# 自己修正に基づく固有表現抽出モデルの指示学習

高橋拓誠　谷口友紀　大熊智子
旭化成株式会社
{takahashi.tkr, taniguchi.tcr, okuma.td}@om.asahi-kasei.co.jp

## 概要

固有表現抽出における過抽出やラベルの認識誤りは，典型的な性能低下の要因であり，大規模言語モデルにおいても頻出する課題である．本稿では，固有表現抽出を学習する過程で，抽出エラーを自己修正 (Self-Correction) するための指示学習を提案する．本研究における自己修正の学習は，固有表現抽出の教師データのみ参照し，抽出エラーの検証や修正に関する追加アノテーションを必要としない．化学分野の特許を対象とした固有表現抽出において，自己修正に基づく指示学習を導入することで，固有表現の抽出エラーを大幅に抑制できることを示した．

## 1　はじめに

固有表現抽出 (Named Entity Recognition; NER) は，文章中から人名や組織名などの固有表現の抽出および分類を行う自然言語処理の基礎的な下流タスクである．従来は抽出モデル [1] による分類問題として扱うことが多かったが，近年の大規模言語モデル (Large Language Model; LLM) の発展に伴い，いくつかの研究 [2, 3, 4] では生成モデルによるアプローチが抽出モデルより高性能であると報告されている．

しかしながら，LLM によるアプローチも万能ではなく，少量データ [5] や特定のデータセット [2, 4] においては抽出モデルの方が高性能であり，依然として抽出エラーが発生する．NER における抽出エラーを抑制するため，VerifiNER[6] では Self-Consistency[7] に基づく事後修正を可能とした．しかしながら，大量のエンティティが登録された知識ベースの利用を前提としており，こうした大規模知識ベースを常に用意することは現実的ではない．

一方，LLM の高度な推論能力を活かした自己修正 (Self-Correction; SC) に基づくアプローチが近年注目を集めている [8, 9, 10]．LLM が自律的に修正することから，必ずしも外部知識の類を必要としない利点はあるものの，その効果は算術推論 [11]，常識推論 [12]，コード生成 [13] のようなタスクに集中しており，情報抽出で効果を示した例はない．

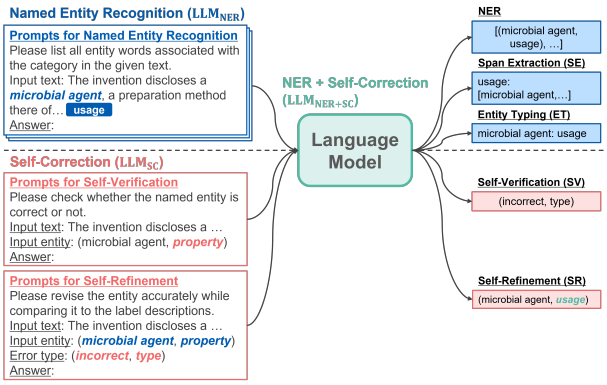


**図 1**　固有表現抽出と自己修正の指示学習 (LLM$_{NER+SC}$)．

本研究では，LLM が自らの誤りを検証 (Self-Verification; SV) し，検証結果に基づき自らの誤りを修正することで回答を洗練 (Self-Refinement; SR) する方法を提案する．具体的には，NER を指示学習する過程で SC を同時指示学習することで，自らの誤りを検証・修正するための知識を導入する (図 1)．本研究における SC の学習データは，NER の教師データと学習済みの NER モデルのみ参照し自動構築するため，誤りパターンの生成や抽出エラーの検証・修正に関する**人手の追加アノテーションを必要としない**．本研究の貢献は以下の 3 点である．

- NER における抽出エラーを定義し，修正可能な抽出エラーを事前分析により明らかにする．
- NER と SC を同時指示学習することで，NER の性能が向上することを示す．
- 推論時に SC を実行することで，固有表現の抽出エラーを大幅に抑制できることを示す．

## 2　予備分析

**NER における抽出エラーの定義:** NER の抽出エラーは，抽出すべきではない固有表現を抽出した場合 (False Positive; FP) と抽出すべき固有表現を抽出しなかった場合 (False Negative; FN) に分類できる [6]．

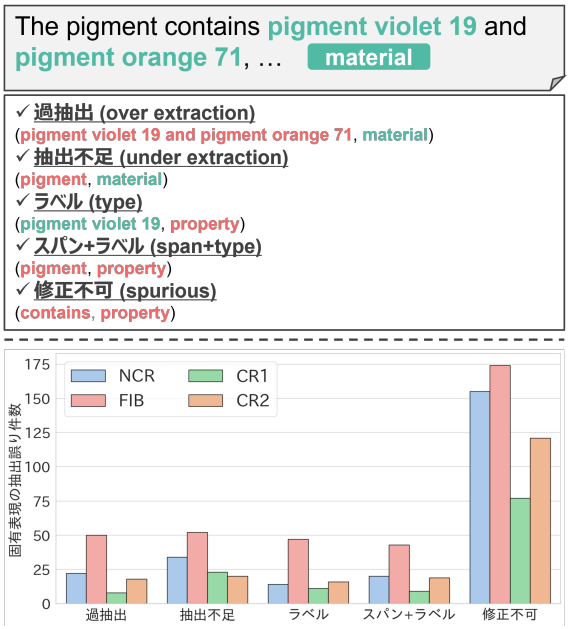


**図 2** NER における抽出エラーの定義 (上図) および予備実験における固有表現の抽出エラーの統計量 (下図).

特に，本稿では FP の抑制に焦点を当てて 5 種類の抽出エラーのパターンを定義した (図 2)．具体的には，**過抽出**および**抽出不足**はエンティティのスパンの部分不一致に起因するエラー，**ラベル**はエンティティのラベル不一致に起因するエラーとして定義される．さらに，エンティティおよびラベルの不一致に起因するエラーとして**スパン+ラベル**を定義した．一方で，抽出したエンティティが正解のエンティティと部分一致しないものを**修正不可**として定義した．このエラーに該当する場合，スパンやラベルの一部を修正しても正解の固有表現とは一致しない．

**予備実験:** 化学分野の特許をもとに構築した NER のデータセット[1]を用いて，公開 LLM[2]の指示学習を行い評価した．図 2 に示す予備実験の結果から，指示学習した LLM においても FP による抽出エラーは依然として頻発することがわかった．観測された抽出エラーのうち，修正不可を除く 4 種類のエラーは全体の 4 割程度を占めており，これらはエンティティまたはラベルの一部を修正することで正解の固有表現と一致する．したがって，**自己検証 (SV) の結果に基づき自己洗練 (SR) することで抽出エラーを抑制する**ことが期待される．一方で，修正不可のエラーは SR による修正はできないものの，**SV で検出して最終的な抽出結果から除外することで抽出エラーを抑制する**ことが可能である．

1) データセットの詳細は付録 A を参照されたい．
2) https://huggingface.co/google/gemma-2-2b-it

# 3 自己修正可能な固有表現抽出

本研究では，固有表現抽出 (NER) と自己修正 (SC) を同時指示学習するモデルを提案する．同時指示学習モデルのアーキテクチャとして，InstructUIE[2] を採用する．図 1 に提案モデルの概要を示す．以降では，固有表現抽出モデル $\mathrm{LLM}_{\mathrm{NER}}$ の指示学習 (3.1 節) と自己修正モデル $\mathrm{LLM}_{\mathrm{SC}}$ の指示学習 (3.2 節) を個別に説明するが，これらを同時指示学習することで提案モデル $\mathrm{LLM}_{\mathrm{NER+SC}}$ を構築する点に注意されたい．

## 3.1 固有表現抽出の指示学習

本節では，入力文 $T$ に含まれるラベル $l_i$ のエンティティ $e_i$ からなる固有表現 $s_i = (e_i, l_i)$ のリスト $S = \{s_1, ..., s_N\}$ を生成するため，固有表現抽出 (NER)，スパン抽出 (Span Extraction; SE)，エンティティ型推定 (Entity Typing; ET) を同時指示学習する．

**NER:** 入力 $X_{\mathrm{NER}} = [I_{\mathrm{NER}}; T]$ から固有表現の系列 $S = \{s_1, ..., s_N\}$ を得るための指示学習を行う．なお，$I_{\mathrm{NER}} = [P_{\mathrm{NER}}; L; D]$ であり，NER タスクの命令文 $P_{\mathrm{NER}}$，ラベル候補 $L = \{l_1, ..., l_M\}$，各ラベルの定義文の集合 $D = \{d_1, ..., d_{|L|}\}$ から構成される．

**SE:** 指定したラベル $\bar{l}_j \in L$ のエンティティの系列 $E = \{e_1, ..., e_{N'}\}$ を生成する指示学習により，入力 $X_{\mathrm{SE}} = [I_{\mathrm{SE}}; T]$ から出力系列 $Y_{\mathrm{SE}} = \{\tilde{e}_1, ..., \tilde{e}_n\}$ を得ることができる．ここで，$I_{\mathrm{SE}} = [P_{\mathrm{SE}}; L; D; \bar{l}_i]$ である．

**ET:** エンティティ $\bar{e}_i$ のラベル $l_i$ を生成する指示学習により，入力 $X_{\mathrm{ET}} = [I_{\mathrm{ET}}; T]$ から出力 $\tilde{l}_i$ を得ることができる．ここで，$I_{\mathrm{ET}} = [P_{\mathrm{ET}}; L; D; \bar{e}_i]$ である．

## 3.2 自己修正の指示学習

本節では，入力文 $T$ から抽出した固有表現 $\bar{s}_i$ が正しいか自ら検証し，検証結果に基づき自ら修正するために必要な知識を自己検証 (SV) と自己洗練 (SR) を同時指示学習することで獲得する．

**SV:** 固有表現 $\bar{s}_i = (\bar{e}_i, \bar{l}_i)$ の正誤判定 $v_i$[3]およびエラー分類 $c_i$[4]を生成する．すなわち，入力 $X_{\mathrm{SV}} = [I_{\mathrm{SV}}; T]$ から出力 $Y_{\mathrm{SV}} = (\bar{v}_i, \bar{c}_i)$ を得るための指示学習を行う．ここで，$I_{\mathrm{SV}} = [P_{\mathrm{SV}}; L; D; \bar{s}_i]$ である．

**SR:** 固有表現 $\bar{s}_i = (\bar{e}_i, \bar{l}_i)$ と自己検証 $\bar{Y}_{\mathrm{SV}} = (\bar{v}_i, \bar{c}_i)$ に基づき，修正後の固有表現 $\hat{s}_i$ を生成する．すなわち，入力 $X_{\mathrm{SR}} = [I_{\mathrm{SR}}; T]$ から出力 $\hat{s}_i$ を得るための指示学習を行う．$I_{\mathrm{SR}} = [P_{\mathrm{SR}}; L; D; \bar{s}_i; \bar{Y}_{\mathrm{SV}}]$ である．

3) 正解 (correct) または不正解 (incorrect) の二値分類とした．
4) 2 節の予備分析より，5 クラスのエラー分類を定義した．

### 3.3 固有表現抽出の自己修正

推論時は，(1) NER，(2) SV，(3) SR の順に処理することで，固有表現抽出の自己修正を実現する．

**NER の推論**では，テキスト $T$ を含む入力 $X_{\text{NER}}$ を学習済みモデル $\text{LLM}_{\text{NER+SC}}$ に与え，得られた結果を Parser により固有表現のリストに変換する．したがって，$\tilde{S} = \text{Parser}(\text{LLM}_{\text{NER+SC}}(Y_{\text{NER}}|X_{\text{NER}}))$ により，固有表現の候補リスト $\tilde{S} = \{\tilde{s}_1, ..., \tilde{s}_n\}$ を生成する．

**SV の推論**では，NER の推論で得られた固有表現 $\tilde{s}_i$ に対して，$\text{LLM}_{\text{NER+SC}}(Y_{SV}|X_{SV})$ により正誤判定 $\tilde{v}_i$ とエラー分類 $\tilde{c}_i$ を生成する．ここで，正誤判定 $\tilde{v}_i$ が正解 (correct) である固有表現 $\tilde{s}_i$ は，SR を実行せず最終回答 $A$ に追加する．また，正誤判定 $\tilde{v}_i$ が不正解 (incorrect) かつエラー分類が修正不可 (spurious) である場合は，SR を実行せず対象の固有表現 $\tilde{s}_i$ を棄却する．その他の場合，固有表現 $\tilde{s}_i$ に対して SR を実行することで，自らの誤りを修正する．

**SR の推論**では，固有表現 $\tilde{s}_i$ および自己検証 $Y_{SV}$ に基づき，モデル $\text{LLM}_{\text{NER+SC}}$ により修正する．具体的には，$\text{LLM}_{\text{NER+SC}}(\hat{s}_i|X_{SR})$ により，修正後の固有表現 $\hat{s}_i$ を生成し，回答候補 $\tilde{A}$ に追加する．なお，回答候補 $\tilde{A}$ に追加された固有表現 $\hat{s}_i$ を対象に SV と SR を再実行することで修正精度を高める．この処理は，$\hat{s}_i$ が (a) 最終回答 $A$ に追加される，(b) SV の結果に基づき棄却される，(c) あらかじめ指定した回数に到達する，まで繰り返し実行される．

### 3.4 自己修正データセットの自動構築

本研究では，NER の学習データ $\mathcal{D}_{\text{NER}}$ および固有表現抽出モデル $\text{LLM}_{\text{NER}}$ のみ参照し，SC の学習データ $\mathcal{D}_{\text{SC}}$ を自動構築する．

具体的には，$K$ 分割した NER の学習データ $\mathcal{D}_{\text{NER}} = \{d_1, ..., d_K\}$ より，$d_i$ を用いてモデル $\text{LLM}^i_{\text{NER}}$ を学習する．その後，$d_j (i \neq j)$ からサンプリングした任意のデータ $u$ に対して，$\tilde{S} = \text{Parser}(\text{LLM}^i_{\text{NER}}(Y_{\text{NER}}|X_{\text{NER}}))$ により，固有表現の候補リスト $\tilde{S} = \{\tilde{s}_1, ..., \tilde{s}_n\}$ を生成する．なお，多様な誤りパターンを生成するため，本過程は Top-K サンプリング [14] によるデコーディング[5]を行う．最終的に，ここで得られた固有表現 $\tilde{s}_k$ について $u$ の正解データ $S = \{s_1, ..., s_N\}$ を参照しながら，正誤判定 $v_k$，エラー分類 $c_k$，修正後の固有表現 $\hat{s}_k$ 自動付与し，自己修正の教師データ $z = (u, \tilde{s}_k, v_k, c_k, \hat{s}_k)$ を $\mathcal{D}_{\text{SC}}$ に追加する．

5) top-$k$ = 40 に設定し 16 個の系列を生成した．

## 4 評価実験

本稿では，“固有表現抽出における**抽出エラーを LLM が自ら検証し修正できるか？**” という観点で評価を実施する．先行研究 [15] に倣い，評価方法が公平 (最も良い初期応答に対する修正可能性の評価) かつ現実的 (正解データを参照しない自己修正の評価) であることに留意して評価した．

### 4.1 実験設定

**データセット:** 化学分野の特許を対象に人手でアノテーションした固有表現抽出 (NER) のデータセットを用いて評価する．本データセットは，4 つのドメイン (NCR，FIB，CR1，CR2) から構成されており，In-domain および Out-of-domain の設定で評価する．データセットの詳細は付録 A を参照されたい．

**ベースライン:** 抽出モデルのベースラインとして，BERT[1] と LUKE[16] に CRF 層 [17] を追加した $\text{BERT}_{\text{base}}$+CRF と $\text{LUKE}_{\text{large}}$+CRF を用意した．生成モデルのベースラインとして，OpenAI の GPT-4o[6]を採用した．なお，GPT-4o はゼロショット設定および文脈内学習 (In-Context Learning; ICL) を利用した設定の 2 種類を比較した．提案モデルの LLM は gemma-2[7]を採用し，QLoRA[18] による量子化を行い指示学習した．詳細は付録 B を参照されたい．

**評価指標:** 抽出した固有表現 $\tilde{s}_i = (\tilde{e}_i, \tilde{l}_i)$ が正解の固有表現リスト $S = \{s_1, ..., s_N\}$ に含まれる場合を正解とみなし，Micro-F1 で性能評価した．

### 4.2 評価結果

**In-domain の評価:** 表 1 より，自己修正 (SC) に基づく NER の同時指示学習を行った提案モデル $\text{LLM}_{\text{NER+SC}}$ がベースラインと比較して大きく性能向上していることがわかる．この結果から，**NER の学習過程で自らの誤りの検証や修正を学習する**ことは，NER の性能向上に大きく寄与するといえる．さらに，推論時に SC を実行したモデル $\text{LLM}_{\text{NER+SC}}$ + Self-Correction は，平均性能をさらに向上させることを示した．この結果から，**推論時に自己検証 (SV) し，検証結果に基づき自己洗練 (SR) することは，NER の抽出精度の向上に貢献する**といえる．一方，SV の性能が 100%であると仮定した場合の NER の性能 (Self-Correction w/ Oracle SV) は，

6) https://platform.openai.com/docs/models/gpt-4o
7) https://huggingface.co/google/gemma-2-2b-it

**表 1** In-domain および Out-of-domain における NER の評価結果.

| | In-domain | | | | | Out-of-domain | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | **NCR** | **FIB** | **CR1** | **CR2** | **mean** | **NCR** | **FIB** | **CR1** | **CR2** | **mean** |
| GPT-4o (zero-shot) | 0.423 | 0.381 | 0.323 | 0.463 | 0.397 | 0.423 | 0.381 | 0.323 | 0.463 | 0.397 |
| GPT-4o (ICL, 5-shot) | 0.700 | 0.540 | 0.534 | 0.704 | 0.619 | 0.622 | 0.486 | 0.476 | 0.686 | 0.567 |
| $\text{BERT}_{\text{base}}$+CRF | 0.692 | 0.435 | 0.399 | 0.621 | 0.537 | 0.400 | 0.341 | 0.413 | 0.477 | 0.408 |
| $\text{LUKE}_{\text{large}}$+CRF | 0.761 | 0.519 | 0.516 | 0.740 | 0.634 | 0.606 | 0.477 | 0.569 | 0.680 | 0.583 |
| $\text{LLM}_{\text{NER}}$ | 0.769 | 0.569 | 0.558 | 0.726 | 0.655 | 0.622 | **0.509** | 0.544 | 0.680 | 0.589 |
| $\text{LLM}_{\text{NER+SC}}$ | 0.772 | **0.576** | 0.625 | **0.747** | 0.680 | 0.642 | 0.507 | 0.551 | **0.705** | 0.601 |
| $\text{LLM}_{\text{NER+SC}}$ + Self-Correction | **0.787** | 0.552 | **0.660** | 0.739 | **0.684** | **0.661** | 0.486 | **0.604** | 0.697 | **0.612** |
| Self-Correction w/ Oracle SV | 0.888 | 0.838 | 0.799 | 0.916 | 0.860 | 0.814 | 0.801 | 0.752 | 0.909 | 0.819 |

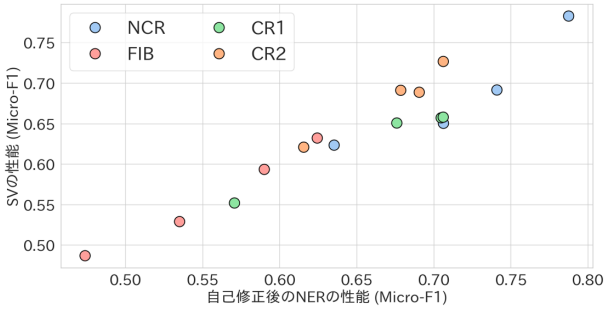


**図 3** SV の性能と最終的な NER の性能の相関 ($r = 0.951$). In-domain および Out-of-domain の設定で評価した.

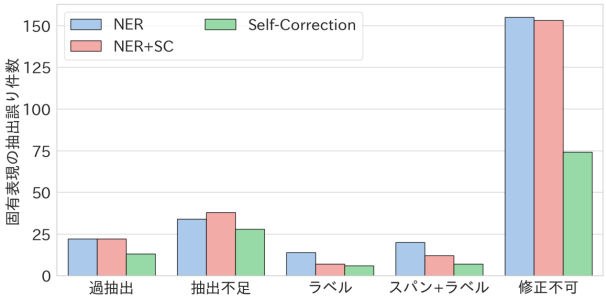


**図 4** ドメイン NCR における抽出エラーの発生数. NER は $\text{LLM}_{\text{NER}}$, NER+SC は $\text{LLM}_{\text{NER+SC}}$, Self-Correction は $\text{LLM}_{\text{NER+SC}}$ + Self-Correction に対応する.

さらに高い値を示した．すなわち，**SR により回答を修正する効果は大きく**，SC が性能向上に寄与するかどうかは，**SV に大きく依存する**と推察される．

**Out-of-domain の評価:** 表 1 より，自己修正に基づく指示学習の効果は，Out-of-domain においても健在であることがわかる．さらに，推論時に SC を適用すると平均性能が向上することから，Out-of-domain においても SC による修正は効果的であるといえる．

## 4.3 分析

**SV の性能と SC の性能の関係性:** 4.2 節より，SC の効果は SR よりむしろ SV に大きく依存する可能性が示された．図 3 に SV の性能と SC 適用後の NER 性能の関係性を示す．SV と最終的な NER の性能には，非常に強い正の相関 ($r = 0.951$) を確認できることから，**SC の効果は SV に大きく依存する**ことがわかった．SC の効果がフィードバック生成 (本研究における SV に対応する処理) の性能に大きく依存することは，先行研究 [19, 10, 20] でも同様に指摘されていることから明らかである．

**自己修正により抽出エラーを抑制できたか？** 図 4 に抽出エラーの発生頻度を示す．$\text{LLM}_{\text{NER+SC}}$ より，NER と SC を同時指示学習することでラベル誤りを伴う抽出エラーが抑制されていることがわかる．さらに，推論時に SC を適用することで，修正不可とした抽出エラーを大幅に抑制するとともに，スパン誤りやラベル誤りを伴う修正可能なエラーの発生頻度も減少することが示された．したがって，本稿で提案した NER と SC の同時指示学習および推論時に SC を実行することは，NER をより精緻化することに貢献する．また，実際のエラー修正の事例については付録 C を参照されたい．

# 5 おわりに

本研究では，固有表現抽出における抽出エラーを抑制するため，固有表現抽出と自己修正を同時指示学習するモデルを提案した．自己修正に基づく指示学習は，固有表現抽出の性能を向上させるとともに，推論時に自己修正を適用することでさらに抽出エラーを抑制可能であることを示した．

今後の課題として，4.3 節で示したように自己検証 (SV) の性能を改善することが挙げられる．また，関係抽出などの他の情報抽出タスクにおいても同様に効果が得られるか検証する予定である．

# 参考文献


[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pp. 4171–4186, 2019.

[2] Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. Instructuie: Multi-task instruction tuning for unified information extraction. **arXiv preprint arXiv:2304.08085**, 2023.

[3] Guochao Jiang, Ziqin Luo, Yuchen Shi, Dixuan Wang, Jiaqing Liang, and Deqing Yang. ToNER: Type-oriented named entity recognition with generative language model. In **Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)**, pp. 16251–16262, 2024.

[4] Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. UniversalNER: Targeted distillation from large language models for open named entity recognition. In **The Twelfth International Conference on Learning Representations**, 2024.

[5] Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 10572–10601, 2023.

[6] Seoyeon Kim, Kwangwook Seo, Hyungjoo Chae, Jinyoung Yeo, and Dongha Lee. VerifiNER: Verification-augmented NER via knowledge-grounded reasoning with large language models. In **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 2441–2461, 2024.

[7] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In **The Eleventh International Conference on Learning Representations**, 2023.

[8] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[9] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In **The Twelfth International Conference on Learning Representations**, 2024.

[10] 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. In **The Twelfth International Conference on Learning Representations**, 2024.

[11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. **arXiv preprint arXiv:2110.14168**, 2021.

[12] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pp. 4149–4158, 2019.

[13] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. **arXiv preprint arXiv:2108.07732**, 2021.

[14] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 889–898, 2018.

[15] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. **Transactions of the Association for Computational Linguistics**, Vol. 12, pp. 1417–1440, 2024.

[16] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep contextualized entity representations with entity-aware self-attention. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 6442–6454, 2020.

[17] John D. Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In **International Conference on Machine Learning**, 2001.

[18] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023.

[19] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In **The Twelfth International Conference on Learning Representations**, 2024.

[20] Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In **The Twelfth International Conference on Learning Representations**, 2024.

**表 2**　NCR および FIB の統計量．† はサンプルあたりの平均値を示す．

| | NCR | | | FIB | | |
|---|---|---|---|---|---|---|
| | train | dev. | test | train | dev. | test |
| サンプル数 | 831 | 193 | 194 | 676 | 161 | 126 |
| 単語数 † | 67.5 | 63.6 | 64.7 | 51.1 | 52.3 | 57.5 |
| 固有表現数 † | 5.0 | 4.7 | 5.0 | 4.5 | 5.0 | 5.0 |
| usage | 845 | 168 | 189 | 541 | 120 | 114 |
| material | 2,750 | 619 | 662 | 2,181 | 605 | 445 |
| property | 579 | 118 | 112 | 304 | 72 | 77 |

**表 3**　CR1 および CR2 の統計量．† はサンプルあたりの平均値を示す．

| | CR1 | | | CR2 | | |
|---|---|---|---|---|---|---|
| | train | dev. | test | train | dev. | test |
| サンプル数 | 191 | 64 | 77 | 628 | 116 | 137 |
| 単語数 † | 49.1 | 56.8 | 58.9 | 64.0 | 68.4 | 52.3 |
| 固有表現数 † | 3.0 | 3.7 | 3.1 | 4.8 | 5.6 | 4.3 |
| usage | 142 | 49 | 71 | 539 | 120 | 109 |
| material | 411 | 178 | 161 | 2,251 | 474 | 435 |
| property | 24 | 7 | 8 | 238 | 52 | 51 |

## A　データセットの詳細

本稿では，化学分野の特許を対象として人手によるラベル付けを行い構築した固有表現抽出 (NER) のデータセットを用いて検証した．検証に使用する特許は，非結晶性樹脂 (NCR)，繊維 (FIB)，結晶性樹脂 (CR1，CR2) のドメインを対象に収集した．表 2 と表 3 に検証に使用したデータセットの統計量を示す．なお，指示学習および文脈内学習 (ICL) の事例は train，モデルの分析 (2 節，4.3 節) は dev.，モデルの性能評価 (4.2 節) は test を用いて実験した．

本データセットでは，**用途 (usage)**，**材料 (material)**，**特性 (property)** の 3 種類の固有表現ラベルが用意されている．各ラベルの定義と事例を以下に示す．

**用途 (usage)**　材料の用途あるいは機能を示す名詞 (複合名詞を含む) に付与するラベルである (例: organic photocuring，medical tube，silicon rubber)

**材料 (material)**　用途を実現するための構成要素を表す固有表現であり，形容詞を含む名詞に付与する (例: separator，silicone oil，polyimide wet gel)

**特性 (property)**　材料が持つ物理的な特性または用途に関係する性質を表す (例: bending property，super-hydrophobicity，melt strength)

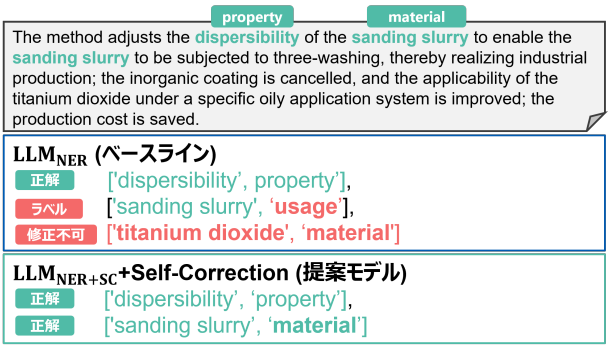


**図 5**　ベースラインによる抽出結果および提案モデルによる抽出結果の修正例．

## B　モデルの設定

LLM の指示学習では，QLoRA[18] による量子化を適用した．なお，$\alpha = 128$，$r = 64$ に設定した．また，バッチサイズは 1，gradient accumulation steps は 8，学習率は 2e-5 に設定し学習を進めた．推論時は，貪欲法によるデコーディングを実施した．なお，出力のサブワード数の上限は，固有表現抽出 (NER) は 128，自己検証 (SV) は 12，自己洗練 (SR) は 16 に設定した．推論時の自己修正 (SC) における SV および SR の繰り返し処理 (3.3 節) は 8 回を上限とした．

## C　固有表現抽出の修正例

図 5 にベースラインの固有表現抽出結果と提案手法における固有表現の修正例を示す．ベースライン LLM$_{NER}$ は，正解の固有表現 (“dispersibility”, “property”) を正しく抽出できているものの，“sanding slurry” のラベルを “usage” と誤認識するほか，正解の固有表現に部分一致しない無関係なエンティティ “titanium dioxide” を誤抽出している．一方，提案手法 LLM$_{NER+SC}$ + Self-Correction はベースラインが誤認識したラベルを修正することで (“sanding slurry”, “material”) を導出できた．さらに，自己検証 (SV) により不要な固有表現[8] を棄却することで，修正不可な予測結果 (“titanium dioxide”, “material”) の誤抽出を避けることができた．

8)　3.3 節より，SV の予測結果が $(\tilde{v}_i, \tilde{c}_i) = (\text{incorrect}, \text{spurious})$ となる固有表現 $\tilde{s}_i$ を棄却する．