbf{W}_1 \in \mathbb{R}^{d_{\mathrm{gcn}}/2 \times d_{\mathrm{gcn}}}$，$\mathbf{b}_1 \in \mathbb{R}^{d_{\mathrm{gcn}}/2}$，$\mathbf{w}_2 \in \mathbb{R}^{d_{\mathrm{gcn}}/2}$，$b_2 \in \mathbb{R}$ は学習可能なパラメータである．

Pal et al. [39] に従い，推論時には MC ドロップアウト [?] を用いる．GCN 層にドロップアウトを適用した状態で S 回の順伝播を行い，予測の平均 $\bar{y}_p = \frac{1}{S}\sum_{s=1}^{S} \hat{y}_p^{(s)}$ を最終予測とする．

学習　Pal et al. [39] の枠組みに従い，学習は以下の手順で行う．まず 2 部 GNN（ベースモデル）を学習し，問題表現 $\mathbf{z}_p$ を抽出する．次に距離行列を構築してグラフを推定し，推定グラフ上で GCN を以下の損失関数で学習する：

$$\mathcal{L} = \frac{1}{|\mathcal{T}|}\sum_{p \in \mathcal{T}} \left(\hat{y}_p - y_p\right)^2 + \lambda \cdot \mathrm{tr}\left(\mathbf{H}^{(1)\top} \mathbf{L}_{\hat{\mathcal{G}}} \mathbf{H}^{(1)}\right), \tag{3.56}$$

ここで $\mathcal{T}$ は訓練集合，$\mathbf{L}_{\hat{\mathcal{G}}}$ は推定グラフのラプラシアン行列，λ は正則化の強度を制御するハイパーパラメータである．

4　実験結果

本章では，実際のプログラミング問題に対して難易度予測を行い，各手法の精度を確認する．

4.1　使用データセット，および評価方法

4.1.1　実験データセット

データソース：Codeforces　世界最大規模のオンライン競技プログラミングプラットフォームである Codeforces の問題データを使用する．Codeforces は 170 万人以上のユーザーを抱え，難易度は回答者の反応データをもとに計算されている．本実験では c++の解答コードの存在する問題を使用した．

難易度ラベルの定義とサンプリング　本実験で使用する難易度ラベルは，Codeforces 公式の Rating に基づいている．

- **元の Rating**: Rating 値は通常 100 刻みで 800 から 3500 の範囲で分布する．
- **抽出範囲**: 本研究では，データ分布の安定性を考慮し，800 から 2800 の範囲を対象とした．各難易度（100 刻み）に対して一律に 20 件ずつのデータを抽出した．
- **数値スケールへの変換**: 実験における学習および評価を容易にするため，上記のレーティング範囲（800–2800）を 0 から 20 までの 21 段階の整数値（クラス）に変換して扱う．

データの信頼性と検証　LLM を用いた解法抽出およびフィードバックの妥当性を担保するため，十分なテストケースを内包するデータセット [42] をベースに，以下の基準でフィルタリングを行った．

1. テストケース数が 100 以上，かつ正解データと不正解データが 100 件以上存在する問題であること．

2. テストケースを用いた解答データの検証において，True Positive Rate（TPR）および True Negative Rate（TNR）がいずれも 1 であること．

両者を満たす抽出により，LLM が生成した解法の正誤を正確に評価でき，難易度予測の根拠となる質を担保したデータセットを構築した．

4.1.2 実験設定と評価方法

実験設定 提案手法および比較手法間の公平な性能評価を行うため，ハイパーパラメータチューニングを伴う Nested Cross-Validation（入れ子状交差検証）を採用する．具体的には，モデルの汎化性能を推定するための外側ループを 10 分割（10-fold），各外部ループ内での最適なハイパーパラメータを選択するための内側ループを 3 分割（3-fold）として構成した．

モデル間の比較に際しては，計算リソースおよび探索条件によるバイアスを排除するため，各モデルにおけるハイパーパラメータの試行回数（探索コスト）を一定に揃えて実験を行った．

評価指標 先行研究に倣い，真の難易度ラベル $y_i \in \{0, \ldots, 20\}$ と推定値 $\hat{y}_i$ の一致度・近さを，RMSE，CS($\theta = n$)，MAE，MedAE，および AbsErr P90 により評価する．テストデータ数を N とし，絶対誤差を $e_i = |\hat{y}_i - y_i|$ と定義するとき，各指標は次式で与えられる．

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{4.1}$$

$$\mathrm{CS}(\theta = n) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[e_i \le n], \quad n \in \{0, 1, 3, 5\}, \tag{4.2}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} e_i, \tag{4.3}$$

$$\mathrm{MedAE} = \mathrm{median}(e_1, \ldots, e_N), \tag{4.4}$$

$$\text{AbsErr P90} = P_{90}(e_1, \ldots, e_N). \tag{4.5}$$

ここで，median$(\cdot)$ は中央値を，$P_{90}(\cdot)$ はデータの 90 パーセンタイル値を表す．

4.2 準備; LLM による Embedding モデル

本節では，大規模言語由来の Emmbedding 特化モデルによる凍結特徴量の使用が先行研究と比較して低コストに高精度に難易度予測できることを示す．先行研究の中には問題

文や解答コードから，登場する統語的特徴をカウントし難易度予測に用いる研究 [22] もあったが，これらの手法は単体では予測精度が低く，なおかつ本節の手法とアンサンブルなどで組み合わせることができるため実験の比較は行わない．

使用する言語モデルの概要を表 4.1 に示す．難易度予測の先行研究では，Full Fine-tuning 用のモデルとして BERT や BigBird が用いられていたが，本研究では小規模言語モデルの中で最新かつ高い性能を示す ModernBERT を採用する．

表 4.1: 使用する言語モデルの概要

モデル名	パラメータ数	レイヤ数	隠れ層次元 (d)
BERT-base	110M	12	768
BigBird-base	128M	12	768
ModernBERT-base†	149M	22	768
Qwen3-Embedding†	8B	36	4,096
GPT-3.5-turbo†	20B 以上 (推定)	非公開	非公開

† 本実験において直接使用したモデル．

先行研究で一般的に用いられてきた問題文と解答コードを特徴量として難易度予測の回帰性能を評価した．

表 4.2 の結果，Qwen3 embedding +inst (Fix) + MLP が RMSE 3.89，MAE 2.99 と最も高い予測性能を示した．Full Fine-Tuning を用いた比較手法は，数億規模のパラメータに対し学習データが不足しており，回帰タスクへの十分な適応が困難であった．対して本手法は，LLM 由来の豊富な知識と推論能力を固定パラメータとして活用し，指示追従機能によりタスクに適した特徴量を獲得できたため，後段の軽量な MLP のみで効率的な学習が可能であった．また計算コストの面でも，Fine-tuning が半日以上を要したのに対し，本手法は数十分でモデル構築 (ハイパーパラメータ探索) が完了し，実用上の優位性が確認された．

4.3 LLM による問題-スキル間構造を組み込んだ GNN

本節では，提案手法の有効性を検証する．

4.3.1 LLM を用いて抽出する情報の分析

LLM が正答に至った推論過程から特徴量を抽出し，その有用性を検証する．本実験では，推論モデルとして OpenAI の `o4-mini` を採用し，API 経由で利用する．また，生成

手法	RMSE↓	CS@0↑	CS@1↑	CS@3↑	CS@5↑	MAE↓	MedAE↓	P90↓
SQL-DP (+Qwen3) + MLP	4.33	0.093	0.289	0.580	0.792	3.47	2.95	7.05
GPT-3.5-turbo Fine-tuning	7.87	0.055	0.136	0.341	0.516	6.62	6.50	10.4
ModernBERT Fine-tuning	4.09	**0.100**	0.294	0.623	0.811	3.24	2.75	6.60
ModernBERT (Fix) + MLP	4.48	0.091	0.284	0.573	0.785	3.52	2.95	7.30
Qwen3 embedding (Fix) + MLP	3.94	0.079	**0.298**	0.611	0.847	3.12	2.60	6.45
Qwen3 embedding +inst (Fix) + MLP	**3.89**	**0.100**	0.296	**0.652**	**0.850**	**2.99**	**2.30**	**6.30**

表 4.2: 先行研究との性能比較（10-fold CV）

されたコードの実行およびテストケースによる正誤検証は，ローカル環境上に構築したサンドボックス内で実施する．

スキルに関する情報の比較

予備実験として，スキルの名前，使用法説明，難易度説明の難易度予測タスクへの影響を比較する．まずは問題単位で特徴量を一つ計算できるよう，スキルに関する情報を連結する．ここで，問題 p に含まれるスキルの集合を $\mathcal{S}(p)$ とし，そのサイズを $n_p = |\mathcal{S}(p)|$ とする．また，スキル集合に対して ID 昇順で順序付けを行ったスキル列を $\pi(p) = (s_p^{(1)}, \ldots, s_p^{(n_p)})$ と定義する．テキストの連結操作を $\mathrm{concat}(\cdot,\cdot)$ と表すとき，各情報の連結埋め込み $\mathbf{x}_p$ は次式で定義される．

全スキルのスキル名の連結埋め込み $\mathbf{x}_p^{\text{names}}$ は以下の通りである．

$$\mathrm{NAME}_p := \mathrm{concat}\big(\mathrm{name}_{p,s_p^{(1)}}, \ldots, \mathrm{name}_{p,s_p^{(n_p)}}\big), \tag{4.6}$$

$$\mathbf{x}_p^{\text{names}} := f(\mathrm{NAME}_p) \in \mathbb{R}^d. \tag{4.7}$$

全スキルのスキル使用法説明の連結埋め込み $\mathbf{x}_p^{\text{usage}}$ は以下の通りである．

$$\mathrm{USAGE}_p := \mathrm{concat}\big(\mathrm{usage}_{p,s_p^{(1)}}, \ldots, \mathrm{usage}_{p,s_p^{(n_p)}}\big), \tag{4.8}$$

$$\mathbf{x}_p^{\text{usage}} := f(\mathrm{USAGE}_p) \in \mathbb{R}^d. \tag{4.9}$$

全スキルのスキル難易度説明の連結埋め込み $\mathbf{x}_p^{\text{diff}}$ は以下の通りである．

$$\mathrm{DIFF}_p := \mathrm{concat}\big(\mathrm{diff}_{p,s_p^{(1)}}, \ldots, \mathrm{diff}_{p,s_p^{(n_p)}}\big), \tag{4.10}$$

$$\mathbf{x}_p^{\text{diff}} := f(\mathrm{DIFF}_p) \in \mathbb{R}^d. \tag{4.11}$$

全スキルのスキル名とスキル難易度説明の連結埋め込み $\mathbf{x}_p^{\text{name+diff}}$ は以下の通りである．

$$\begin{aligned}&\text{NAME} + \text{DIFF}_p \\ &\quad := \text{concat}\big(\text{name}_{p,s_p^{(1)}},\, \text{diff}_{p,s_p^{(1)}},\, \ldots,\, \text{name}_{p,s_p^{(n_p)}},\, \text{diff}_{p,s_p^{(n_p)}}\big), \qquad (4.12)\\ &\mathbf{x}_p^{\text{name+diff}} := f(\text{NAME} + \text{DIFF}_p) \in \mathbb{R}^d. \qquad (4.13)\end{aligned}$$

全スキルのスキル使用法説明とスキル難易度説明の連結埋め込み $\mathbf{x}_p^{\text{usage+diff}}$ は以下の通りである．

$$\begin{aligned}&\text{USAGE} + \text{DIFF}_p \\ &\quad := \text{concat}\big(\text{usage}_{p,s_p^{(1)}},\, \text{diff}_{p,s_p^{(1)}},\, \ldots,\, \text{usage}_{p,s_p^{(n_p)}},\, \text{diff}_{p,s_p^{(n_p)}}\big), \qquad (4.14)\\ &\mathbf{x}_p^{\text{usage+diff}} := f(\text{USAGE} + \text{DIFF}_p) \in \mathbb{R}^d. \qquad (4.15)\end{aligned}$$

全スキルのスキル名，スキル使用法説明，およびスキル難易度説明の連結埋め込み $\mathbf{x}_p^{\text{all}}$ は以下の通りである．

$$\begin{aligned}&\text{NAME} + \text{USAGE} + \text{DIFF}_p \\ &\quad := \text{concat}\big(\text{name}_{p,s_p^{(1)}},\, \text{usage}_{p,s_p^{(1)}},\, \text{diff}_{p,s_p^{(1)}},\, \ldots, \\ &\qquad\qquad \ldots,\, \text{name}_{p,s_p^{(n_p)}},\, \text{usage}_{p,s_p^{(n_p)}},\, \text{diff}_{p,s_p^{(n_p)}}\big), \qquad (4.16)\\ &\mathbf{x}_p^{\text{all}} := f(\text{NAME} + \text{USAGE} + \text{DIFF}_p) \in \mathbb{R}^d. \qquad (4.17)\end{aligned}$$

表 4.3 に実験結果を示す． 3 つの特徴量全てを統合した条件（NAME + USAGE + DIFF）が，RMSE（3.80）および MAE（2.97）において最も優れた性能を達成した． 単一の特徴量を用いた場合と比較して，複数の情報を組み合わせることで一貫して精度が向上していた. また，難易度説明のような難易度に関するワードをよく含む文章も単体ではなくスキル名と組み合わせることが予測精度の向上つながるとわかった.

得られたスキルの種類と難易度の関係

次に，人手で付与された既存のスキルタグと，本手法により LLM が生成したスキル情報で問題難易度予測への有効性を比較する．人手で付与されたスキルタグはそのまま Multi-hot ベクトルとする. 一方で LLM が生成したスキル情報については，スキル名でクラスタリングを行う．データセット中の 393 個のユニークなスキル名が 72 個のスキルトピックに統合された. 自動クラスタリングのみでは約 17% のトピックに異種概念の混在が見られたが，手動微調整により適切な分類が達成された. 得られたスキルトピックをスキルタグとして Multi-hot ベクトルとする.

特徴量	RMSE↓	CS@0↑	CS@1↑	CS@3↑	CS@5↑	MAE↓	MedAE↓	P90↓
NAME + USAGE + DIFF	**3.80**	**0.119**	0.306	0.671	**0.866**	**2.97**	2.60	5.78
NAME + DIFF	3.871	0.108	**0.323**	0.647	0.850	3.045	2.65	**5.68**
NAME + USAGE	3.928	0.112	0.298	**0.673**	0.845	3.07	2.65	6.44
USAGE + DIFF	3.949	0.110	0.308	0.671	0.831	3.083	**2.45**	6.46
NAME	4.002	0.091	0.282	0.642	0.843	3.164	2.80	6.08
USAGE	4.083	0.103	0.310	0.633	0.826	3.20	2.75	6.84
DIFF	4.113	0.100	0.277	0.611	0.816	3.28	2.70	6.36

表 4.3: スキル特徴量の組み合わせによる精度比較（10-fold CV）

特徴量	RMSE↓	CS@0↑	CS@1↑	CS@3↑	CS@5↑
人手スキルタグ multi-hot	4.87	0.11	0.23	0.52	0.775
LLM 生成スキルタグ multi-hot	**4.02**	0.11	**0.31**	**0.64**	**0.83**

表 4.4: タグ特徴量の比較（10-fold CV）

表 4.4 に実験結果を示す．評価の結果，LLM 生成タグを用いたモデル（RMSE 4.02）が，人手タグ（RMSE 4.87）を大きく上回る精度を示した．定性的な分析によると，人手タグでは単に「Math」と分類されていた項目が，LLM 生成タグでは「Matrix Manipulation」や「Modular Arithmetic」といった，より具体的かつ難易度予測に資する粒度で再定義されていることが確認された．これは，LLM が難易度と直接結びつくアルゴリズムの詳細な特性を捉えていることを示唆している．

組み合わせによる難易度予測比較

次に，LLM により抽出された各情報の組み合わせを変えて予測精度の変化を検証する．なお，前項の予備実験の結果に基づき，スキルに関する情報（名前・使用法説明・難易度説明）は全て連結して一つの「スキル情報」として扱う．

また，本手法の有効性を検証するため，比較実験として LLM に対し Zero-shot で直接難易度数値を推定させる検証も行う．先行研究の設定に準拠し，0 から 21 の範囲の整数値を出力するよう指示を与える [19]．実験の結果，LLM に難易度の数値そのものを直接出力させるよりも，本手法のように問題やスキルに関する説明文を一度生成させ，それを特徴量として回帰タスクに活用するアプローチの方が，高精度な予測が可能であることが明らかとなった．

テキスト特徴量の組み合わせについては，表 4.5 に示す通り，「問題文 ‖ スキル ‖ 解答コード ‖ 難易度説明」の組み合わせが最も低い RMSE（3.558）および高い CS@5（0.898）

入力文章	RMSE↓	CS@0↑	CS@1↑	CS@3↑	CS@5↑	MAE↓	MedAE↓	P90↓
問題文 ‖ スキル ‖ 解答コード ‖ 難易度説明	**3.558**	0.096	0.320	0.688	**0.898**	2.856	2.40	5.46
問題文 ‖ スキル ‖ 解答コード	3.563	**0.136**	0.330	0.685	0.897	**2.770**	2.35	**5.36**
解法説明 ‖ 解答コード	3.589	0.120	0.330	**0.699**	0.883	2.812	2.40	6.02
問題文 ‖ 解法説明	3.605	0.098	**0.344**	0.690	0.883	2.838	**2.25**	5.48
問題文 ‖ 解答コード ‖ 難易度説明	3.622	0.105	0.325	0.657	0.878	2.891	2.40	5.74
問題文 ‖ スキル ‖ 難易度説明	3.652	0.105	0.327	0.690	0.869	2.853	2.35	5.94
解法説明 ‖ スキル ‖ 解答コード	3.663	0.091	0.310	0.666	0.871	2.937	2.40	5.60
問題文 ‖ 解法説明 ‖ 解答コード	3.664	0.114	0.310	0.666	0.885	2.881	2.45	5.85
問題文 ‖ 解法説明 ‖ スキル	3.667	0.124	0.332	0.661	0.864	2.869	2.35	5.86
問題文 ‖ 解法説明 ‖ 難易度説明	3.667	0.112	0.327	0.678	0.874	2.889	2.45	5.75
解法説明 ‖ スキル ‖ 解答コード ‖ 難易度説明	3.670	0.105	0.327	0.681	0.876	2.865	2.35	5.94
問題文 ‖ 解法説明 ‖ スキル ‖ 難易度説明	3.694	0.110	0.318	0.695	0.848	2.903	2.45	5.69
問題文 ‖ スキル	3.698	0.119	0.329	0.690	0.874	2.875	2.35	5.76
スキル ‖ 解答コード ‖ 難易度説明	3.724	0.117	0.313	0.647	0.864	2.942	2.45	5.97
問題文 ‖ 解法説明 ‖ スキル ‖ 解答コード	3.752	0.110	0.305	0.666	0.855	2.969	2.45	5.98
問題文 ‖ 解法説明 ‖ スキル ‖ 解答コード ‖ 難易度説明	3.757	0.124	0.342	0.657	0.864	2.922	2.40	6.05
解法説明 ‖ 解答コード ‖ 難易度説明	3.774	0.095	0.339	0.657	0.857	2.965	2.35	5.87
問題文 ‖ 難易度説明	3.787	0.107	0.330	0.652	0.862	2.968	2.50	5.87
スキル	3.800	0.119	0.306	0.671	0.866	2.970	2.60	5.78
解法説明 ‖ スキル	3.801	0.122	0.337	0.659	0.859	2.936	**2.25**	6.27
問題文 ‖ 解法説明 ‖ 解答コード ‖ 難易度説明	3.806	0.107	0.332	0.681	0.859	2.938	2.30	5.85
解答コード ‖ 難易度説明	3.841	0.122	0.313	0.645	0.857	3.033	2.65	6.18
解法説明 ‖ 難易度説明	3.853	0.115	0.325	0.645	0.855	3.016	2.55	6.44
スキル ‖ 難易度説明	3.861	0.079	0.277	0.642	0.854	3.105	2.65	5.99
問題文 ‖ 解答コード	3.897	0.098	0.306	0.647	0.854	3.093	2.65	6.24
スキル ‖ 解答コード	3.905	0.115	0.296	0.657	0.840	3.077	2.65	6.45
解法説明 ‖ スキル ‖ 難易度説明	3.906	0.110	0.311	0.644	0.843	3.037	2.30	6.27
難易度説明	3.921	0.096	0.296	0.654	0.838	3.098	2.65	6.32
解法説明	3.930	0.105	0.327	0.649	0.859	3.026	**2.25**	6.37
解答コード	4.067	0.105	0.284	0.611	0.838	3.209	2.70	6.46
問題文	4.406	0.076	0.282	0.582	0.809	3.475	2.80	7.35
o4-mini zero-shot (問題文 ‖ 解答コード)	6.262	0.033	0.076	0.272	0.532	5.49	5.20	9.50

表 4.5: 入力文章の組み合わせによる精度比較（10-fold CV）

を達成した. 全体的な傾向として，上位のスコアを記録した組み合わせは複数の情報を統合しており，単一の情報のみを用いるよりも多角的な情報を組み合わせた方が予測精度が向上することが確認された. 一方で，単一の特徴量を用いた場合の中では，スキル特徴量のみを用いたケースが最も高い精度（RMSE 3.800）を示しており，難易度予測においてスキル情報が重要な役割を果たしていることが示唆される.

しかし，組み合わせのパターンは膨大であり，探索的な選択には限界がある. そこで次節では，これらの自然言語情報の最適な組み合わせを自動的に選択するための手法について比較検討を行う.

手法	入力	RMSE↓	CS@0↑	CS@1↑	CS@3↑	CS@5↑	MAE↓	MedAE↓	P90↓
MLP	連結	3.757	0.124	0.342	0.657	0.864	2.922	2.40	6.05
Set Trans.	個別	3.491	0.117	0.317	0.704	0.888	2.768	2.45	5.36
2 部 GNN (Random)	個別	3.561	0.115	0.334	0.714	0.886	2.760	2.35	5.66
2 部 GNN	個別	3.335	0.117	0.356	0.733	0.897	2.584	**2.00**	5.29
2 部 GNN + ProbGraph GCN	個別	3.420	0.117	0.313	0.700	0.897	2.560	2.20	5.34
2 部 GNN + ProbGraph MAP 推定 GCN	個別	**3.280**	**0.124**	**0.358**	**0.745**	**0.902**	**2.530**	**2.00**	5.20

表 4.6: ベースラインとの精度比較（10-fold CV の平均と標準偏差）

4.3.2 問題-スキルの関係および，問題間の関係を活用した難易度予測

本実験では，LLM によって抽出された複数のテキストを，それぞれ個別の Embedding モデルで埋め込み，独立した特徴ベクトルとして扱う．そのため，この予測タスクにおける入力は，単一の長いベクトルではなく，5 つ以上のベクトルからなる集合（Set）として定義される．

単純な結合では，入力要素の順序に依存しないという集合の性質（置換不変性）を扱えず，また各特徴量間の動的な相互作用を捉えることが困難である．このような可変長集合データの処理において，現在最も標準的かつ高い精度を示しているのが，Lee らによって提案された Set Transformer [34] である．

Set Transformer は，Transformer の注意機構を可変長集合データ向けに拡張したモデルであり，入力要素間の関係性を捉えながら特徴を集約することに優れている．MIL タスクにおいて，従来の DeepSets 等の手法を凌駕する精度を記録しており，集合入力からの回帰タスクにおける強力なベースラインモデルとして位置づけられる．本実験では，この Set Transformer をベースラインとする．Set Transformer への入力は提案手法と同条件に揃え，$\mathbf{x}_p^{\mathrm{stmt}}$,$\mathbf{x}_p^{\mathrm{code}}$,$\mathbf{x}_p^{\mathrm{sol}}$,$\mathbf{x}_p^{\mathrm{diff}}$ および各スキル $s \in \mathcal{S}(p)$ に対して $\mathbf{h}_s^{(0)}$, $\mathbf{x}_{p,s}^{\mathrm{name}}$, $\mathbf{x}_{p,s}^{\mathrm{name}\|\mathrm{usage}}$, $\mathbf{x}_{p,s}^{\mathrm{name}\|\mathrm{diff}}$ を与える．

表 4.6 に，各手法による難易度予測精度の比較結果を示す．Set Transformer においては，スキル情報 (名前・使用法説明・難易度説明) に関して個別に入力とするケースとスキル単位でまとめてから入力とするケースの 2 通りを試した結果，個別に入力するケースで RMSE が小さくなったためこちらを表に記載している．またすべての文章を結合して MLP で予測する方法を表の一番上に示す．

評価の結果，提案手法である GNN + ProbGraph 推定 GCN が，RMSE（3.280）および MAE（2.530）を含むほぼ全ての指標において最も優れた精度を達成した．以下に，各比較実験から得られた知見を詳述する．

設定	RMSE↓	(std)	CS@0↑	CS@1↑	CS@3↑	CS@5↑	MAE↓	MedAE↓	P90↓
2 部 GNN（Full）	**3.335**	**0.380**	0.117	0.356	**0.733**	0.897	**2.584**	**2.00**	**5.29**
w/o エッジ特徴量連結	3.439	0.440	0.122	0.346	0.730	**0.902**	2.678	2.15	5.39
w/o Attention 集約	3.523	0.433	0.136	**0.358**	0.709	0.886	2.729	2.35	5.59
w/o Edge Gating	3.547	0.408	**0.148**	0.339	0.700	0.888	2.755	2.25	5.52

表 4.7: Ablation 実験：各コンポーネント除去の影響（10-fold CV）

グラフ構造の有効性の検証

まず，問題-スキル間構造の導入効果を検証するため，正しいグラフ構造を用いた GNN と，ノード間の接続をランダムに設定した GNN (Random) の精度を比較した．表より，ランダムなグラフ構造を用いた場合（RMSE 3.561）と比較して，適切なグラフ構造を用いた GNN（RMSE 3.335）は大幅に精度が向上していることが確認できる．両モデルは同数のパラメータを持ち，層の深さも同一であることから，この精度差はモデルの表現力向上によるものではなく，問題とスキル間の依存関係（グラフ構造）を明示的に扱うことが難易度予測において本質的に有効であることを示唆している．

また，GNN ベースの手法は Set Transformer をも上回る精度を示しており，全結合的な相互作用よりも，スキルと問題の関係性に基づいた局所的な構造化が予測に寄与していると考えられる．

確率的グラフ推定の効果

次に，問題間の潜在的な関係性を捉えるためのグラフ推定機構の効果について考察する．2 部 GNN + ProbGraph GCN は，事前に定義されたグラフ構造のみを用いたモデルであり，GNN + ProbGraph MAP 推定 GCN は，学習過程で動的にグラフ構造（隣接行列）を推定・更新する機構を組み込んだモデルである．

結果として，グラフ推定を行わない 2 部 GNN + ProbGraph GCN（RMSE 3.340）は，ベースラインの GNN（RMSE 3.335）と比較して性能が停滞，あるいはわずかに悪化する傾向が見られた．これは，固定的なグラフ構造だけでは捉えきれないノイズや，不完全な接続関係が予測の妨げになった可能性が考えられる．一方で，確率的グラフ推定を導入した 2 部 GNN + ProbGraph 推定 GCN は，RMSE を 3.280 まで改善し，最も良い精度を記録した．これは，モデルがタスクを解く過程で，難易度予測にとって真に重要な問題間の潜在的なつながりを適応的に学習・補完したためであると推察される．特に CS@5（0.904）や P90（2.13）といった指標での改善が顕著であり，予測を大きく外すケース（外れ値）の抑制に，動的なグラフ構造の最適化が寄与していることが考えられる．

メッセージパッシング機構の寄与（Ablation）

表 4.7 の結果に基づき，各コンポーネントの寄与を確認する．まず，エッジ特徴量連結を除去すると RMSE は 3.335 から 3.439 へ悪化し，MAE/MedAE/P90 も一貫して増加した．この結果は，問題 p とスキル s の組 (p, s) に依存するエッジ情報を明示的に利用する設計が，予測精度の向上に寄与していることを示唆する．次に，Attention 集約を除去すると RMSE は 3.523 まで悪化し，特に MedAE（2.35）および P90（5.59）は比較手法の中で最も悪い値を示した．これは，近傍スキルの寄与を一様に扱う場合と比較して，Attention による重み付けが予測の安定性や大きな誤差の抑制に効果的であることを示している．最後に，Edge Gating を除去した場合は RMSE（3.547）および MAE（2.755）において最も性能低下が大きく，エッジ特徴量に基づいて情報の通過量を調整する機構が，全体的な予測精度（平均的な誤差）の低減に不可欠であることが分かる．以上より，提案 2 部 GNN では，Edge Gating と Attention による適応的集約が主要な改善要因であり，エッジ特徴量連結が追加的に精度を上積みする役割を担っていると言える．

5　おわりに

本研究では，プログラミング問題の難易度推定において，大規模言語モデル（LLM）を用いて解答プロセスに介在する「解法説明」や「必要スキル」といった潜在的な難易度要因を顕在化させ，これらをグラフニューラルネットワーク（GNN）の特徴量として活用する手法を提案した．具体的には，問題とスキルの関係性を 2 部グラフとしてモデル化し，さらに問題間の潜在的な依存関係を適応的に推定することで，問題とスキルの相互作用や問題間の関連性を考慮した難易度推定を実現した． 実験の結果，提案手法は LLM のパラメータを固定した計算コストの低い学習設定でありながら，RMSE 3.280 を達成し，従来の Fine-tuning ベースの手法やベースラインモデルを上回る精度を示した．

今後の課題としては，抽出されたスキル間の階層構造の自動獲得が挙げられる．また，本研究ではスキルを予測精度の向上を目的とした特徴量として利用することに主眼を置いており，個々のスキルが難易度に与える影響の定量的な説明や，予測根拠の解釈性については今後の検討課題として残されている．

最後に，本研究は，自然言語の問題文とプログラミング言語の解答コードという直接比較が困難な二つのモダリティからなるプログラミング問題に対して，両者の間を結ぶ情報として解法説明や必要スキル等を補完するアプローチを提案した．ここで着目した「問題」と「その解決に必要なスキル」の関係性は，プログラミング問題に限定されない普遍的な構造である．したがって，本手法は他分野の問題難易度予測への応用も期待され，さらなる検証が望まれる．

参考文献

[1] Inn-Chull Choi and Youngsun Moon. Predicting the difficulty of efl tests based on corpus linguistic features and expert judgment. Language Assessment Quarterly, Vol. 17, No. 1, pp. 18–42, 2020.

[2] Zhengyang Wu, Ming Li, Yong Tang, and Qingyu Liang. Exercise recommendation based on knowledge concept prediction. Knowledge-Based Systems, Vol. 210, p. 106481, 2020.

[3] Gui Ying Han and Xi Zuo Li. An intelligent test paper generation algorithm based on adjustment of overall difficulty degrees. Applied Mechanics and Materials, Vol. 411, pp. 2879–2882, 2013.

[4] Kazuma Fuchimoto, Shin-Ichi Minato, and Maomi Ueno. Automated parallel test forms assembly using zero-suppressed binary decision diagrams. IEEE Access, Vol. 11, pp. 112804–112813, 2023.

[5] Frank B Baker, Seock-Ho Kim, et al. The basics of item response theory using R, Vol. 10. Springer, 2017.

[6] Ronald K Hambleton and Russell W Jones. Comparison of classical test theory and item response theory and their applications to test development. Educational measurement: issues and practice, Vol. 12, No. 3, pp. 38–47, 1993.

[7] Yasmine H El Masri, Steve Ferrara, Peter W Foltz, and Jo-Anne Baird. Predicting item difficulty of science national curriculum tests: The case of key stage 2 assessments. The Curriculum Journal, Vol. 28, No. 1, pp. 59–82, 2017.

[8] Pedro Hontangas, Vicente Ponsoda, Julio Olea, and Steven L Wise. The choice of item difficulty in self-adapted testing. European Journal of Psychological Assessment, Vol. 16, No. 1, p. 3, 2000.

[9] John Rust and Susan Golombok. Modern psychometrics: The science of psychological assessment. Routledge, 2014.

[10] Luca Benedetto, Andrea Cappelli, Roberto Turrin, and Paolo Cremonesi. R2de: a nlp approach to estimating irt parameters of newly generated questions. In Proceedings of the tenth international conference on learning analytics & knowledge, pp. 412–421, 2020.

[11] Anastassia Loukina, Su-Youn Yoon, Jennifer Sakano, Youhua Wei, and Kathy Sheehan. Textual complexity as a predictor of difficulty of listening items in language proficiency

tests. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 3245–3253, 2016.

[12] Samah AlKhuzaey, Floriana Grasso, Terry R Payne, and Valentina Tamma. Text-based question difficulty prediction: A systematic review of automatic approaches. International Journal of Artificial Intelligence in Education, Vol. 34, No. 3, pp. 862–914, 2024.

[13] Ricardo Conejo, Eduardo Guzmán, Jose-Luis Perez-De-La-Cruz, and Beatriz Barros. An empirical study on the quantitative notion of task difficulty. Expert Systems with Applications, Vol. 41, No. 2, pp. 594–606, 2014.

[14] Elena Verdú Pérez, Luisa M Regueras Santos, María Jesús Verdú Pérez, Juan Pablo de Castro Fernández, and Ricardo García Martín. Automatic classification of question difficulty level: Teachers' estimation vs. students' perception. In 2012 frontiers in education conference proceedings, pp. 1–5. IEEE, 2012.

[15] Fu-Yuan Hsu, Hahn-Ming Lee, Tao-Hsing Chang, and Yao-Ting Sung. Automated estimation of item difficulty for multiple-choice tests: An application of word embedding techniques. Information Processing & Management, Vol. 54, No. 6, pp. 969–984, 2018.

[16] Jia Xu, Tingting Wei, and Pin Lv. Sql-dp: A novel difficulty prediction framework for sql programming problems. International Educational Data Mining Society, 2022.

[17] Ya Zhou and Can Tao. Multi-task bert for problem difficulty prediction. In 2020 international conference on communications, information system and computer engineering (cisce), pp. 213–216. IEEE, 2020.

[18] Juntae Kim, Eunjung Cho, and Dongbin Na. Problem-solving guide: Predicting the algorithm tags and difficulty for competitive programming problems. AAAI 2024 Workshop, 2024.

[19] Chihiro Yoshida, Makoto Matsushita, and Yoshiki Higo. Estimating the difficulty of programming problems using fine-tuned llm. In 2024 IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA), pp. 28–34. IEEE, 2024.

[20] Alexander Scarlatos, Nigel Fernandez, Christopher Ormerod, Susan Lottridge, and Andrew Lan. Smart: Simulated students aligned with item response theory for question difficulty prediction. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 25082–25105, 2025.

[21] Victoria Yaneva, Kai North, Peter Baldwin, Le An Ha, Saed Rezayi, Yiyun Zhou, Sagnik Ray Choudhury, Polina Harik, and Brian Clauser. Findings from the first shared task on automated prediction of difficulty and response time for multiple-choice questions. In Ekaterina Kochmar, Marie Bexte, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Anaïs Tack, Victoria Yaneva, and Zheng Yuan, editors, Proceedings of the 19th Workshop

on Innovative Use of NLP for Building Educational Applications (BEA 2024), pp. 470–482, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[22] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. A similarity-based theory of controlling mcq difficulty. In 2013 second international conference on e-learning and e-technologies in education (ICEEE), pp. 283–288. IEEE, 2013.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, Vol. 30, , 2017.

[24] Foteini Grivokostopoulou, Ioannis Hatzilygeroudis, and Isidoros Perikos. Teaching assistance and automatic difficulty estimation in converting first order logic to clause form. Artificial Intelligence Review, Vol. 42, No. 3, pp. 347–367, 2014.

[25] Yuto Tomikawa and Masaki Uto. Difficulty-controllable multiple-choice question generation using large language models and direct preference optimization. arXiv preprint arXiv:2510.19265, 2025.

[26] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025.

[27] Ben Mann, Nick Ryder, Melanie Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, Vol. 1, No. 3, p. 3, 2020.

[28] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. ACM computing surveys, Vol. 55, No. 12, pp. 1–38, 2023.

[29] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations, 2022.

[30] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, Vol. 36, pp. 8634–8652, 2023.

[31] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. arXiv preprint arXiv:2305.11738, 2023.

[32] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In International Conference on Machine Learning, pp. 10764–10799. PMLR, 2023.

[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE transactions on neural networks, Vol. 20, No. 1, pp. 61–80, 2008.

[34] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. In International Conference on Machine Learning, Vol. 4, 2019.

[35] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[36] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In Artificial intelligence and statistics, pp. 464–472. PMLR, 2016.

[37] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In International conference on machine learning, pp. 1263–1272. Pmlr, 2017.

[38] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699, 2020.

[39] Soumyasundar Pal, Antonios Valkanas, Florence Regol, and Mark Coates. Bag graph: Multiple instance learning using bayesian graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, Vol. 36, pp. 7922–7930, 2022.

[40] Soumyasundar Pal, Saber Malekmohammadi, Florence Regol, Yingxue Zhang, Yishi Xu, and Mark Coates. Non parametric graph learning for bayesian graph neural networks. In Conference on uncertainty in artificial intelligence, pp. 1318–1327. PMLR, 2020.

[41] Vassilis Kalofolias. How to learn a graph from smooth signals. In Artificial intelligence and statistics, pp. 920–929. PMLR, 2016.

[42] Zihan Wang, Siyao Liu, Yang Sun, Hongyan Li, and Kai Shen. Codecontests+: High-quality test case generation for competitive programming. arXiv preprint arXiv:2506.05817, 2025.