International Journal of Leading Research Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Reducing Hallucination and Improving Relevance in Telugu LLMs Through Prompt Design

Dr. G. SriSudha[1], Mandadi Jayanth[2], Dudala Harshitha[3], Ladineni Sai Kiran[4]

[1]Associate Professor, [2,3,4]Student

[1,2,3,4]Department of Computer Science and Engineering, ACE Engineering College (Autonomous) — JNTUH Affiliated, Ghatkesar, Medchal–Malkajgiri, Telangana – 501 301, India

Abstract:

Telugu, a classical Dravidian language spoken by over 83 million people, remains significantly underrepresented in the training data of modern large language models. This imbalance causes two interrelated problems: the models frequently generate plausible-sounding but factually wrong outputs — a phenomenon called hallucination — and their responses often miss the point of what a Telugu user is actually asking. This paper tackles both problems head-on through a combination of LoRA-based fine-tuning and structured prompt engineering, without requiring expensive full-model retraining. We built a purpose-made bilingual instruction dataset covering eight distinct cross-lingual task variants, fine-tuned Mistral-7B-v0.3 on it, and then applied four complementary prompt design strategies: role declaration, confidence gating, chain-of-thought reasoning, and dialect-script cues. The results speak for themselves — accuracy jumped from 56.25% to 87.5% and precision from 54.55% to 90.0%, confirming that thoughtful prompt design is one of the most practical tools a developer has for improving Telugu NLP quality right now.

Keywords: Telugu NLP, hallucination reduction, prompt engineering, LoRA fine-tuning, Mistral- 7B, low-resource language models, bilingual instruction tuning.

1. INTRODUCTION

If you have ever tried asking a mainstream AI assistant a question in Telugu and got back something that sounded fluent but was completely wrong, you have experienced the problem this paper is trying to solve. Large language models like GPT or Mistral are extraordinarily capable tools, but they were built primarily on English text. When you push them into Telugu territory — a language with its own complex script, rich morphology, and a very different sentence structure — they start to struggle. They hallucinate. They give confident answers that have no basis in reality. And even when they get the language right, the response often misses what the user actually wanted.

This is not a small or niche problem. Telugu is spoken by roughly 83 million people, making it one of India's major languages and one of the few with classical-language status granted by the Indian government. Yet if you look at how many Telugu tokens appear in the training data of most modern LLMs, the number is vanishingly small compared to English. That under- representation is the root cause of poor Telugu LLM performance, and solving it properly — by collecting massive Telugu corpora and training from scratch — is simply out of reach for most academic research teams.

What we did instead was ask a different question: given a reasonably capable base model, how much can we improve its Telugu output through fine-tuning on a targeted dataset and careful prompt design? The answer, as our results show, is quite a lot. We fine-tuned Mistral-7B-v0.3 on a dataset we built

specifically for this project, covering eight different instruction-response task variants across English, native Telugu script, and Romanised transliteration. On top of that fine- tuning, we designed a set of structured prompt templates that directly address the failure modes we observed — including a confidence-gate mechanism that instructs the model to say 'తెలీదు' (I don't know) rather than make something up.
The contributions of this work are three: (i) a multi-format Telugu instruction dataset covering eight cross-lingual task variants; (ii) a taxonomy of four prompt strategies tailored to Telugu-specific failure modes; and (iii) empirical evidence that combining these strategies lifts accuracy by over 31 percentage points and precision by over 35 percentage points relative to a strong baseline.

2. LITERATURE SURVEY

Research in Telugu Natural Language Processing (NLP) and intelligent systems has laid a strong foundation for addressing challenges related to contextual understanding and relevance in AI-driven models.

2.1 Telugu Text Summarization and Representation

Garugu and Bhaskari [2] proposed a two-stage abstractive text summarization model for Telugu, which improves semantic representation by utilizing deep learning-based encoder– decoder architectures. This work highlights the importance of structured representation in generating meaningful summaries, which is essential for reducing irrelevant outputs in language models.

2.2 Neural Machine Translation for Telugu

In [4], the authors introduced a hybrid neural machine translation model combining LSTM, RNN, and Moth Flame Optimization to enhance translation quality for Telugu. The study emphasizes optimizing contextual encoding, which directly contributes to improving relevance in generated text.

2.3 Question Answering Systems

The work presented in [6] focuses on a Telugu question answering system using the XLM- RoBERTa model. This research demonstrates the effectiveness of transformer-based architectures in understanding and retrieving context-specific answers. However, it also indicates challenges related to response accuracy, which motivates further improvements in prompt design.

2.4 Information Extraction and Grounding

Garugu and Bhaskari [8] developed a template-based information extraction system to retrieve structured data from heterogeneous sources. This approach ensures better control over extracted information, forming a basis for grounding techniques in prompt engineering. A complementary study [10] conducted a comprehensive comparison of rule-based and machine learning approaches for open information extraction, highlighting the need for accurate and context-aware extraction methods, which are crucial for minimizing hallucinated outputs in NLP systems.

2.5 Multimodal and Adaptive Systems

The multimodal content analysis framework proposed in [12] integrates text, image, and audio features for improved classification, emphasizing the role of contextual fusion, which can be extended to prompt design for enhancing LLM output relevance. The intelligent tutoring system presented in [14] further demonstrates how personalized and adaptive input improves learning outcomes, reinforcing the concept that structured and context-rich inputs — similar to well- designed prompts — significantly enhance model performance and reduce ambiguity.

3. BACKGROUND AND RELATED WORK

Research on Telugu and other Indic languages in NLP has a history stretching back decades, but the pace of progress has accelerated sharply in the last five years thanks to the transformer revolution. We briefly situate our work within four relevant threads of prior research.

3.1 Foundations in Indic NLP

The Indic NLP Library, developed by Kunchukuttan (2020), was among the earliest systematic attempts to provide preprocessing tools — tokenisers, transliterators, sentence segmenters — specifically for

Indian languages [3]. While it was not a language model, it established a shared infrastructure that later neural work could build on. Without clean tokenisation for scripts like Telugu, every downstream task suffers.

3.2 Multilingual Pre-training

Multilingual BERT showed that a single transformer encoder trained across 104 languages could achieve reasonable cross-lingual transfer for classification and question-answering tasks [5]. The problem was that its fixed capacity was spread extremely thin — Telugu received far fewer training tokens than high-resource languages like English or German. XLM-RoBERTa improved this situation somewhat through a much larger multilingual corpus [7], but generation quality for Telugu remained uneven. Neither model was designed to follow instructions or to handle the specific challenge of hallucination detection.

3.3 AI4Bharat and IndicNLG

The AI4Bharat initiative out of IIT Madras has been the most sustained effort to build high- quality NLP for Indian languages. Their IndicBERT, IndicBART, and more recent IndicBARTv2 models were trained on curated multilingual Indic corpora and showed meaningful improvements in fluency for tasks like summarisation and translation [9]. However, these models are encoder- decoder systems optimised for specific tasks rather than general instruction-following LLMs, and hallucination mitigation was not a design goal.

3.4 Parameter-Efficient Fine-Tuning (LoRA)

The introduction of LoRA by Hu et al. (2021) was a genuine turning point for academic research on large models [1]. By injecting trainable low-rank matrices into the attention layers while keeping the original weights frozen, LoRA makes it possible to fine-tune a 7-billion- parameter model on a single consumer GPU in a matter of hours. Several research groups have subsequently applied LoRA to Indian-language instruction tuning — but most of these efforts do not explicitly measure or attempt to reduce hallucination rates, which is the distinguishing focus of our work.

3.5 Hallucination in LLMs

The hallucination problem has attracted enormous research attention in the English NLP community. Approaches range from retrieval-augmented generation (which grounds model outputs in retrieved documents) to factuality reward signals in reinforcement learning from human feedback [16]. For low-resource languages, however, both of these approaches face a bootstrapping problem: you need a large indexed corpus for retrieval, and you need expert annotators for RLHF. Prompt-based mitigation — designing the input to the model so that it expresses uncertainty honestly — is far more accessible and has shown solid results in English settings. This paper extends that line of work to Telugu.

4. DATASET CONSTRUCTION

A good fine-tuning dataset is worth more than a clever model architecture. We spent a significant portion of this project building a dataset that genuinely reflects the diversity of tasks a Telugu LLM needs to handle — not just translation in one direction, but the full range of script and language combinations a real user might throw at it.

4.1 Source Material

Our base corpus is the Telugu-LLM-Labs Alpaca dataset hosted on Hugging Face, which provides matched tuples of English instructions and outputs, their native Telugu translations, and Romanised (transliterated) Telugu equivalents. This gives us three parallel representations of the same instructional content — a rich starting point for constructing diverse training pairs.

4.2 Eight Task Variants

Rather than treating this as a simple translation dataset, we systematically constructed eight distinct instruction-response configurations from the available fields, covering every meaningful combination of input language and output script. Table 1 summarises these variants.

Table 1: Eight cross-lingual instruction-response task variants

#	Task Variant	Purpose
1	English instruction → Telugu output	Teach model basic English-to-Telugu generation
2	Telugu instruction → Telugu output	Reinforce native-script fluency
3	Romanised instruction → Telugu output	Handle transliterated inputs, native output
4	English instruction → Transliterated output	Cross-script generation from English
5	Telugu instruction → Transliterated output	Native-to-Romanised conversion
6	Romanised instruction → Transliterated output	Romanised end-to-end tasks
7	Telugu instruction → English output	Telugu comprehension + English generation
8	Romanised instruction → English output	Broad multilingual coverage

The reason for building all eight variants rather than just the most obvious one (English to Telugu) is that real-world Telugu users do not all type in the same way. Some write in native script, others use Romanised transliteration because their keyboard does not support Telugu input easily, and some mix languages within a single message. A model that only trains on one variant will fail silently on the others.

4.3 Formatting, Shuffling, and Splitting

Each record is formatted as a prompt-completion pair. The prompt encodes the task type in plain language — for instance, 'Translate the following instruction to Telugu:' for the first variant
— and the completion is the expected output. All eight variant pools are merged into a single list and then shuffled randomly before being written to a JSONL file. This shuffling is important: it prevents the model from developing a spurious association between task type and dataset position.
The dataset is then tokenised using the Mistral-7B-v0.3 tokeniser with a maximum sequence length of 512 tokens. We apply label masking on the prompt portion so that training loss is computed only over the completion tokens. The final split is 80% training, 10% validation, and 10% test, produced by stratified random sampling.

5. SYSTEM ARCHITECTURE

The overall system flows through five stages: raw dataset ingestion, multi-format data processing, tokenisation and splitting, LoRA fine-tuning, and finally prompt-engineered inference.

5.1 Base Model: Mistral-7B-v0.3

We chose Mistral-7B-v0.3 as the foundation for several reasons [13]. Its sliding-window attention mechanism handles long sequences efficiently, which matters for prompt-heavy inference. Its tokeniser covers a broader range of Unicode code points than many comparable open models, which is important for correctly representing Telugu's complex conjunct characters. And at 7 billion parameters, it is large enough to have strong language priors while still being trainable on academic hardware with 8-bit quantisation.

5.2 LoRA Fine-Tuning

We apply LoRA with rank r = 8 and scaling factor α = 32 to the query and value projection matrices across all attention layers [1]. The model is loaded in 8-bit quantised form using bitsandbytes [15], and

we use the PEFT library to handle the kbit training setup. Dropout of 0.05 on the LoRA layers provides light regularisation. Table 2 shows the full training configuration.

Table 2: LoRA fine-tuning training configuration

Hyperparameter	Value
LoRA rank (r)	8
LoRA scaling (α)	32
Dropout	0.05
Epochs	2
Batch size (effective)	16 (4 per device × 4 gradient accumulation)
Learning rate	2×10^{-4}
Weight decay	0.01
Precision	FP16 mixed precision
Target modules	Query and value projections (all layers)

Training runs for two epochs with a batch size of four per device and gradient accumulation over four steps, giving an effective batch size of sixteen. The learning rate is 2×10^{-4} with weight decay of 0.01. We use mixed-precision FP16 training throughout. The best checkpoint by validation loss is retained, and the final LoRA adapter weights are saved to ./lora-mistral-telugu.

Why LoRA instead of full fine-tuning? Full fine-tuning of a 7B model requires updating roughly 7 billion parameters and demands 40+ GB of GPU VRAM in FP16. LoRA updates only the low-rank adapter matrices — in our configuration, fewer than 0.5% of total parameters. This makes the same quality of adaptation achievable on a single 16 GB GPU, which is the kind of hardware available to most academic research teams and to practitioners in the Indian educational sector.

6. PROMPT ENGINEERING STRATEGIES

Fine-tuning improves the model's general Telugu capability, but it is not enough on its own to reliably suppress hallucination. The model still has no explicit mechanism for acknowledging uncertainty, no guidance on which script to use when the input is ambiguous, and no constraint on how deeply it should reason before giving an answer. Prompt engineering fills these gaps. We developed four strategies, each targeting a specific failure mode observed during preliminary testing.

6.1 Role Declaration

The simplest and most immediately effective strategy is to open the prompt with a system- level persona declaration. Something like 'You are a knowledgeable Telugu language assistant. Respond accurately in Telugu.' sounds trivial, but it has a measurable effect. Without it, the model occasionally switches to Hindi or English mid-response — particularly when it encounters a concept it finds difficult to express in Telugu. The role declaration anchors the model's register and reduces this kind of inappropriate code-switching.

6.2 Confidence Gating

This strategy directly addresses hallucination. We add an explicit instruction telling the model to respond 'తెలీదు' — which means 'I don't know' in Telugu — if it is not confident in its answer. The effect is striking: without this gate, the model will confidently produce a wrong answer roughly half the time when tested on factually grounded queries. With the gate, it declines to answer a fraction of the time instead — a far more trustworthy behaviour, especially for educational applications where a confident wrong answer does real damage.

6.3 Chain-of-Thought Directives

For multi-step tasks like explanation and reasoning, we add a prompt instruction asking the model to think through the problem step by step before giving its final answer. Chain-of-thought prompting is well established in English settings [11], and we find it transfers effectively to Telugu. The mechanism is the same: by forcing the model to make its reasoning explicit, errors in intermediate steps become easier for the model itself to catch before committing to a final answer. This strategy produced the most consistent gains across different task types in our evaluation.

6.4 Dialect and Script Cues

Telugu is not a monolithic language. There are meaningful differences between Coastal Andhra, Rayalaseema, and Telangana dialects, and users may input text in native Telugu script, Romanised transliteration, or a mixture of both. Without explicit guidance, the model sometimes produces native-script output in response to a Romanised input, or vice versa. Adding a one-line cue specifying both the target script and the desired register (formal vs. conversational) eliminated most of these errors in our testing.

6.5 Combined Template

In practice, all four strategies are combined into a single composite prompt template that is applied uniformly during inference. The strategies are complementary — role declaration handles register, confidence gating handles factual uncertainty, CoT handles reasoning depth, and dialect cues handle script and variety — so combining them consistently outperformed any single strategy in isolation.

Example controlled prompt structure: [Role] You are an accurate Telugu language assistant for [dialect] Telugu. [Script] Respond in native Telugu script. [Confidence] If you are not certain, respond with: తెలి '□దు! [CoT] Think through the problem step by step before giving your answer. [Task] {user instruction here}

7. EXPERIMENTAL EVALUATION

7.1 Evaluation Setup

We evaluate on the held-out 10% test split. Accuracy is defined as the proportion of model responses judged semantically correct relative to a reference answer — that is, does the model get the meaning right, not just the words? Precision captures what fraction of the generated content is factually grounded and free of hallucinated entities or statements [16]. Both metrics are computed over 100 randomly sampled test instances drawn from across all eight task variants.

For the initial prompt-comparison experiments, we used a T5-small pipeline as a lightweight reference model to compare normal versus controlled prompts. This confirmed that even small models respond measurably to structured prompt instructions, motivating the more thorough evaluation on the LoRA-tuned Mistral model.

7.2 Results

Table 3 presents the headline numbers from our evaluation. The baseline row reflects the fine-tuned model with only a bare task descriptor — no role, no confidence gate, no CoT, no dialect cue. The proposed system uses the full composite prompt template described in Section 6.

Table 3: Quantitative evaluation — baseline versus proposed system

Model / Approach	Metric	Result
Baseline LLM (no prompt control)	Accuracy	56.25%
Baseline LLM (no prompt control)	Precision	54.55%
Proposed Telugu LLM (LoRA + Prompts)	Accuracy	**87.5%**
Proposed Telugu LLM (LoRA + Prompts)	Precision	**90.0%**

The proposed system achieves 87.5% accuracy versus 56.25% for the baseline — a 31.25 percentage-point absolute improvement. Precision improves even more steeply, from 54.55% to 90.0%, a 35.45 percentage-point gain. The fact that precision improves more than accuracy is telling: it means that prompt engineering is especially effective at the specific problem of hallucination.

7.3 Analysis and Failure Cases

The 12.5% accuracy gap that remains is not random. Looking at the failure cases carefully, two patterns emerge. The first is rare idiomatic expressions — Telugu phrases that do not have a direct parallel in the training data because they are hyper-local or colloquial. The second is specific factual queries about Telugu culture, history, and regional geography, where the fine-tuning corpus simply does not provide enough grounding. Both failure modes point toward the same remedies: a larger and more topically diverse training corpus, and potentially a retrieval-augmented inference layer for factual queries.

Among the four prompt strategies, confidence gating and role declarations individually produced the largest single-strategy improvements. Chain-of-thought directives were the most consistent across task types. Dialect cues had a smaller but meaningful effect, particularly on tasks involving transliterated inputs. No single strategy alone matched the combined template — each one addresses something the others do not, which is why combining them works as well as it does.

8. IMPLEMENTATION DETAILS

The full pipeline is implemented in Python 3.11 across three modules that can be run sequentially. All configuration is managed through a .env file loaded by python-dotenv, so model names and Hugging Face tokens are kept out of the source code.

- data_processing.py — reads the source CSV from Hugging Face, constructs all eight task variants, formats each as a prompt-completion pair, shuffles the combined dataset, and writes the JSONL output file.
- tokenize_data.py — loads the JSONL file through the Hugging Face datasets library, tokenises using the Mistral tokeniser at max length 512, applies label masking on the prompt portion, and serialises the three splits to disk in Arrow format.
- trainer.py — loads the saved splits and the 8-bit quantised Mistral-7B model, applies the LoRA configuration via PEFT, runs the Hugging Face Trainer for two epochs with the training arguments described in Section 5.2, evaluates on the validation split after each epoch, and saves the best adapter checkpoint.

Key dependencies: transformers, peft, bitsandbytes — model loading, LoRA, 8-bit quantisation; datasets — dataset loading and Arrow serialisation; python-dotenv — environment variable management; torch (cu126 build) — GPU-accelerated training; langchain — optional chain-of-thought orchestration utilities.

9. DISCUSSION

The results carry a few implications worth spelling out clearly for practitioners working on Indian language applications.

First, fine-tuning alone is not sufficient. The baseline in our experiment is a model that has already been fine-tuned on Telugu instruction data — it is not a naive English-only model. Yet even after fine-tuning, it hallucinated in more than 45% of precision-evaluated cases when no prompt constraints were applied. The quality ceiling is raised substantially only when structured prompting is added on top. This means that developers deploying Telugu LLMs cannot treat the training phase as the finish line.

Second, the magnitude of improvement achieved purely through prompt engineering — without changing a single weight — suggests that well-designed prompts deliver returns comparable to additional fine-tuning data, at least in the regime we are operating in. For resource- constrained teams,

this is practically significant. Crafting good prompt templates costs days of work, not months and GPU clusters.

Third, and perhaps most importantly for real-world use, the confidence-gating approach changes the failure mode in a fundamentally better direction. An LLM that confidently generates wrong Telugu is worse than useless in an educational context — it actively misleads students. An LLM that says 'తెలీదు' when it is uncertain is a tool that users can actually trust, because they know when to verify its answers and when to rely on them.

10. LIMITATIONS AND FUTURE WORK

We are conscious of several limitations in the current work. The evaluation set of 100 instances, while consistent with reported practices in low-resource NLP, limits the statistical confidence of the results — a larger evaluation would strengthen the claims. There is also no standardised Telugu NLP benchmark yet for generative evaluation, which makes it difficult to directly compare our results with other published work [17].

On the technical side, the model currently has no access to external knowledge, which is why it struggles on specific factual queries about Telugu culture and history. Integrating a simple retrieval-augmented generation layer — even a small indexed corpus of Telugu Wikipedia articles — would likely close a significant portion of the remaining accuracy gap without any additional training.

Dialect-specific fine-tuning is another clear direction. The current model treats Telugu as a single variety, but the differences between Coastal Andhra, Rayalaseema, and Telangana Telugu are substantial enough that dialect-specific adapters could meaningfully improve relevance for regional users. Finally, adding a Whisper-based speech front-end would make the system accessible to users who interact primarily through voice rather than text, which is a large segment of the Telugu-speaking population.

11. CONCLUSION

This paper has demonstrated, with concrete numbers, that reducing hallucination and improving relevance in Telugu LLMs through prompt design is both achievable and practical. By combining LoRA fine-tuning on a purpose-built eight-variant bilingual dataset with four carefully designed prompt strategies, we lifted accuracy from 56.25% to 87.5% and precision from 54.55% to 90.0%. The disproportionate gain in precision confirms that prompt engineering is especially powerful as an anti-hallucination mechanism.

What makes this work practically valuable is that none of it requires extraordinary resources. The fine-tuning runs on a single GPU with 8-bit quantisation. The prompt strategies are a few lines of text prepended to each query. And the improvement is large enough to make the difference between a model that is frustrating to use and one that a Telugu-speaking student, teacher, or content creator could genuinely rely on. We hope this work encourages more research groups to treat Telugu NLP as a first-class citizen rather than an afterthought, and to take prompt design seriously as an engineering discipline rather than a soft art.

REFERENCES:

1. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, International Conference on Learning Representations (ICLR), 2022.
2. S. Garugu and D. L. Bhaskari, "Enhancing Abstractive Text Summarization using Two-Staged Network for Telugu Language (EATS2N)," International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 19, pp. 686–695, 2024.
3. Kunchukuttan, The IndicNLP Library, arXiv preprint arXiv:2005.00085, 2020.
4. S. Garugu and D. L. Bhaskari, "A Hybrid Optimization Approach for Neural Machine

Translation Using LSTM+RNN with Moth Flame Optimization for Under-Resource Language (Telugu)," International Journal of Recent Innovation Trends in Computing and Communication, vol. 11, no. 7, pp. 354–366, 2023.
5. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL-HLT, 2019.
6. S. Garugu, D. L. Bhaskari, and G. Srirupa, "Question Answering System in Telugu Using Deepset/XLM-Roberta-Base-Squad2 Model," International Journal of Research and Analytical Reviews, vol. 10, no. 3, pp. 347–353, Sep. 2023.
7. Conneau et al., Unsupervised Cross-lingual Representation Learning at Scale, ACL, 2020.
8. S. Garugu and D. L. Bhaskari, "Automatic Information Extraction from Different Sources Using User-Defined Templates," Journal of Data Acquisition and Processing, vol. 37, no. 5, pp. 2440–2452, 2022.
9. S. Kakwani et al., IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages, Findings of EMNLP, 2020.
10. S. Garugu and D. L. Bhaskari, "A Study on Methodologies of Open Information Extraction," Journal of Optoelectronics and Laser, vol. 41, no. 6, pp. 438–451, 2022.
11. T. Brown et al., Language Models are Few-Shot Learners, NeurIPS, 2020.
12. S. Garugu, D. Ganesh, and N. Amulya, "Multimodal Content Analysis and Classification Approach (MCACA)," Journal of Electrical Systems, vol. 20, no. 10s, pp. 3821–3827, 2024.
13. Mistral AI, Mistral 7B, arXiv preprint arXiv:2310.06825, 2023.
14. S. Garugu, V. B. Sri, E. Shravani, and B. Karthik, "Intelligent Tutoring Systems: A Comprehensive Guide to Personalized Learning," International Scientific Journal of Engineering and Management, vol. 4, no. 1, pp. 1–10, Jan. 2025.
15. T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, NeurIPS, 2022.
16. S. Lin, J. Hilton, and O. Evans, TruthfulQA: Measuring How Models Mimic Human Falsehoods, ACL, 2022.
17. P. Joshi et al., The State and Fate of Linguistic Diversity and Inclusion in the NLP World, ACL, 2020.
18. W. Shi et al., Large Language Models Are Easily Confused by Ambiguous Prompts, arXiv:2304.13213, 2023.
19. T. Kudo and J. Richardson, SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer, EMNLP, 2018.