自然言語情報を用いた未知タスクの性能予測

佐藤 拓真 [1,2,a]　平岡 太郎 [3,2]　鈴木 陽登 [4,2]　菅原 朔 [2,5]
[1] 奈良先端科学技術大学院大学　[2] 国立情報学研究所
[3] 北海道大学　[4] 慶應義塾大学　[5] 東京大学
[a] sato.takuma.sq6@naist.ac.jp

概要

大規模言語モデルが個々の具体的なユースケース (未知タスク) においてどの程度の性能を発揮するかを、自前のデータセットやベンチマークを構築せずとも予測できれば、研究やビジネス上の意思決定において有用である。本研究では、未知タスクについての自然言語による簡潔な説明と 1 題のみの例題から、そのタスクについての各種分析テキストを LLM によって生成し、その埋め込みを性能予測器への入力特徴量として用いる手法によって、未知タスクの性能予測の正確さを向上させた。分析の結果、とりわけ、例題に対する模範解答が予測の正確さの向上に大きく寄与することが分かった。

1　はじめに

大規模言語モデル (LLM) のユーザが、個々のユースケースにおけるモデルのパフォーマンスを予想したり、性能の高いモデルを選択するには、様々な下流タスクにおけるベンチマークスコアを参照することが通常である。しかしながら、それらのベンチマークとユーザがターゲットとする実際のタスクには、多かれ少なかれ乖離がある [1]。したがって、ターゲットタスクでモデルがどの程度のパフォーマンスを発揮できるかを正確に把握するためには、実際にベンチマークを作成し実験を行う必要があるが、これには大きな労力とコストがかかる [2, 3]。

このような問題を解決するための方法として、既知タスクにおける実験の結果を使用して、未知タスクにおけるモデルの性能を予測する手法が探求されてきた (図 1 上) [4, 5, 6]。これらの研究の動機の一つは、もし正確な性能予測が実現されれば、ベンチマークを自前で作成し実際に実験を行う手間が省かれることである [7]。先行研究では、カテゴリカルな特徴量を入力とする機械学習モデル [8, 9] や、テキストを入力とする LLM を性能予測器として用いる方法 [7] が提案されている。しかしながら、これまでの研究で達成されてきた未知タスクの性能予測の正確さは、実用上必ずしも十分なものではない。我々はこの課題を、性能予測器に与える情報の不十分さに起因するものと考えた。先行研究において性能予測器に入力される情報は、予測対象タスクの粗いカテゴリやモデルのパラメータ数、タスクと実験設定のテキストによる簡潔な説明など単純なものに留まっており、これらのみではタスクの性質を十分に表現できない。本研究ではこの問題に対処するため、タスクの性質を自然言語で記述したテキストから得られる embedding を、未知タスクにおけるモデル性能の予測に使う手法を提案する (図 1 下)。

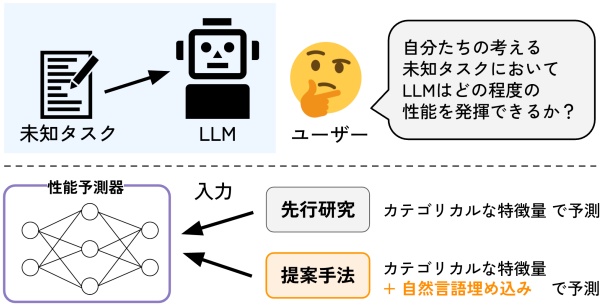

図 1　本研究の概念図。**上:** 本研究は、個々のユーザがターゲットする未知タスク上で LLM が発揮する性能を正確に予測することを目標とする。**下:** 先行研究では性能予測器への入力としてカテゴリカルかつ単純な特徴量のみを用いていたが、本研究の提案手法ではタスクの性質を記述した自然言語テキストの埋め込みを付加的に用いる。

本研究は、未知タスクにおけるモデルの性能を予測するタスクを取り扱う。このタスクは、「性能予測の対象となるタスクの性質を表現する特徴量 (タスク特徴量)」と「性能予測の対象となるモデルの ID (モデル特徴量)」を入力として、当該タスクにおける当該モデルの性能 (例: accuracy) を出力する回帰タスクである。例えば以下のような場面が現実の

ユースケースとして考えられる。

- 社内のある作業を LLM で自動化する企画があるが、その作業の自動化を LLM がどの程度正確に行うことができるかが分からないため、企画を実行に移すべきかが判断できない。
- LLM のベンチマークデータを作成する研究に着手しようとしているが、最新のモデルに対してそのベンチマークが十分な難度を持っているかを判断できない。

このタスク上での実験によって、以下のリサーチクエスチョン (RQ) とそれに対する回答を得た。

RQ1　タスクの性質を自然言語で記述したテキストの embedding を性能予測器への入力特徴量として用いること (提案手法) は、未知タスクの性能予測の正確さ向上に寄与するか？ →する。ただし、重視する評価指標によっては embedding を入力特徴量とする性能予測器よりも、LLM に当該記述のテキストを直接入力して予測性能を出力させたほうがよい場合もある。

RQ2　RQ1 に対して「寄与する」との結果が得られる場合、どのような自然言語記述の embedding を用いることが、性能予測の正確さ向上に大きく寄与するか？ →タスクの例題への模範解答を用いることが最も大きく寄与する。

2　タスクと関連研究

まず、本研究が取り扱う性能予測 (performance prediction) タスク [4] を定式化する。特定の LLM を $m \in \mathcal{M}$、特定のタスクを $t \in \mathcal{T}$ とする。$\mathcal{M}, \mathcal{T}$ はそれぞれモデル集合、タスク集合である。m が t において達成するスコアを $s(m,t) \in \mathbb{R}$ とし、スコアの集合を $\Omega \subseteq \mathcal{M} \times \mathcal{T}$ とする。ここで、スコア行列 $S \in \mathbb{R}^{|\mathcal{M}| \times |\mathcal{T}|}$ の (m,t) 要素を $S_{m,t} = s(m,t)$ で定義する。本研究で扱うタスクは、入力として組 (m,t) が与えられた時に $S_{m,t}$ を予測して出力する、回帰的な行列補完タスクである。

性能予測タスクの研究は、**既知タスク**を対象とするもの [4, 8, 9] と**未知タスク**を対象とするもの [5, 6, 7] に分けることができる。既知タスクを対象とする設定では、$s(m,t)$ の予測器の学習において m 以外のモデルが t で達成したスコアを用いることを許容する。学習データに t についてのデータが含まれるため、予測器にとって t は既知のタスクである。既知タスクの性能は、モデルやタスクのカテゴリカルなメタ情報 (例えば、パラメータ数や評価指標の種類) を特徴量として MLP を学習することで、高い正確さで予測できることが分かっている [8]。学習と予測を協調フィルタリング (§3) の枠組みにおいて行うことも有効であると報告されている [9]。一方、未知タスクを対象とする設定では、$s(m,t)$ の予測器の学習において、t に対するいかなるモデルのスコアも用いることを許容しない。つまり、学習において予測器は t に関する情報に一切接しておらず、この意味で t は予測器にとって未知のタスクである。未知タスクの性能予測は既知タスクと比べて遥かに困難である [8, 9] が、これを実現することはビジネスや研究における意思決定に寄与するため [7]、重要な課題である。本研究は、未知タスクを対象とする性能予測のために新手法を提案するものである。

3　提案手法

未知タスクの性能予測の正確さを向上するため、タスクの性質の自然言語記述を embedding 化し、性能予測器への入力特徴量として使用するニューラル協調フィルタリング (NCF) [10] の手法を提案する。まず、行列分解を用いた協調フィルタリングと NCF について、本研究の設定に読み換えながら簡単に導入を行い、それから提案手法について説明する。

行列分解を用いた協調フィルタリング (MFCF) は、推薦システムで広く使われる技術である [11, 12]。推薦システムにおいては、$S_{u,i}$ がユーザ $u \in \mathcal{U}$ のアイテム $i \in \mathcal{I}$ に対する嗜好 (例えばレビュー点数) を表現するようなスコア行列 S を考える。これを本研究の設定に読み換えるには、モデル $m \in \mathcal{M}$ がタスク $t \in \mathcal{T}$ で達成するスコアを $S_{m,t}$ で表現するようなスコア行列 S を考えればよい。ここで、S を $S = P^T \cdot Q$ として、モデルとタスクの特性を表現する潜在ベクトル行列 $P \in \mathbb{R}^{d \times |\mathcal{M}|}, Q \in \mathbb{R}^{d \times |T|}$ に分解する。この P, Q が MFCF における学習の対象である。

NCF [10] は、MFCF における内積という線形な相互作用関数をニューラルネットワークという非線形な関数に置き換えることで嗜好予測の性能を向上させる手法であり、性能予測タスクにおいても有効であることが報告されている [9]。NCF のアーキテクチャは、GMF ブランチと MLP ブランチの 2 つに分かれる。まず、モデルとタスクのカテゴリカルな特徴 (例えば、モデル ID やタスクカテゴリ) を表現する one-hot ベクトルを、GMF 用と

MLP 用の埋め込み行列[1]によって射影し、それぞれ $p_m^{\mathrm{GMF}}, q_t^{\mathrm{GMF}}, p_m^{\mathrm{MLP}}, q_t^{\mathrm{MLP}}$ を得る。次に、GMF/MLP ブランチは、それぞれ以下のようにしてブランチ埋め込み $\boldsymbol{x}_{\mathrm{GMF}}, \boldsymbol{x}_{\mathrm{MLP}}$ を得る[2]。

$$\boldsymbol{x}_{\mathrm{GMF}} = p_m^{\mathrm{GMF}} \odot q_t^{\mathrm{GMF}} \tag{1}$$

$$\boldsymbol{x}_{\mathrm{MLP}} = \mathrm{MLP}(\mathrm{concat}(p_m^{\mathrm{MLP}}, q_t^{\mathrm{MLP}})) \tag{2}$$

ブランチ埋め込みを縦結合して出力層の重みベクトル $\boldsymbol{h}$ との内積を計算し、シグモイド関数に入力することで、最終的なスコアの予測値 $S_{m,t}$ を得る。

$$S_{m,t} = \sigma(\boldsymbol{h} \begin{bmatrix} \boldsymbol{x}_{\mathrm{GMF}} \\ \boldsymbol{x}_{\mathrm{MLP}} \end{bmatrix}) \tag{3}$$

提案手法は、高性能な LLM[3]によってタスクについて分析したテキストを生成し、そのテキストの埋め込みをタスク埋め込み q_t^{GMF}, q_t^{MLP} に付加する手法である。これにより、モデルはタスクの性質をより連続的かつ詳細に捉えて性能予測を行うことができると期待した。まず、t のタスク名・1 問の例題・簡潔な説明を LLM に入力し、以下の項目について分析したテキスト $a_1^t, a_2^t, a_3^t, a_4^t$ を出力として得る[4]。

$$a_1^t : \text{タスクの詳細な分析} \tag{4}$$

$$a_2^t : \text{タスクの難易度の見積もり} \tag{5}$$

$$a_3^t : \text{例題に対する模範解答} \tag{6}$$

$$a_4^t : \text{タスクに必要と推定される能力} \tag{7}$$

これらと 1 問分の例題テキスト a_5^t をテキストエンコーダ (`Qwen/Qwen3-Embedding-4B`) に入力し、埋め込み $e_1^t, e_2^t, e_3^t, e_4^t, e_5^t$ を得る。得られた埋め込みを q_t^{GMF}, q_t^{MLP} に結合し、新たなベクトル q'^{GMF}_t, q'^{MLP}_t を作る。その後は、(1) (2) (3) と同様

1) これらの埋め込み行列も学習によって得る。
2) 先行研究 [10] においては、カテゴリカルなモデル特徴量としてパラメータ数や学習トークン数が $p_m^{\mathrm{GMF}}, p_m^{\mathrm{MLP}}$ の作成に使われているが、昨今の高性能な商用 LLM ではしばしばそれらの情報は秘匿されているため、本研究ではこれらを用いない。タスク側のカテゴリカルな特徴量としては、評価指標・回答形式・親タスクを one-hot ベクトルとして用いた。
3) 具体的には、GPT-5-mini を用いた。
4) それぞれの項目は、以下の期待のもと用いた。a_1^t では、タスクの性質についての直接的な分析を用いることで、各モデルがそれを成功させることが可能かを性能予測器が予想しやすくなると期待した。a_2^t では、タスクの難易度分析によって最終的なスコアの収まる範囲が予測しやすくなると期待した。a_3^t では、模範解答がそのタスクを解くのに必要なスキルやプロセスを表現すると期待した。a_4^t では、どのような種類の能力を要求するタスクであるかを知ることが、そのタスクの遂行可能性を予測するのに有用であると期待した。

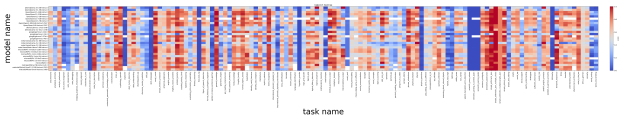

図 2 スコア行列のヒートマップ。行がモデルを、列がタスクを示す。赤色が濃くなるほど高いスコアを示す。

に、最終的なスコアの予測値 $S_{m,t}$ を得る。

$$q'^{MF}_t = \mathrm{concat}(q_t^{GMF}, e_1^t, e_2^t, e_3^t, e_4^t, e_5^t) \tag{8}$$

$$q'^{MLP}_t = \mathrm{concat}(q_t^{MLP}, e_1^t, e_2^t, e_3^t, e_4^t, e_5^t) \tag{9}$$

$$\boldsymbol{x}'_{\mathrm{GMF}} = p_m^{\mathrm{GMF}} \odot q'^{\mathrm{GMF}}_t \tag{10}$$

$$\boldsymbol{x}'_{\mathrm{MLP}} = \mathrm{MLP}(\mathrm{concat}(p_m^{\mathrm{MLP}}, q'^{\mathrm{MLP}}_t)) \tag{11}$$

$$S_{m,t} = \sigma(\boldsymbol{h} \begin{bmatrix} \boldsymbol{x}'_{\mathrm{GMF}} \\ \boldsymbol{x}'_{\mathrm{MLP}} \end{bmatrix}) \tag{12}$$

4 実験

4.1 実験設定

データ データとして、各種 LLM が BIG-bench [13] のタスクにおいて達成するスコアを収集した。126 個のタスクに対して 29 個のモデルでスコアを収集した。95 の (モデル, タスク) の組み合わせを実行時間の都合から省略し、最終的に 3,559 つのスコアを得た。以下、このデータを「スコア行列」と呼ぶ (図 2)。スコア行列の密度は 97.4%である。対象のタスクとモデルは付録の §A に示す。

比較手法 提案手法を 3 つのベースライン手法と比較する。

Mean 訓練データにおけるスコアの平均値を予測スコアとして出力する。

NCF Zhang ら (2024) [9] の手法の再現実装[5]。

LLM Park ら (2025) [7] の手法の再現実装[6]。タスクの説明や評価指標などの自然言語記述と予測対象のモデル名を LLM (GPT-5) に入力し、reasoning の後に根拠等と併せて予測スコアを出力させる[7]。なお、LLM w/ SemInfo においては、本研究で使用するタスクの自然言語記述 (式 (4-7)) を LLM への入力に付加する。

評価指標 予測の正確さの評価には回帰指標・誤差指標・ランク指標としてそれぞれ 2 つの指標を用

5) 再現実装の正しさを担保するため、当該研究 [9] における実験データを用いて学習した場合に報告されているスコアと同程度の結果が得られることを確認した。
6) NCF と同様な仕方で再現実装の正しさを確認した。
7) 本研究中で、LLM は「性能予測のターゲット」「性能予測器への入力特徴量 (埋め込み) の元となる自然言語テキストを生成する」「性能予測を行う」という 3 つの異なる場面と役割で登場する。これらを混同しないよう注意されたい。

表 1　実験結果。もっとも良好なスコアを**太字**で示す。また、自然言語情報を使用しないベースラインのなかで最も高性能な手法である NCF のスコアと提案手法のスコアについて、各 fold で得られたスコアを対応のある標本として対応のある両側 t 検定を行った。有意差は $p < 0.05$ を † で示す。

		回帰指標 (↑)		誤差指標 (↓)		ランク指標 (↑)	
	手法	R^2	r	MAE	MSE	Acc@0	Acc@3
ベースライン	mean	-0.0378	—	0.2411	0.0834	0.0752	0.1965
	NCF [9]	0.3268	0.6364	0.1924	0.0545	0.0812	0.4672
	LLM [7]	0.0335	0.4198	0.2128	0.0773	0.0828	0.4266
	LLM w/ SemInfo	-0.0767	0.5416	0.2097	0.0776	**0.1179**	**0.5540**
提案手法	NCF w/ SemInfo	**0.4218**†	**0.6905**	**0.1719**†	**0.0460**†	0.0894	0.5036

い、以下に示す計 6 つの指標で提案手法の有効性を検証する。各指標の形式的な定義は付録 §B に示す。

回帰指標　決定係数 R^2 とピアソンの相関係数 r を用いる。これらの指標は、スコアの単位に依存せず、予測の正確さを平均や分散について相対的に定量化するために用いた。

誤差指標　Mean Absolute Error (MAE) Mean Squared Error (MSE) を用いる。MSE は MAE よりも大きな誤差を重大視するような指標と言える。予測の正確さを直感的に把握するために用いた。

ランク指標　Acc@0 (モデル中の予測順位が真の順位と完全に一致したモデルの割合) と Acc@3 (予測順位と真の順位の誤差が 3 以内であったモデルの割合) を用いる。これらは、各モデルの性能の高低の順序をどれだけ正しく予測できているかを定量化するために用いた。

実験では、スコア行列の列要素（タスク) を単位として全データを train / dev / test セットに分割した[8)]。つまり、test セットでは予測器が学習時に遭遇していないタスク (未知タスク) に対する性能予測を行った。評価には 10-fold CV を採用した。各 fold における test セット上の各指標を算出したうえで、その平均に基づいてベースライン手法と提案手法を比較した。予測器の学習に用いたハイパーパラメータは付録 §C に示す。

4.2　結果

実験の結果 (表 1)、回帰指標と誤差指標では提案手法がベースライン手法を上回るスコアを達成した。r を除いた各指標では、対応のある両側 t 検定において $p < 0.05$ として統計的有意差が認められた。ランク指標では、LLM[7] の手法の入力に提案手法のテキスト特徴量 (式 (4-7)) を付加した場合に、最良の結果が得られた。つまり、ランク指標を重視する場合にもタスクの性質の自然言語記述を用いたほうがよいことは変わらないが、その場合予測器としては NCF ではなく LLM を用いたほうがよいことが示された。

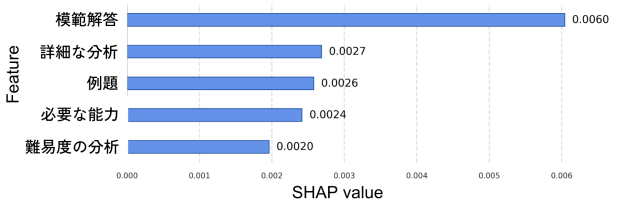

図 3　各テキスト特徴量の性能 (MAE) への寄与度を示す SHAP 値。

4.3　有用な特徴量の分析

それぞれのテキスト特徴量 (式 (4-7)) の性能予測の正確性 (MAE) への寄与度として、各特徴量の使用の有無の全組み合わせについて 2^5 回の実験を行い SHAP 値 [14, 15] (付録 §D) を計算した結果を、図 3 に示す。分析結果から、未知タスクの性能予測の正確さを高めるには、当該タスクの例題における模範的な回答プロセスのテキスト埋め込みを用いることが最も効果的であることが分かった。

5　おわりに

本研究では、タスクについての分析を記述した自然言語情報を用いることで、未知タスクの性能予測の正確さを向上させられることを示した。今後は、性能予測を更に正確に行うための手法を探求するほか、個々のモデルのタスクにおける性能を予測するための情報としてどのようなものが有用であるかを分析することで、「タスク自体の性質」の効果的な表現と分析の仕方について研究を進める。

8)　具体的には、まず全タスクを 9:1 の割合で train / test に分割し、さらに train に含まれるタスクを 9:1 の割合で train / dev に分割した。

謝辞

本研究は JST 創発的研究支援事業 JPMJFR232R の支援を受けたものです。

参考文献

[1] Michael Saxon, Ari Holtzman, Peter West, William Yang Wang, and Naomi Saphra. Benchmarks as microscopes: A call for model metrology. In **First Conference on Language Modeling**, 2024.

[2] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M. Bender, Emily Denton, and Alex Hanna. Data and its (dis)contents: A survey of dataset development and use in machine learning research. **Patterns**, Vol. 2, No. 11, p. 100336, 2021.

[3] Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 930–957, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[4] Mengzhou Xia, Antonios Anastasopoulos, Ruochen Xu, Yiming Yang, and Graham Neubig. Predicting performance for natural language processing tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 8625–8646, Online, July 2020. Association for Computational Linguistics.

[5] Zining Zhu, Soroosh Shahtalebi, and Frank Rudzicz. Predicting fine-tuning performance with probing. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, **Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing**, pp. 11534–11547, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[6] Lorenzo Pacchiardi, Konstantinos Voudouris, Ben Slater, Fernando Martínez-Plumed, Jose Hernandez-Orallo, Lexin Zhou, and Wout Schellaert. PredictaBoard: Benchmarking LLM score predictability. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, **Findings of the Association for Computational Linguistics: ACL 2025**, pp. 15245–15266, Vienna, Austria, July 2025. Association for Computational Linguistics.

[7] Jungsoo Park, Ethan Mendes, Gabriel Stanovsky, and Alan Ritter. Look before you leap: Estimating llm benchmark scores from descriptions, 2025.

[8] Qinyuan Ye, Harvey Fu, Xiang Ren, and Robin Jia. How predictable are large language model capabilities? a case study on BIG-bench. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 7493–7517, Singapore, December 2023. Association for Computational Linguistics.

[9] Qiyuan Zhang, Fuyuan Lyu, Xue Liu, and Chen Ma. Collaborative performance prediction for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 2576–2596, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In **Proceedings of the 26th International Conference on World Wide Web**, WWW '17, p. 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. **Computer**, Vol. 42, No. 8, pp. 30–37, 2009.

[12] Yehuda Koren and Robert Bell. **Advances in Collaborative Filtering**, pp. 77–118. Springer US, Boston, MA, 2015.

[13] Aarohi Srivastava, Abhinav Rastogi, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.

[14] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, **Contributions to the Theory of Games II**, pp. 307–317. Princeton University Press, Princeton, 1953.

[15] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

A　スコア行列収集の対象

§4.1 のスコア行列収集において対象としたモデルと BIG-bench のタスクを以下に示す。

```
Qwen/Qwen2.5-0.5B-Instruct, Qwen/Qwen2.5-1.5B-Instruct,
Qwen/Qwen2.5-3B-Instruct, Qwen/Qwen2.5-7B-Instruct,
Qwen/Qwen2.5-14B-Instruct, Qwen/Qwen2.5-32B-Instruct,
Qwen/Qwen2.5-72B-Instruct, Qwen/Qwen2.5-3B-Instruct,
Qwen/Qwen2.5-7B-Instruct, Qwen/Qwen3-30B-A3B-Instruct-2507,
Qwen/Qwen3-4B-Instruct-2507, allenai/Llama-3.1-Tulu-3-8B,
allenai/Llama-3.1-Tulu-3.1-8B, google/gemma-3-12b-it,
google/gemma-3-1b-it, google/gemma-3-27b-it, google/gemma-3-4b-it,
meta-llama/Llama-3.1-8B-Instruct,
meta-llama/Llama-3.2-1B-Instruct,
meta-llama/Llama-3.2-3B-Instruct,
meta-llama/Meta-Llama-3-8B-Instruct,
microsoft/Phi-3-medium-128k-instruct,
microsoft/Phi-3-medium-4k-instruct,
microsoft/Phi-3-mini-128k-instruct,
microsoft/Phi-3-mini-4k-instruct, microsoft/Phi-3.5-MoE-instruct,
microsoft/Phi-3.5-mini-instruct, microsoft/phi-4,
mistralai/Mistral-7B-Instruct-v0.3,
mistralai/Mistral-Small-3.1-24B-Instruct-2503,
mistralai/Mistral-Small-Instruct-2409
```

```
anachronisms, analogical_similarity, analytic_entailment,
ascii_word_recognition, auto_categorization, auto_debugging,
bridging_anaphora_resolution_barqa, causal_judgment,
checkmate_in_one, chinese_remainder_theorem,
code_line_description, codenames, common_morpheme,
contextual_parametric_knowledge_conflicts, crash_blossom,
crass_ai, cryobiology_spanish, cryptonite, dark_humor_detection,
date_understanding, disambiguation_qa,
discourse_marker_prediction, disfl_qa, dyck_languages,
emoji_movie, emojis_emotion_prediction, empirical_judgments,
english_proverbs, english_russian_proverbs, entailed_polarity,
entailed_polarity_hindi, epistemic_reasoning,
evaluating_information_essentiality, fantasy_reasoning,
few_shot_nlg, figure_of_speech_detection,
formal_fallacies_syllogisms_negation,
gender_inclusive_sentences_german, general_knowledge,
geometric_shapes, gre_reading_comprehension,
hindi_question_answering, hindu_knowledge, hinglish_toxicity,
human_organs_senses, hyperbaton, identify_math_theorems,
identify_odd_metaphor, implicatures, implicit_relations,
intent_recognition, international_phonetic_alphabet_nli,
international_phonetic_alphabet_transliterate,
irony_identification, kannada, known_unknowns,
language_identification, linguistics_puzzles, logic_grid_puzzle,
logical_args, logical_fallacy_detection, logical_sequence,
mathematical_induction, matrixshapes, medical_questions_russian,
metaphor_boolean, misconceptions, misconceptions_russian,
mnist_ascii, moral_permissibility, movie_dialog_same_or_different,
movie_recommendation, navigate, nonsense_words_grammar,
novel_concepts, object_counting, odd_one_out, operators,
parsinlu_qa, parsinlu_reading_comprehension, penguins_in_a_table,
persian_idioms, phrase_relatedness, physical_intuition, physics,
physics_questions, play_dialog_same_or_different,
presuppositions_as_nli, qa_wikidata, question_selection,
reasoning_about_colored_objects, repeat_copy_logic, rephrase,
riddle_sense, ruin_names, salient_translation_error_detection,
scientific_press_release, semantic_parsing_in_context_sparc,
semantic_parsing_spider, sentence_ambiguity,
similarities_abstraction, simple_arithmetic_json,
simple_arithmetic_json_multiple_choice,
simple_arithmetic_multiple_targets_json, simple_ethical_questions,
simple_text_editing, snarks, social_iqa, social_support,
sports_understanding, strategyqa, sufficient_information,
suicide_risk, swahili_english_proverbs,
swedish_to_german_proverbs, tellmewhy, temporal_sequences, tense,
timedial, understanding_fables, vitaminc_fact_verification,
what_is_the_tao, wino_x_german, winowhy, word_sorting,
word_unscrambling
```

B　評価指標の形式的な定義

タスクに含まれる問題数を n、i 番目の問題についてのモデルの真のスコアとその平均をそれぞれ $y_i, \bar{y}$ とし、予測されたスコアとその平均をそれぞれ $\hat{y}_i, \bar{\hat{y}}$ とする。また、タスク t におけるモデル m の真の順位を $r_{t,m}$、予測されたスコアに基づく順位を $\hat{r}_{t,m}$ とする。このとき、各指標は以下で定義される。

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\bar{y} - y_i)^2} \tag{13}$$

$$r = \frac{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)\left(\hat{y}_i - \bar{\hat{y}}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}\sqrt{\sum_{i=1}^{n}\left(\hat{y}_i - \bar{\hat{y}}\right)^2}} \tag{14}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y| \tag{15}$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y)^2 \tag{16}$$

$$\mathrm{Acc@0} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\mathbf{1}[r_{t,m} = \hat{r}_{t,m}] \tag{17}$$

$$\mathrm{Acc@3} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\mathbf{1}[|r_{t,m} - \hat{r}_{t,m}| \le 3] \tag{18}$$

C　学習時のハイパーパラメータ

最大 epoch は 25,000 epoch、バッチサイズは 512、seed は 42、損失関数は Huber loss に設定した。dev セットでの MSE を基準に、patience を 500 epoch とする early stopping を適用した。また、全手法で共通して、学習率のスケジューラとして PyTorch の `ReduceLROnPlateau` クラスを用いた。学習開始時の学習率は 0.5、patience は 250 epoch、減衰率は 0.5、最小学習率は 0.01 に設定した。NCF における潜在ベクトルの次元数は 4,096 とした。

D　SHAP 値の計算

学習済み予測器を

$$\hat{s}_{m,t} = f_\theta(m, t, e_1^t, e_2^t, e_3^t, e_4^t, e_5^t) \tag{19}$$

とし、各 e_i^t を 1 つの特徴量グループとみなす（$F = \{1, 2, 3, 4, 5\}$）。テスト集合 $\mathcal{D}_{\text{test}}$ 上の平均予測値

$$\bar{s}_{\text{test}} = \frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{(m,t)\in\mathcal{D}_{\text{test}}} f_\theta(m, t, e_1^t, e_2^t, e_3^t, e_4^t, e_5^t) \tag{20}$$

を説明対象とし、部分集合 $S \subseteq F$ に対する値関数を

$$v(S) = \frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{(m,t)\in\mathcal{D}_{\text{test}}} \mathbb{E}_{\tilde{e}\sim\mathcal{B}}\left[f_\theta(m, t, e_S^t, \tilde{e}_{F\setminus S})\right] \tag{21}$$

と定義する。このとき、特徴量 $i \in F$ の SHAP 値は

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!}\left(v(S\cup\{i\}) - v(S)\right) \tag{22}$$

で計算される。