インストラクションチューニングにおけるノイズ注入の分析と明示的な正則化法の提案

重藤優太郎　新保仁
千葉工業大学 STAIR Lab　理化学研究所 AIP
{shigeto,shimbo}@stair.center

概要

NEFTune は，インストラクションチューニングにおいてトークン埋め込みにランダムノイズを注入する手法であり，下流タスク性能の向上が報告されている．しかし，NEFTune が暗黙的に最適化している目的関数は，これまで明示的には示されていなかった．本研究では，標準的な二次近似に基づき，NEFTune におけるノイズ注入がヘッセ行列のトレースに基づく正則化と関係していることを示す．この関係に着想を得て，我々は対応する明示的な正則化を直接最適化する手法を提案する．具体的には，ヘッセ行列の代用として，トークン埋め込み空間およびロジット空間の Fisher 情報行列に基づく，二種類の計算形式によるサンプリングベースの推定手法を提案する．実験により，これらの手法が現実的な計算コストで最適化可能であることを確認した．

1　はじめに

インストラクションチューニングは，指示と応答のペアからなる教師データを用いて大規模言語モデルをファインチューニングし，人間の意図との整合性および下流タスク性能を向上させる手法である．

近年，インストラクションチューニング時にトークン埋め込みへランダムノイズを注入することで，下流タスク性能が向上することが報告されている [1]．この手法は NEFTune と呼ばれ，主にロバスト性の観点から説明されてきた一方で，ノイズ注入が暗黙的にどのような目的関数を最適化しているのかは明確ではない．

本研究では，NEFTune のノイズ注入を分析し，漸近的な設定の下で，それがトークン埋め込みに対するヘッセ行列に基づく正則化項として解釈できることを示す．この分析により，ノイズ注入による効果を明示的な正則化項として捉え直すことが可能となる．

これらの知見に基づき，本論文では，対応する正則化目的関数を直接最適化する手法を提案する．この正則化では，計算が困難なヘッセ行列の代替として，トークン埋め込み空間およびロジット空間の Fisher 情報行列を用い，サンプリングに基づく二種類の推定手法として定式化される．実験により，現実的な計算コストでの最適化が可能なことを確認した..

2　問題設定

本研究では，事前学習済み言語モデルに対するインストラクションチューニングを考える．

訓練データとして，トークン列 $Y = (y_1, \ldots, y_n)$ を用いる．このトークン列は，通常，プロンプトと応答を連結して構成される．学習では，各位置 i におけるトークン単位のクロスエントロピー損失の総和を最小化する．$\mathbf{x}_j \in \mathbb{R}^d$ を訓練トークン y_j の埋め込みベクトルとする．予測位置 i において，モデル f は，予測位置 i より前のトークン埋め込み列 $\mathbf{X}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$ を入力として，ロジット $\mathbf{z} = f(\mathbf{X}_{<i}) \in \mathbb{R}^{|\mathcal{Y}|}$ を出力する．ここで $\mathcal{Y}$ はトークンの語彙集合を表す．この損失項は $\ell(f(\mathbf{X}_{<i}), y_i)$ で与えられる．

以下の解析では，$j < i$ を満たす単一のトークン埋め込み $\mathbf{x} = \mathbf{x}_j$ が，予測位置 i における損失にどのような影響を与えるかを考察する．

記法の煩雑さを避けるため，$\mathbf{x}$ 以外の文脈埋め込みに関する f の明示的な依存関係を省略し，$f(\mathbf{x})$ を $f(\mathbf{X}_{<i})$ の略記として用いる．これは，他の文脈埋め込みを固定した上で，損失を $\mathbf{x}$ の関数として解析するためである．予測位置 i を固定し，$y = y_i$ と書くと，損失は $\ell(f(\mathbf{x}), y)$ と表される．

NEFTune　NEFTune [1] による学習では，トークン埋め込み $\mathbf{x}$ が用いられるたびに，独立同分布な一

様ノイズ $\boldsymbol{\epsilon} \sim \mathcal{U}(-a, a)$ をサンプリングし，$\mathbf{x}$ に加算する．その結果，最適化される損失は $\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y)$ となる．NEFTune における変更点は，埋め込みにノイズを注入する点のみに限定されており，それ以外の学習手順は通常のインストラクションチューニングと同一である．それにもかかわらず，NEFTune はノイズを加えない標準的なインストラクションチューニングと比較して，下流タスク性能の向上が報告されている．

3 ノイズ注入に関する分析

本研究では，NEFTune におけるノイズ注入下での期待損失を解析する．トークン埋め込み $\mathbf{x}$ にノイズを注入した後の損失に対して，$\mathbf{x}$ のまわりで二次のテイラー展開を行うと，以下を得る:

$$\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y) \simeq \ell(f(\mathbf{x}), y) + \nabla_{\mathbf{x}} \ell(f(\mathbf{x}), y)^{\top} \boldsymbol{\epsilon} + \frac{1}{2} \boldsymbol{\epsilon}^{\top} \mathbf{H}_{\mathbf{x}} \boldsymbol{\epsilon},$$

ここで，$\mathbf{H}_{\mathbf{x}} = \nabla_{\mathbf{x}}^2 \ell(f(\mathbf{x}), y)$ は，入力トークン埋め込みに関するクロスエントロピー損失のヘッセ行列を表す．次に，$\boldsymbol{\epsilon} \sim \mathcal{U}(-a, a)$ に関して期待値を計算し，整理すると以下の式が得られる（導出は付録 A に示す）．

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}} [\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y)] \simeq \ell(f(\mathbf{x}), y) + \frac{a^2}{6} \operatorname{Tr}(\mathbf{H}_{\mathbf{x}}). \quad (1)$$

この結果より，ノイズを加えた目的関数を最小化することは，漸近的には，元の損失関数にヘッセ行列のトレースに基づく正則化項を加えた目的関数の最小化と対応することがわかる．$\operatorname{Tr}(\mathbf{H}_{\mathbf{x}})$ は，$\mathbf{x}$ に関する損失関数の曲率を示す量であり，この項を抑制することは局所的な滑らかさを促進し，小さな摂動に対するロバスト性の向上につながる．

4 正則化

前節で示したノイズ注入とヘッセ行列に基づく正則化との関係（式 (1)）に基づき，本研究では，ノイズ注入の代替として，正則化項を明示的に含む目的関数の最適化を考える．しかしながら，ヘッセ行列に対する直接的な正則化は計算量の観点から現実的ではない．そこで本研究では，サンプリングに基づく二種類の正則化項を考えることで，実用的な最適化を可能にする．

4.1 トークンスコアベースペナルティ

提案する正則化では，モデルの出力確率分布 $\mathbf{p} = \operatorname{softmax}(f(\mathbf{x}))$ からサンプリングされたトークンインデックス $y' \sim \mathbf{p}$ に関するヘッセ行列 $\mathbf{H}_{\mathbf{x}} = \nabla_{\mathbf{x}}^2 \ell(f(\mathbf{x}), y')$ の期待値と Fisher 情報行列との間に成り立つ以下の関係を利用する:

$$\mathop{\mathbb{E}}_{y' \sim \mathbf{p}} [\mathbf{H}_{\mathbf{x}}] = \mathbf{F}_{\mathbf{x}}.$$

この Fisher 情報行列は以下で計算される．

$$\mathbf{F}_{\mathbf{x}} = \mathop{\mathbb{E}}_{y' \sim \mathbf{p}} \left[(\nabla_{\mathbf{x}} \log p_{y'}) (\nabla_{\mathbf{x}} \log p_{y'})^{\top} \right]$$

本研究では，式 (1) におけるヘッセ行列の代理 (surrogate) として Fisher 情報行列を用いる．なお，NEFTune と本提案法におけるヘッセ行列の違いについては，付録 B で議論する．本論文で提案するペナルティ（正則化）項は，以下である:

$$\begin{aligned} R &= \operatorname{Tr}(\mathbf{F}_{\mathbf{x}}) \\ &= \operatorname{Tr}\Big(\mathop{\mathbb{E}}_{y' \sim \mathbf{p}} \left[(\nabla_{\mathbf{x}} \log p_{y'}) (\nabla_{\mathbf{x}} \log p_{y'})^{\top} \right] \Big) \\ &= \mathop{\mathbb{E}}_{y' \sim \mathbf{p}} \left[\operatorname{Tr}\left((\nabla_{\mathbf{x}} \log p_{y'}) (\nabla_{\mathbf{x}} \log p_{y'})^{\top} \right) \right] \\ &= \mathop{\mathbb{E}}_{y' \sim \mathbf{p}} \left[\left\| \nabla_{\mathbf{x}} \log p_{y'} \right\|_2^2 \right]. \end{aligned}$$

この正則化項は，対数尤度の勾配 $\nabla_{\mathbf{x}} \log p_{y'}$ に依存しており，これは一般に（Fisher）スコアと呼ばれる量である．

実際の計算では，この期待値を厳密に求めることは困難であるため，独立同分布にサンプリングされた m 個のトークン $y'_i \sim \mathbf{p}, i = 1, \ldots, m$ による平均で代用する:

$$\hat{R}_{\text{token-F}} = \frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_{\mathbf{x}} \log p_{y'_i} \right\|_2^2. \quad (2)$$

4.2 ロジットスコアベースペナルティ

計算量をさらに削減するため，本研究では Fisher 重み付きヤコビ行列に基づくペナルティ [2] を検討する．

$$\mathbf{F}_{\mathbf{x}} = \mathbf{J}^{\top} \mathbf{F}_{\mathbf{z}} \mathbf{J},$$

この $\mathbf{J} = \nabla_{\mathbf{x}} \mathbf{z} \in \mathbb{R}^{|\mathcal{Y}| \times d}$ は入力トークン埋め込みに関するロジットのヤコビ行列であり，$\mathbf{F}_{\mathbf{z}} \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ はロジット空間における Fisher 情報行列である．ロジット空間における Fisher 情報行列の定義 ($\mathbf{F}_{\mathbf{z}} = \mathbb{E}_{y' \sim \mathbf{p}} \left[(\nabla_{\mathbf{z}} \log p_{y'}) (\nabla_{\mathbf{z}} \log p_{y'})^{\top} \right]$,) を用いると，

正則化項は以下となる．

$$R = \mathrm{Tr}\left(\mathbf{J}^\top \mathbf{F}_{\mathbf{z}} \mathbf{J}\right) = \mathop{\mathbb{E}}_{y' \sim \mathbf{p}} \left[\left\| \mathbf{J}^\top \nabla_{\mathbf{z}} \log p_{y'} \right\|_2^2 \right].$$

実際の計算では，サンプル平均を用いる．

$$\hat{R}_{\text{logit-F}} = \frac{1}{m} \sum_{i=1}^{m} \left\| \mathbf{J}^\top \nabla_{\mathbf{z}} \log p_{y'_i} \right\|_2^2.$$

この式はヤコビアンを明示的に含むため計算コストが高そうに見えるが，実際には，標準的な VJP（vector-Jacobian Product）計算を用いることで緩和できる．具体的には，クロスエントロピー目的関数に対して成り立つ以下の関係を利用する．

$$\nabla_{\mathbf{z}} \log p_{y'} = \mathbf{e}_{y'} - \mathbf{p},$$

これにより，以下の式が得られる．

$$\begin{aligned} \mathbf{J}^\top \nabla_{\mathbf{z}} \log p_{y'_i} &= \mathbf{J}^\top (\mathbf{e}_{y'_i} - \mathbf{p}) \\ &= \nabla_{\mathbf{x}} \left(f(\mathbf{x})^\top (\mathbf{e}_{y'_i} - \mathbf{p}) \right). \end{aligned}$$

この最後の式では確率ベクトル $\mathbf{p}$ を定数として扱い，計算グラフから切り離している（detach している）．これによって，$\mathbf{J}$ を明示的に構成することなく正則化項を計算できる．

$$\hat{R}_{\text{logit-F}} = \frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_{\mathbf{x}} \left(f(\mathbf{x})^\top (\mathbf{e}_{y'_i} - \mathbf{p}) \right) \right\|_2^2 \tag{3}$$

なお，実験では，式 (2) と 式 (3) ともに，注意マスクや応答部分のみを対象とするラベルマスクなど，学習バッチから導出される二値マスクを用いて，正則化をトークンの部分集合にのみ適用する．このマスクは学習可能なパラメータではなく，正則化の適用範囲を有効なトークンに限定するためにのみ用いられる．

5 関連研究

ノイズ注入を用いた学習が正則化として機能することは，古くから知られている [3, 4, 5]．それらの研究においては，テイラー近似を利用し，入力へのノイズ注入が，ノイズを加えない場合の損失関数およびヘッセ行列やヤコビ行列に基づく正則化と関連することが示されている．

BERT のファインチューニングにおいては，Hua ら [6] は，中間層において，ノイズを加えた埋め込み表現とノイズを加えていない埋め込み表現の差を正則化項として用いる手法を提案した．実験において高い性能を示すとともに，リプシッツ連続性および Tikhonov 正則化との関係を明らかにしている．

Yadav [7] は，NEFTune におけるノイズ分布の違いが与える影響を分析した．適切にスケールを調整した一様分布ノイズとガウス分布ノイズは，高次元設定において類似した振る舞いを示すことを報告している．さらに，対称なノイズ注入法を提案し，NEFTune を上回る性能を示している．

6 実験

提案手法の有効性を検証するため，インストラクションチューニングにおいて，token-F（式 (2)）および logit-F（式 (3)）の 2 種類の定式化を適用し，複数の下流ベンチマークで評価する。

6.1 実験設定

本研究では，ベースモデルとして Llama-3.1-8B [8] を用いる[1)]．学習は，Tülu 3 データセット [9] のフィルタリング済みバージョンを用いて実施した．本データセットは，約 86.6 万件の対話で構成された学習データである．学習のハイパーパラメータは，NEFTune におけるノイズスケーリング係数および提案手法における正則化係数を除き，Tülu 3 のインストラクションチューニング設定と同一のものを利用した．提案法の正則化係数は，学習中に線形ウォームアップを適用した．ノイズ係数，正則化係数，ウォームアップステップは，予備実験により調整した．式 (2) と式 (3) のサンプル数 m は 1 に設定した．ハイパーパラメータの詳細については，付録 C に記載する．

本実験は，通常のクロスエントロピー損失で学習したベースライン，トークン埋め込みにノイズを注入し，クロスエントロピー損失で学習した NEFTune，クロスエントロピー損失に式 (2) を組み合わせた提案法（token-F），クロスエントロピー損失に式 (3) を組み合わせた提案法（logit-F），の 4 種類を比較する．

モデル性能を包括的に評価するため，多様なベンチマークを用いて評価を行う．具体的には，MMLU [10]，MixEval [11]，AlpacaEval [12, 13]，MT-Bench [14]，IFEval [15]，TruthfulQA [16] を用いた．異なる乱数シードを用いて独立に 3 回学習を行い，それらの評価結果の平均値および標準偏差を実験結果として報告する．

手法	MMLU	MixEval	AlpacaEval	MT-Bench	IFEval	TQA
ベースライン	0.611 ± 0.002	0.598 ± 0.004	9.128 ± 0.234	7.058 ± 0.055	0.684 ± 0.003	**0.320** ± 0.002
NEFTune	**0.617** ± 0.000	0.598 ± 0.003	9.183 ± 0.764	7.046 ± 0.166	**0.690** ± 0.003	0.318 ± 0.004
提案法 (token-F)	0.612 ± 0.002	**0.605** ± 0.001	**10.018** ± 0.317	**7.073** ± 0.118	0.685 ± 0.009	**0.320** ± 0.001
提案法 (logit-F)	0.612 ± 0.002	0.595 ± 0.006	9.518 ± 0.527	7.016 ± 0.145	0.689 ± 0.006	0.319 ± 0.003

表 1　実験結果．異なるシードで 3 回学習を行い，それらの評価結果の平均値および標準偏差を示している．

手法	MMLU	MixEval	AlpacaEval	MT-Bench	IFEval	TQA
token-F	**0.622** ± 0.002	**0.603** ± 0.004	8.052 ± 0.528	**6.860** ± 0.144	0.576 ± 0.006	**0.347** ± 0.001
ウォームアップなし	0.621 ± 0.001	0.602 ± 0.003	**8.097** ± 0.324	6.728 ± 0.140	**0.579** ± 0.005	0.344 ± 0.002
一様サンプリング	0.299 ± 0.079	0.311 ± 0.043	1.258 ± 0.796	3.138 ± 1.154	0.274 ± 0.039	0.286 ± 0.027
logit-F	**0.625** ± 0.001	0.602 ± 0.007	7.804 ± 0.982	**6.744** ± 0.106	0.582 ± 0.003	0.344 ± 0.002
ウォームアップなし	0.626 ± 0.002	**0.610** ± 0.006	**8.139** ± 0.627	6.663 ± 0.022	**0.585** ± 0.009	0.345 ± 0.002
一様サンプリング	0.618 ± 0.001	0.605 ± 0.003	7.472 ± 0.583	6.705 ± 0.066	0.583 ± 0.006	**0.349** ± 0.002

表 2　アブレーション実験．学習データ量を 10%に削減した条件で学習を行なった．

6.2　実験結果

表 1 に実験結果を示す．token-F を用いた明示的正則化は，NEFTune と同等またはそれ以上の性能となった．特に AlpacaEval および MT-Bench では最も高いスコアを得た（差はわずかだが MixEval でもトップ）．MMLU および IFEval においては NEFTune が最高スコアを記録したものの，提案手法との差は小さい．安全性ベンチマークである TruthfulQA においては，すべての手法が同程度の性能を示しており，提案する正則化が（他の手法と比較して）安全性を損なっていないことが確認できる．全体として，3 回の独立した学習を行なった結果を報告している．スコアの標準偏差は，ほとんどのベンチマークにおいては小さい値となっているが，AlpacaEval および MT-Bench では大きな値となった．これらのベンチマークにおいては，提案手法は NEFTune より小さな標準偏差になっており，より安定した性能が得られていると思われる．

6.3　アブレーション実験

提案した正則化手法の各要素の寄与を検証するため，アブレーション実験を行なった．実験の効率化のため，アブレーション実験では，学習データの 10%のみを用いて学習を行なった．表 2 に実験結果を示す．正則化係数に対する線形ウォームアップスケジュール（表中の「ウォームアップなし」）は除去しても性能への影響は小さく，本設定においては，ウォームアップは必須ではないことが確認できた．一方で，モデル分布のサンプリングを一様サンプリングに置き換えた場合には（表中の「一様サンプリング」），性能が大きく低下することが確認された．この結果は，モデルの出力分布にしたがったサンプリングが重要であることを示している．これらの傾向は，token-F ベースおよび logit-F ベースの双方の計算において一貫して観測された．

7　まとめ

インストラクションチューニングにおけるノイズ注入は，簡便かつ有効な手法として知られている一方で，それによって暗黙的に学習されている目的関数は明示的には示されてこなかった．本研究では，NEFTune に代表されるノイズ注入手法を二次近似の枠組みで解析し，それがヘッセ行列のトレースに基づく暗黙的な正則化として解釈できることを示した．

この視点に基づき，本研究では Fisher 情報行列に基づく正則化を定式化し，明示的に正則化の最適化が行えることを示した．大規模な出力空間においても現実的な時間で計算を行うために，トークンスコアに基づく計算法（token-F）とロジットスコアに基づく計算法（logit-F）の 2 種類の方法を提示した．

実験において，提案手法は埋め込みへのノイズ注入を行うことなく，複数の標準的なベンチマークにおいて NEFTune と同等またはそれ以上の性能を達成できることを確認した．

1）　https://huggingface.co/meta-llama/Llama-3.1-8B

謝辞

本研究は JSPS 科研費 JP24K20842,JP24K02963 の助成を受けたものです。

参考文献

[1] Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. NEFTune: Noisy embeddings improve instruction finetuning. In **The Twelfth International Conference on Learning Representations**, 2024.

[2] James Martens. New insights and perspectives on the natural gradient method. **Journal of Machine Learning Research**, Vol. 21, pp. 1–76, 2020.

[3] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. **Neural Computation**, Vol. 7, No. 1, pp. 108–116, 1995.

[4] Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. Adding noise to the input of a model trained with a regularized objective. **arXiv preprint arXiv:1104.3250**, 2011.

[5] Stefan Wager, Sida Wang, and Percy Liang. Dropout training as adaptive regularization. In **Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1**, NIPS'13, p. 351–359, Red Hook, NY, USA, 2013. Curran Associates Inc.

[6] Hang Hua, Xingjian Li, Dejing Dou, Chengzhong Xu, and Jiebo Luo. Noise stability regularization for improving BERT fine-tuning. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, **Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pp. 3229–3241, Online, June 2021. Association for Computational Linguistics.

[7] Abhay Yadav. Understanding and improving noisy embedding techniques in instruction finetuning. In **Second Conference on Language Modeling**, 2025.

[8] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. **arXiv preprint arXiv:2407.21783**, 2024.

[9] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In **Second Conference on Language Modeling**, 2025.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In **International Conference on Learning Representations**, 2021.

[11] Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. MixEval: Deriving wisdom of the crowd from LLM benchmark mixtures. In **The Thirty-eighth Annual Conference on Neural Information Processing Systems**, 2024.

[12] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator for instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

[13] Yann Dubois, Percy Liang, and Tatsunori Hashimoto. Length-controlled AlpacaEval: A simple debiasing of automatic evaluators. In **First Conference on Language Modeling**, 2024.

[14] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In **Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track**, 2023.

[15] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. **arXiv preprint arXiv:2311.07911**, 2023.

[16] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics.

A　ノイズ注入とヘッセ行列

入力トークンの埋め込みを $\mathbf{x} \in \mathbb{R}^d$ に，ノイズ $\boldsymbol{\epsilon} \in \mathbb{R}^d$ を注入することを考える．トークン埋め込みにノイズを加算した損失関数を $\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y)$ とした場合，入力トークン埋め込み $\mathbf{x}$ の周りで二次のテイラー展開を行うと，以下となる．

$$\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y) \simeq \ell(f(\mathbf{x}), y) + \nabla_{\mathbf{x}}\ell^{\top}\boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^{\top}\mathbf{H}_{\mathbf{x}}\boldsymbol{\epsilon}, \quad (4)$$

この $\mathbf{H}_{\mathbf{x}} = \nabla^2_{\mathbf{x}}\ell(f(\mathbf{x}), y)$ は入力トークン埋め込みに関するヘッセ行列である．

次に，平均がゼロのノイズ $\boldsymbol{\epsilon}$ に関して，期待値を計算することで，次式が得られる．

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}[\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y)] \simeq \ell(f(\mathbf{x}), y) + \frac{1}{2}\mathrm{Tr}\Big(\mathbf{H}_{\mathbf{x}} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\top}]\Big). \quad (5)$$

NEFTune においては，ノイズは一様分布を仮定している: $\boldsymbol{\epsilon}_i \sim \mathrm{Unif}(-a, a)$, そのため，$\mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}$ and $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\top}] = (a^2/3)\mathbf{I}$, となり，損失関数の期待値は以下となる

$$\mathop{\mathbb{E}}_{\boldsymbol{\epsilon}}[\ell(f(\mathbf{x}+\boldsymbol{\epsilon}), y)] \simeq \ell(f(\mathbf{x}), y) + \frac{a^2}{6}\mathrm{Tr}(\mathbf{H}_{\mathbf{x}}). \quad (6)$$

この式から，NEFTune におけるノイズ注入は，二次の近似の下で，ヘッセ行列のトレースに基づく正則化を行なっていると考えられる．

B　NEFTune と提案法におけるヘッセ行列の比較

ここでは，NEFTune により暗黙的に導入されるヘッセ行列と提案法で用いるヘッセ行列の期待値との違いを明示する．まず，$\mathbf{z} = f(\mathbf{x})$ をモデルの出力ロジットとし，$\mathbf{p} = \mathrm{softmax}(\mathbf{z})$ を対応する出力確率分布とする．

式 (1) におけるヘッセ行列 $\mathbf{H}_{\mathbf{x}} = \nabla^2_{\mathbf{x}}\ell(f(\mathbf{x}), y)$ は，連鎖律により次のように分解される．

$$\mathbf{H}_{\mathbf{x}} = \mathbf{J}^{\top}\mathbf{H}_{\mathbf{z}}\mathbf{J} + \sum_{j=1}^{|\mathcal{Y}|}\frac{\partial \ell}{\partial z_j}\nabla^2_{\mathbf{x}} z_j, \quad (7)$$

ここで $\mathbf{J} = \nabla_{\mathbf{x}}\mathbf{z}$ は入力トークン埋め込みに関するロジットのヤコビ行列である．

一方，提案する正則化では，このヘッセ行列の代理としてモデルの出力確率分布 $\mathbf{p}$ からサンプリングされたトークンインデックス $\mathbf{p} = \mathrm{softmax}(\mathbf{z})$ に関するヘッセ行列 $\mathbf{H}_{\mathbf{x}} = \nabla^2_{\mathbf{x}}\ell(f(\mathbf{x}), y')$ の期待値を用いる

$$\mathop{\mathbb{E}}_{y'\sim\mathbf{p}}[\mathbf{H}_{\mathbf{x}}] = \mathbf{J}^{\top}\mathbf{H}_{\mathbf{z}}\mathbf{J}. \quad (8)$$

これらの式より，NEFTune (式 (7)) と提案法 (式 (8)) の異なりは，式 (7) に現れる第 2 項の有無にあることが分かる．

C　ハイパーパラメータ

本研究では，学習データとしてフィルタリングされた Tülu3 を利用した．[2] エポック数，学習率，バッチサイズなどの基本的なハイパーパラメータは，既存の Tïlu 3 インストラクションチューニングの設定を採用した[3]．NEFTuen のノイズスケール，提案した正則化をクロスエントロピー損失と組み合わせる際の重み係数，この係数のウォームアップステップ数は，予備実験により個別に調整を行なった．予備実験は，学習データセットの 1%（Tülu 3 は複数のデータの組み合わせになっているので，stratified sampling を行なった）を用いてモデルの学習を行い，MT-Bench により評価した．この予備実験で最も高い性能を示したハイパーパラメータを本番実験で採用した．具体的には，NEFTune のノイズスケールは 5.0，token-F の係数は 2×10^{-6} でウォームアップステップが 600，logit-F の係数は 1×10^{-6} でウォームアップステップが 800，を用いた．

式 (2) と式 (3) のサンプル数 m は 1 に設定した

2) https://huggingface.co/datasets/allenai/tulu-3-sft-olmo-2-mixture-0225

3) https://github.com/allenai/open-instruct/blob/main/docs/tulu3.md#llama-31-tulu-3-8b-sft-reproduction