日本語タスクにおける LLM を用いた疑似学習データ生成の検討

藤井 巧朗 [1]　勝又 智 [2]
[1] 横浜国立大学大学院理工学府　[2] 株式会社レトリバ
tkr.fujii.ynu@gmail.com　satoru.katsumata@retrieva.jp

概要

大規模言語モデル (LLM) を用いることで，学習データが少量の状況においても高い性能が期待できることが報告されている．しかし，LLM はパラメータ数が多いため，推論コストが大きいという課題が存在する．そこで，本研究では 6 個の下流タスクにおいて，LLM から疑似学習データを生成し，そのデータで小規模モデルを学習する JapaGen が LLM にどの程度迫れるかを検証した．その結果，フォーマルなテキストを入力とする分類タスクで有効であることを明らかにした．

1　はじめに

近年，大規模言語モデル (Large Language Model; LLM) に関する研究が盛んに行われている．Brown ら [1] は GPT-3 を開発し，LLM にタスクの説明文と数件の教師データを入力することで，LLM のパラメータ更新を行うことなく高い精度を達成した．日本語においても，同様に高い性能を達成できる場合もあることが報告されている[1)]．

このように LLM は従来の機械学習モデルと比べて優れた特徴を持つ一方で，パラメータ数が多く，推論時には GPU リソースの制約などの運用コストが高い．この課題に対する研究として，LLM に疑似学習データを生成させ，そのデータを用いて小規模モデルを学習させるもの (**広義の知識蒸留** と呼称) が存在する．Ye ら [2] は英語の様々なタスクに対して，GPT-2 XL [3] を利用した Nucleus sampling [4] による多様な疑似データを作成した．その疑似データで DistilBERT [5] を訓練することで，GPT-2 XL より優れた性能となることを報告した．その後，より精巧な疑似データを生成するための研究 [6, 7, 8, 9] や，生成した疑似データによる小規模モデルの学習が有効なタスクを分析した研究 [10] などが存在するが，日本語での検証は我々の知る限り存在しない．

本研究では日本語の複数のタスクについて，GPT-3.5 を利用した疑似データ生成による広義の知識蒸留 (**JapaGen** と呼称) を検証した．具体的には，JGLUE 中の分類タスクと JSTS，ニュースピック分類，SNS テキスト分類について検証を実施した．検証として，Ye ら [2] のような Decoding 戦略による多様な疑似データ生成を試みたが，多様な疑似データの生成は困難であることがわかった．

そこで，本研究ではタスク知識を利用した疑似データ生成手法 **Knowledge-Assisted Data Generation (KADG)** を提案する．この手法は Ye ら [2] のように Decoding 戦略に依存せず，LLM への入力にタスク知識を導入することで多様なテキストを生成する．

貢献．　本研究では，(1) 日本語タスクにおいて，LLM を用いた疑似データ生成による広義の知識蒸留 JapaGen を検証した．(2) さらに，多様な疑似データ生成を行うためにタスク知識を Prompt に含める KADG を提案した．(3) 実験の結果，JapaGen はフォーマルなテキストを入力とするタスクにおいて Zero-Shot Prompting や Few-Shot の性能を上回った．(4) KADG によって疑似データの多様性が向上し，教師データ (Gold) の分布に近づいたが，必ずしも下流タスク性能が向上するとは限らないことが分かった．

2　LLM を用いた従来の推論

LLM は従来の機械学習モデルと比べて，少量の学習データでも高い性能を発揮することが知られている．Brown ら [1] は大量のテキストデータで学習された LLM (GPT-3) を作成し，このモデルにタスクの説明文と数件の教師データのみを入力することで，高い性能を達成した．この入力というのは，LLM のパラメータ更新は行わず，LLM がテキストを生成する際の条件部として入力すること (Prompting) を指しており，この入力に対する出力テキスト Predict

1) https://github.com/Stability-AI/lm-evaluation-harness/tree/jp-stable

の確率分布は下記のように表現できる．

$$\text{Predict} \sim \text{Prob}_{\text{LLM}}(\cdot|\text{Prompt}) \tag{1}$$

Prompt が LLM に入力するテキストである．また，彼らは Prompt に入力する教師データの件数をもとに，数件であれば Few-Shot，一件も含まず，タスクの説明文のみを入力する際は Zero-Shot と呼称した．本研究でも同様に使用する教師データの件数をもとに Zero/Few-Shot の区別を行う．Zero/Few-Shot の利点は，学習データの効率化に加え，学習データの収集コストを下げることにも繋がる．一方で，LLM はパラメータ数が多いため，推論時の運用コストが大きくなる．本研究では Zero-Shot と同様の教師データ件数を利用しつつ，推論コストを下げるために，LLM から疑似データを生成し，そのデータを用いて小規模モデルを学習することを検討する．

3 疑似データを用いた推論

Ye ら [2] は簡潔なタスクとラベルの説明文を LLM に入力することで，Zero-Shot 設定での疑似データ生成を実施した．本研究でも，同様の入力を日本語タスクに向けて作成し，疑似データ生成を行った．

3.1 Zero-Shot 疑似データ生成

本研究では，単一テキスト分類タスクまたは 2 文間の関係性推定タスクについて LLM を用いた疑似データ生成 **JapaGen** を行った．

単一テキスト分類タスク． あるクラス $y_c \in Y; |Y| = C$ に対応する疑似データ $\tilde{D}_{y_c} = \{(\tilde{x}_{c,j}, y_c)\}_{j=1}^{M}$ を LLM を用いて作成する．ただし，M はラベルあたりの疑似データ数を指す．この疑似データテキスト $\tilde{x}_{c,j}$ は次式 (2) により生成する．

$$\tilde{x}_{c,j} \sim \text{Prob}_{\text{LLM}}(\cdot|\text{T}(\text{task}, y_c)) \tag{2}$$

ただし，$\text{T}(\text{task}, y_c)$ はタスク task とクラス y_c に関して説明を行う Prompt を作成する関数である．他のクラスも同様に疑似データを作成し，疑似データセット $\tilde{D} = [\tilde{D}_{y_1}, \tilde{D}_{y_2}, \ldots, \tilde{D}_{y_C}]$ を作成する．

2 文間の関係推定タスク． 最初に，1 文目 $\tilde{x}^1_{c,j}$ を前述した 1 文入力と同様に生成する．

$$\tilde{x}^1_{c,j} \sim \text{Prob}_{\text{LLM}}(\cdot|\text{T}(\text{task})) \tag{3}$$

ただし，T(task) ではクラスは使用せず，タスク task のみから Prompt を作成している．2 文目 $\tilde{x}^2_{c,j}$ は，先ほどの入出力 T(task) および $\tilde{x}^1_{c,j}$ を LLM に入力して作成する．

$$\tilde{x}^2_{c,j} \sim \text{Prob}_{\text{LLM}}(\cdot|\text{T}(\text{task}), \text{T}(\text{task}, \tilde{x}^1_{c,j}, y_c)) \tag{4}$$

ただし，$\text{T}(\text{task}, \tilde{x}^1_{c,j}, y_c)$ は 1 回目で生成されたテキストとクラスの説明文に基づいて Prompt を作成する．このように，クラス y_c に関する M 個のデータ $\tilde{D}_{y_c} = \{(\tilde{x}^1_{c,j}, \tilde{x}^2_{c,j}, y_c)\}_{j=1}^{M}$ を作成する．これを全クラスに実施し，疑似データセット $\tilde{D}$ を作成する．

3.2 多様な疑似データ生成

前述の疑似データ生成手法は式 (1) に基づいて，Top-p などの生成パラメータによって調整した分布から生成している．我々の事前実験では，Ye ら [2] と同様の方法で多様な日本語テキストの生成を試みたが，いくつかのタスクで類似したテキストが多く生成された (§4.2)．そこで，本研究ではタスクの知識を Prompt に含めることによって多様なテキストの生成を試みる．

タスクごとにそのタスクに関する内容語集合 E_{task} を人手で作成し，その中から内容語 $d \in E_{\text{task}}$ をランダムに選択して Prompt を作成する．

$$d \sim E_{\text{task}} \tag{5}$$

$$\tilde{x}_{c,j} \sim \text{Prob}_{\text{LLM}}(\cdot|\text{T}(\text{task}, y_c, d)) \tag{6}$$

$\text{T}(\text{task}, y_c, d)$ はタスク task の説明文と，クラス y_c の説明文，内容語 d に関連する Prompt を生成している．この処理を §3.1 と同様に全てのクラスに対して実施し，疑似データセット $\tilde{D}$ を作成する．内容語を考慮した疑似データ生成方法を **Knowledge-Assisted Data Generation (KADG)** と呼ぶ．KADG は教師データを使用しないという点では Zero-Shot と見なすこともできるが，タスク知識を利用しているため，これまでの Zero-Shot と同一と述べるのは適当ではないと考える．そこで本研究では Zero-Shot に対して KADG を適応した場合の問題設定を **Zero-Shot*** と呼称し，Zero-Shot と区別する．

4 実験

本研究では様々な日本語タスクで，広義の知識蒸留が LLM の Zero-Shot Prompting や Few-Shot 手法にどの程度迫るか検証する．

比較手法． LLM を用いた疑似データ生成手法として，§3 で述べた Zero-Shot および Zero-Shot*の 2 手法を比較する．また，§2 で述べた Zero-Shot および Few-Shot での Prompting 及び，Few-Shot データのみで学習したモデルの結果を比較する．上限値とし

表 1 6 個の日本語タスクにおける各手法の性能比較．各値は 5 シード値の平均 (標準偏差) を表す．

Method		MARC-ja Acc.	JSTS Pearson / Spearman	JNLI Acc.	JCoLA Mcc.	News Acc.	COVID-19 Acc.	Avg.
Fully Supervised (Gold)		95.78(0.1)	91.42(0.3) / 87.47(0.5)	90.19(0.4)	40.62(1.2)	95.75(0.4)	78.49(0.3)	82.82
Few-Shot	Prompting	97.38(0.2)	84.06(0.9) / 78.50(2.0)	35.86(5.3)	26.00(2.9)	44.82(2.9)	65.44(3.4)	61.72
	Fine-Tuning	61.57(8.5)	16.92(12.8) / 14.80(11.3)	37.72(13.4)	-0.85(3.5)	51.98(5.3)	42.24(9.4)	37.40
Zero-Shot	Prompting	94.82(0.2)	70.98(0.3) / 68.53(0.6)	41.53(1.0)	24.76(1.2)	40.27(1.3)	62.76(0.6)	57.66
	JapaGen	77.76(5.4)	72.96(0.0) / 72.47(0.1)	46.49(1.5)	18.17(1.7)	57.37(2.1)	34.36(6.4)	54.23
Zero-Shot*	w/ KADG	83.24(6.0)	71.12(1.5) / 71.49(1.2)	46.04(0.4)	-	-	-	-

て各データセット内の Gold データをそのまま利用した場合 (Fully Supervised) との比較も行う．

実験データ． JGLUE [11] のうち MARC-ja, JSTS, JNLI および JCoLA の 4 タスクを用いた．また，多様なドメインで評価を行うため，トピック分類タスク (News) および SNS における事実性分類 (COVID-19) を用いた[2)]．JGLUE は test セットが公開されていないため，train セットを 8:2 に分割して学習し，dev セットを test セットとして評価を行った．

実装． LLM は `gpt-3.5-turbo-0613` を使用し，Fine-Tuning 時は，モデルに日本語 BERT を用いた．疑似データ生成におけるパラメータおよび各タスクにおける学習パラメータは付録 §A.3 に示す．また，Fine-Tuning に関しては 5 回シード値を変えて実施しており，実験結果はその平均を報告している．

4.1 広義の知識蒸留に関する結果

各手法における下流タスク性能を表 1 に示す．

Zero-Shot での比較． JapaGen は Prompting の性能を JSTS，JNLI および News の 3 タスクで上回り，MARC-ja，JCoLA および COVID-19 の 3 タスクで下回った．前者 3 タスクのテキストはフォーマルである一方，後者 3 タスクのうち MARC-ja と COVID-19 はインフォーマルであるという特徴がある[3)]．また，JCoLA はタスクの性質上，文法的に誤ったテキストを生成する必要があるが，そのような Prompt 設計は困難であった．以上より，Zero-Shot 設定において JapaGen はフォーマルなテキストを入力とするタスクで有効であることが示唆される．

JapaGen で用いた BERT モデルは約 110M であるにもかかわらず[4)]，全タスク平均において，JapaGen は Prompting と比較してわずか 3.43 しか下回らない．したがって，モデルサイズと性能のトレードオフを考慮すると，JapaGen は効果的であると言える．

Few-Shot との比較． JapaGen は Few-Shot Fine-Tuning と比較して，全タスク平均で 24.37 ポイントも上回り，各タスクでは COVID-19 以外の 5 タスクで上回った．これは，同じモデルサイズである場合，JapaGen (Zero-Shot) は Few-Shot の性能を大きく上回ることを意味する．Few-Shot Prompting との比較では，JapaGen は JNLI および News の 2 タスクで性能を上回る．この 2 タスクは Zero-Shot Prompting での結果と一致する．JSTS で性能が下回ったのは，類似度のみから疑似データを生成するのは困難であるためだと考えられる．具体的には，疑似データ生成時に付与するクラスは {0.,1.,2.,3.,4.,5.} の 5 個としたが，実際には 2 文の類似度は 0. ~ 5. の実数値であり，小数点以下の類似性を正確に表現できない．以上から，Zero-Shot JapaGen はフォーマルなテキストの分類タスクにおいて効果的である．これらのタスクでは，モデルサイズが大きく，Gold データ数も多い Few-Shot Prompting をも上回った．

Fully Supervised との比較． JapaGen で生成した疑似データは Gold データと同程度かそれ以上のデータサイズなのにもかかわらず，JapaGen は全 6 タスクで Fullly Supervised の性能を大きく下回った．これは LLM により生成した疑似データの分布は Gold の分布と大きく異なることを示唆する．次節で Gold と疑似データの分布について分析する．

KADG における性能． KADG は MARC-ja のみ JapaGen の性能を上回った．次節でこの分析を行う．

4.2 分析

出現トークン数分布． Gold および疑似テキストの出現トークンとその出現数のヒストグラム分布を図 1 に示す．また，両ヒストグラムの重複度を表

2) タスクの詳細は付録 §A.1 に，各タスクの評価指標は §A.2 に記載．また，KADG に関しては MARC-ja，JSTS および JNLI に対して実施した．
3) テキストスタイルにおいてフォーマルかどうかを表す．
4) `gpt-3.5-turbo` のパラメータ数は公開されていないが，公開されている LLM は数 ~ 数百 B のモデルサイズである．

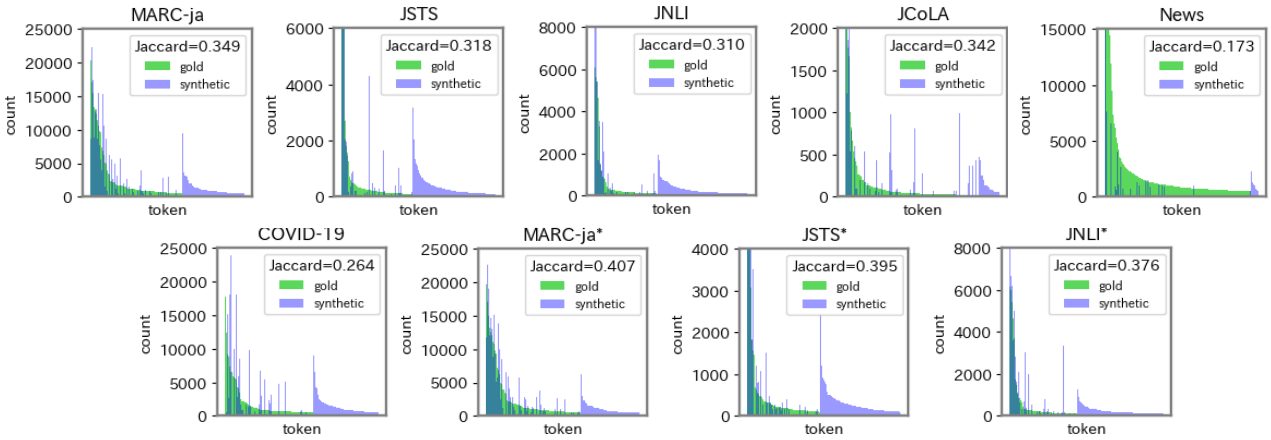

図 1　Gold と疑似データにおける出現トークンと出現数の分布．Jaccard 係数はヒストグラムの一致度を示し，値が高いほど一致することを示す．ヒストグラムには 500 回以上出現するトークンを表示した．*は KADG を表す．

表 2　各種データの多様性．値は小さい方が多様である．MAR. は MARC-ja，COV. は COVID-19 を指す．

Method	Self-BLEU (%)					
	MAR.	JSTS	JNLI	JCoLA	News	COV.
Gold	40.53	72.93	72.94	56.66	62.97	43.14
JapaGen	91.67	74.89	69.97	65.80	79.90	84.31
w/ KADG	84.97	76.12	73.13	-	-	-

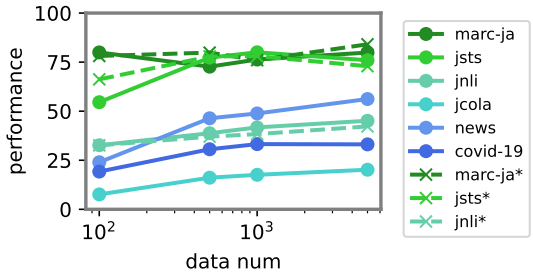

図 2　データ数に対する下流タスク性能の推移．

す重み付き Jaccard 係数 [12] も算出した．

全タスクに共通して疑似データにのみ出現するトークンが多く存在することが分かる．News は他タスクと比較して Jaccard 係数が小さいのにもかかわらず JapaGen が Few-Shot の性能を上回ることから，Gold と疑似データの分布の一致は下流タスク性能の十分条件にすぎないと言える．また，KADG によって Jaccard 係数は増加したことから，KADG は疑似データを Gold 分布に近づけるのに有効である．

多様性分析．　Gold と疑似データの多様性を分析する．多様性の測定には Self-BLEU (*cf.* §A.5) [13] を用いた．Self-BLEU は値が小さいほど多様であることを示す指標である．結果を表 2 に示す．

表 2 より，MARC-ja と COVID-19 の 2 タスクで，JapaGen の多様性は約 1/2 も低い．一方，JSTS, JNLI, JCoLA および News では，疑似データは Gold データに近い多様性を保持している．次に，JapaGen と KADG の比較を行う．MARC-ja のみ KADG の多様性が JapaGen を上回る．表 1 の結果と合わせると，KADG による多様化が MARC-ja の精度向上に寄与していると考えられる．Ye ら [2] は多様性を上げるとラベル正確性が低下し，下流タスク性能を悪化させてしまうと報告しているが，KADG で作成した疑似データのラベル正確性を確認したところ，この正確性を低下させることなく多様性を向上させていることがわかった (§A.6)．以上から，KADG によって多様性を増加する場合と多様性を制限してしまう場合があるが，ラベル正確性は維持できるため，KADG はその前者の場合に有効であると言える．

データ数のスケーリング．　疑似データ数による下流タスク性能の推移を図 2 に示す．データ数は {100,500,1000,5000} である．データ数に対して多くの下流タスク性能は増加傾向にある．

5　おわりに

本研究では，LLM によって生成した疑似データで学習した小規模モデルが 6 個の日本語タスクにおいて LLM にどれだけ迫るかを検証した．その結果，フォーマルなテキストを入力とする分類タスクにおいて，Zero-Shot および Few-Shot Prompting を上回った．また，タスク知識を Prompt に導入する KADG を提案し，疑似データが Gold データの分布に近づき，MARC-ja で多様性と性能が向上した．

限界．　本研究では簡潔な Prompt を用いたが，Prompt 設計により精度向上が期待される．また，疑似データ生成に `gpt-3.5-turbo` より高性能な LLM を使用することも精度向上に繋がると考えられる．

謝辞

この成果は, 国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO) の委託業務 (JPNP18002) の結果得られたものです.

参考文献

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. ZeroGen: Efficient zero-shot learning via dataset generation. In **Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing**, 2022.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In **International Conference on Learning Representations**, 2020.

[5] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019.

[6] Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. ProGen: Progressive zero-shot dataset generation via in-context feedback. In **Findings of the Association for Computational Linguistics: EMNLP 2022**, 2022.

[7] Himanshu Gupta, Kevin Scaria, Ujjwala Anantheswaran, Shreyas Verma, Mihir Parmar, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. TarGEN: Targeted data generation with large language models. **arXiv preprint arXiv:2310.17876**, 2023.

[8] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. In **Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track**, 2023.

[9] John Chung, Ece Kamar, and Saleema Amershi. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. In **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, 2023.

[10] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic data generation with large language models for text classification: Potential and limitations. In **Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing**, 2023.

[11] Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. JGLUE: Japanese general language understanding evaluation. In **Proceedings of the Thirteenth Language Resources and Evaluation Conference**, 2022.

[12] Yue Yu, Yuchen Zhuang, Rongzhi Zhang, Yu Meng, Jiaming Shen, and Chao Zhang. ReGen: Zero-shot text classification via training data generation with progressive dense retrieval. In **Findings of the Association for Computational Linguistics: ACL 2023**, 2023.

[13] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In **The 41st international ACM SIGIR conference on research & development in information retrieval**, 2018.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In **Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics**, 2002.

A 付録

A.1 データセット・タスク

MARC-ja. 商品レビューに対してポジティブかネガティブかを分類.

JNLI / JSTS. JNLI は 2 文の含意関係 (含意 / 矛盾 / 中立) を，JSTS は 2 文の類似度を 0. ~ 5. で推定.

JCoLA. 文が統語的に正しいかを判定.

COVID-19. COVID-19 に関するツイートの事実性 (一般事実 / 個人事実 / 意見 / 感想) を推定[5].

News. Livedoor News のニュース記事を 9 個のカテゴリ (トピック / Sports Watch / IT ライフハック / 家電チャンネル / MOVIE ENTER / 独女通信 / エスマックス / livedoor HOMME / Peachy) に分類[6].

A.2 評価指標

Accuracy. 全推論結果のうち正解した割合.

Pearson. 予測値と正解の線形相関係数.

Spearman. 予測値と正解の順位情報のみに基づく相関係数.

Matthews Correlation Coefficient. クラス不均衡を考慮した分類性能の指標.

A.3 実装

Pytorch [7] および `transformers`[8] を用いて実装した．Few-Shot 設定において，Prompting では各クラス 1 件ずつ，Fine-Tuning では各クラス 5 件ずつランダムにサンプリングして用いた.

疑似データ生成のパラメータ. 生成パラメータは，*max_tokens* を 500，*top-p* を 1.0，*temperature* を 1.2，*frequency penalty* を 0.02 として，一度に 5 件のデータを生成した．また，疑似データ生成数 M は各クラス 25,000 件行った．ただし，実数値を推定する JSTS は {0.,1.,2.,3.,4.,5.} の 6 個をクラスとして疑似データ生成を行った.

下流タスクの学習パラメータ. 実験は NVIDIA TITAN RTX GPU (24GB) 上で行った．Fine-Tuning 時の日本語 BERT モデルとして `cl-tohoku/bert-base-japanese-v3` を用いた．下流タスクにおける学習パラメータを表 3 に示す．ただし，各値は [11] を参考にした．また，各タスクにおける最大入力トークン長を表 4 に示す.

表 3 学習パラメータ．opt. は最適化手法，*bs* はバッチサイズ，*lr* は学習率，*ep* は学習エポック数，*warmup* は warmup ratio，*smooth* は label smooth の温度を指す.

opt.	*bs*	*lr*	β_1	β_2	*ep*	*warmup*	*smooth*
AdamW	32	5e-05	0.9	0.999	4	0.1	0.1

表 4 各タスクにおける最大入力トークン長.

MARC-ja	JNLI	JSTS	JCoLA	News	COVID-19
512	512	512	128	512	384

A.4 Prompt

各タスクに用いた Prompt は https://github.com/retrieva/JapaGen を参照.

A.5 Self-BLEU

§4.2 の多様性分析で用いた Self-BLEU の算出方法について説明する．まず，データセット内の 1 つのテキストについて，それ以外のテキストをリファレンスとした BLEU スコア [14] を算出する．そして，データセット内の全テキストに対して BLEU スコアを算出し，その平均がデータセットの Self-BLEU である．Self-BLEU は表層的な一致度合いを計測する指標であり，低いほど多様性が高いことを表す.

A.6 ラベル正確性に関する分析

生成した疑似データにおけるラベルの正確性を分析する．正確性の測定は，[2] に基づいて実施し，Gold を学習データ，Gold，JapaGen および KADG のデータを評価データとした場合の性能を正確性とする．各タスクにおけるラベル正確性の結果を表 5 に示す．表 2 と合わせて，KADG は JapaGen と同程度かそれ以上のラベル正確性を維持したまま，疑似データの多様性を向上させることができる.

表 5 各種データの正確性．JSTS を除き，値が高い方が正確である．JSTS のみ，教師ラベルとの平均二乗誤差を記載.

Method	**Label Correctness** (%)					
	MAR.	**JSTS**	**JNLI**	**JCoLA**	**News**	**COV.**
Gold	99.06	0.1365	98.01	96.28	98.89	90.87
JapaGen	99.97	1.5402	35.11	66.34	49.84	60.80
w/ KADG	99.96	1.5402	39.37	-	-	-

5) https://www.db.info.gifu-u.ac.jp/covid-19-twitter-dataset/
6) http://www.rondhuit.com/download.html#ldcc
7) https://pytorch.org/
8) https://github.com/huggingface/transformers