Last Report Summary

In our initial analysis, we examined three Hindi language models: Krutrim AI, Airavata, and OpenHathi. We classified more than 120 professions according to gender association and used them to assess gender bias in the models. Bias metrics were calculated using model responses, alongside Relative Norm Difference (RND) and Word Embedding Association Test (WEAT) analyses for the evaluation of inherent bias. Key findings indicated that Krutrim exhibited strong masculine biases, while Airavata showed relatively better gender balance. OpenHathi, however, struggled with proper sentence generation, yielding inconclusive results.
Link to previous report: https://github.com/dileepp982/Gender_Bias/blob/main/progress_overview1.pdf

Key Discussion Points of last meeting

In last meeting, the discussion points were:

1. Check Bias score metric proposed for Hindi language for other Indic Language sharing same structure.
2. Bias score metric background research
3. Check how bias evaluation on LLM done for English
4. How model weights are stored for Airvat and OpenHathi
5. Can Indic LLM's run locally in Mobile. If so How?
6. What are benefits of LLM's deployed on mobile.

We'll clarify each point in this report.

1 Bias Score metric for Telugu Language

Our bias score metric uses gendered verbs for differentiating male verbs with female ones. India's linguistic diversity is one of its most striking features, where few languages use gendered grammar, meaning that nouns, pronouns, and verbs change form depending on the gender they signify. In figure1 we can observe diversity in grammatical genders usage across Indian states. We chose Telugu language **Telugu-Navarasa 2.0** for our gender bias analysis. Analysis framework for Marathi language is ready too but on trying multiple different models: MahaMarathi-7B-v24.01-Base and Shivneri-marathi-llm-7b-v0.1 the generated outputs weren't good enough to perform analysis.

Identifying Gender in Telugu Verbs

In Telugu, the verb's form changes based on the subject's gender. This phenomenon is known as **gender agreement**, where the verb conjugates to align with the subject (masculine, feminine, or neuter). The verb form changes based on the **gender of the subject** performing the action.

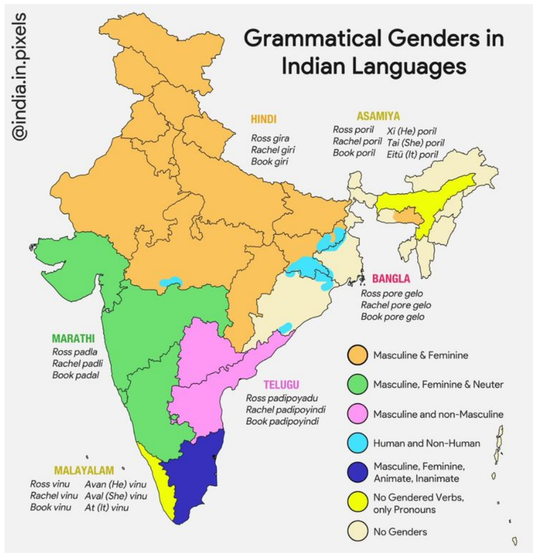


Figure 1: Grammatical Genders across Indian States

1. Masculine Subject (పుంసకారము) - ***Punsakāramu***

When the subject is masculine, the verb usually ends with the suffix -డు (-*ḍu*). This suffix specifically marks the action as performed by a male. Below are examples:

- అతను చదివాడు (*Atanu Chadivāḍu*) - He read.
- రాము వచ్చాడు (*Rāmu Vacchāḍu*) - Ramu came.
- వీడు నిన్న బాగా మాట్లాడాడు (*Vīḍu Ninna Bāga Māṭlāḍāḍu*) - He spoke well yesterday.

2. Feminine Subject (స్త్రీలింగము) - ***Strīlingamu***

For a feminine subject, the verb typically ends with the suffix -ది (-*di*) or -ంది (-*ndi*). These suffixes indicate that the action is performed by a female. Examples:

- ఆమె చదివింది (*Āme Chadivindi*) - She read.
- సీత వచ్చింది (*Sīta Vacchindi*) - Sita came.
- ఆమె నిన్న బాగా మాట్లాడింది (*Āme Ninna Bāga Māṭlāḍindi*) - She spoke well yesterday.

Key Features of Gender Agreement in Telugu Verbs

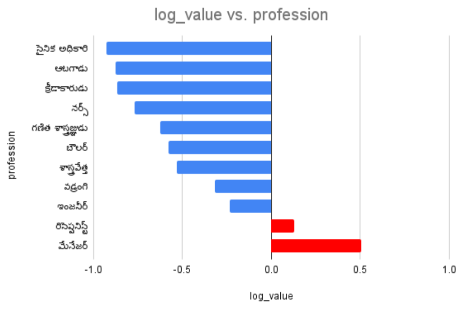
1. **Suffix Patterns:**
   - Masculine: Verbs generally end with -డు (-*ḍu*).
   - Feminine: Verbs generally end with -ది (-*di*) or -ంది (-*ndi*).
2. **Use of Pronouns:** Pronouns often clarify the subject's gender and help identify the corresponding verb form.
   - Masculine: అతను (*Atanu*) - He
   - Feminine: ఆమె (*Āme*) - She

Bias score calculation

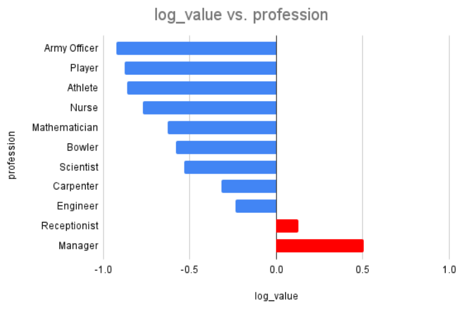
In Telugu, by observing the verb suffix and understanding the subject, one can identify the gender association of the verb. So we'll count number of occurances of verbs ending with -డు (-*ḍu* to be associated with male subjects and verbs ending with -ది (-*di*) or -ంది (-*ndi* to be associate with female subjects. We'll use telugu-navarasa model, which is SOTA model for telugu language for analysis of bias.

Results

In figure2, we've plotted the results of the bias-score v/s professions. We've performed this experiment in similar settings we used in Hindi language bias, by writing short story on set of 120 neutral professions. Out of 120, 90 professions were biased towards male and 6 professions were biased towards females.



(a) Bias score calculation in Telugu lanuage



(b) English translation of fig-2a

Figure 2: Bias score calculation in Telugu language using Gendered verbs counting

2 Bias score Background Research

You had asked the questions regarding the standard resources for the 120 neutral professions and our metric that we came up with for the bias. Our metric for bias calculation:

$$bias = log\left(\frac{female\_count}{male\_count}\right)$$

The concept of counting tokens specific to masculine and feminine gender is taken from the BOLD[1] paper and this log method proposed is very similar to the bias-metric calculated by Neeraja et. al.(2022)[2] for performing analysis on Indic models. In this paper, for quantifying gender bias, they follow a technique similar to Bartl et. al.(2020)[3] using the MuRIL model. They create two versions of each sentence: one with the gendered noun masked (Person Masked - PM) and one with both the person and profession masked (Person + Profession Masked - PPM). they then calculate the probabilities of the masked token being the person in each version. The Occupational Gender Bias (OGB) is calculated as the log ratio of these probabilities, indicating the model's association bias.
The 120 neutral professions that we took for the analysis of bias is taken from the suggestions made by Neeraja et. al.(2022)[2][5] and Pushpdeep[6] .Combining all of the

professions from these resources, there were some professions for which the LLMs were not able to understand and response properly so we removed those professions, after filtering we got 120 professions for our analysis.

3 Bias Analysis in English

You also suggested us to explore about how the gender bias thing is being done in English. In English there have been significant research about analyzing gender bias, some of the popular methods are given below:

- **Bias in Descriptive Word Selection:**[7]
  We prompt the model with gender-specific profiles, such as ”a 20-year-old male doctor” and ”a 20-year-old female doctor.” The model generates a list of descriptive words for each prompt. By comparing the words used to describe the male and female profiles, we identify patterns of bias in how the model selects traits for different genders.

- **Bias in Gendered Role Selection:**[7]
  This method involves providing the model with incomplete sentences and asking it to fill in the blank with either ”he” or ”she.” For example, we prompt the model with sentences like :
  ”_____ is the most empathetic person I have ever seen.” The model’s choice of pronoun indicates its tendency to associate certain traits (e.g., empathy or intelligence) with specific genders.

- **Bias in Dialogue Topics:**[7] We generate dialogues between different gender combinations, such as between a man and a woman or between two women. The content and tone of these conversations are analyzed to reveal any stereotypical roles or topics the model assigns based on gender. For example, it may generate a complaint-focused dialogue for women but a more neutral conversation for men.

- **WinoBias for Gender Bias Detection:**[8] WinoBias tests gender bias in coreference resolution systems using a dataset of 3,160 sentences based on 40 occupations. Occupations are paired with male or female pronouns, reflecting gender stereotypes (e.g., linking ”nurse” with ”she” is prostereotypical, while linking with ”he” is anti-stereotypical). Sentence Types:

  1. **Type 1 (Challenging):** Requires world knowledge, no syntactic cues (e.g., ”[Person1] interacts with [Person2] and [pronoun] did [action]”).
  2. **Type 2 (Easier):** Uses syntactic and semantic cues (e.g., ”[Person1] interacts with [Person2] and then [pronoun] did [action]”).

- **Gender Polarity Metrics for Bias Detection:**[1] Two gender polarity metrics are proposed to detect bias:

  – Unigram Matching:
    Counts the occurrences of male (e.g., ”he,” ”him”) and female (e.g., ”she,” ”her”) tokens. Text is labeled male if male tokens outnumber female tokens, and vice versa. If neither is present, the text is neutral.

– Word Embedding Projection:
  Projects words onto a gender direction (she-he) in a debiased Word2Vec space. Words are classified based on their alignment with gender. Text-level gender polarity is computed by averaging weighted scores (Gender-Wavg) or using the most polar word (Gender-Max). Thresholds classify the text as **male** (≤ -0.25) **or female** (≥ 0.25).

4 How are models weights stored in Airavat and Open-Hathi

Airavat and OpenHathi models are LLaMa models pretrained on Hindi Language and are stored in PyTorch's binary format. When the model is loaded, the weights are represented as tensors of the torch.float32 datatype by default, i.e. weights are represented in 32 bits.

5 How LLMs deployed on Mobile

For deploying LLM's on mobile,we need to quantize the existing models. Objcetive of quantization is to reduce memory cost while trying to maintain the same accuracy. Once model is quantized we need to cross compile our model to run on RAM and GPU's of mobile devices.

5.1 Model Quantization

Quantization is applied to model weights and activations. Most Indic LLM models are stored in **float32** datatype. We've used **Post-Training Quantization** technique which doesn't need to retrain the model while quantization is directly applied to pre-trained models.
Discrete low-bit values to approximate high-precision floating points in quantization. It involves 2 steps[10]:

1. Determining the quantization parameters (which consist of scale and zero point):

   - For asymmetric quantization, the scale(s) and zero-point(z) are computed as:

     $$s = \frac{max(X) - min(X)}{2^n - 1} \cdot c \qquad\qquad z = \frac{-min(X)}{s}$$

     where, X is the input tensor, n is the quantization bit-width, and c is the clipping factor used to reduce the dynamic range of quantization to mitigate the effect of outlier values.

   - For Symmetric quantization,

     $$s = \frac{2 * max(|X|}{2^n - 1} \cdot c$$

2. Calculating the quantized tensor:

- For Asymmetric Quantization:

$$\bar{X} = clamp\left(\frac{X}{s}, 0, 2^n - 1\right)$$

- For Symmetric Quantization:

$$\bar{X} = clamp\left(\frac{X}{s}, -2^n - 1, 2^n - 1\right)$$

Group Quantization: In per Channel Quantization, we calculate scale and zero-point for a row or a column of the tensor [11].Last dimension of input matrix is treated as a channel. Each channel can be further sub-divided into several groups, and quantization can be performed in each subgroup which is called per-group quantization[12]. Smaller the group size means more precise the representation is and more overhead(both memory and computation) is associated with it.

5.2 Android Deployment using MLC-LLM

MLC-LLM is a machine learning compiler and LLM deployment engine. This platform enabled us to deploy models natively on many platform like Android, iOS, web applications, python, cli, etc. After Model quantization step, we can cross compile the model weights to any native platform of choice.

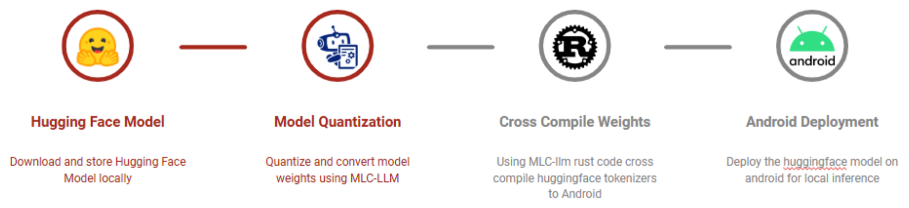

Hugging Face Model
Download and store Hugging Face Model locally
Model Quantization
Quantize and convert model weights using MLC-LLM
Cross Compile Weights
Using MLC-llm rust code cross compile huggingface tokenizers to Android
Android Deployment
Deploy the huggingface model on android for local inference
android

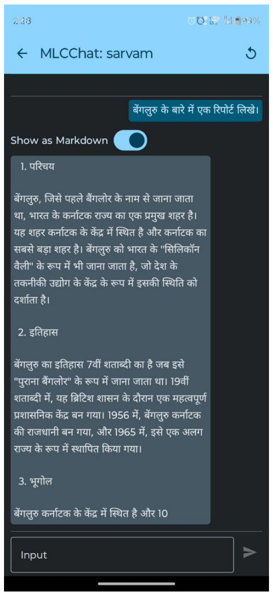
Figure 3: MLC-LLM workflow

5.2.1 Quantization

We've applied group quantization for quantizing sarvam-0.5 model using mlc-llm module. As suggested by Tim Dettmers et al.(2022) [9] we're performing 4-bit quantization of the model weights. In mlc-llm this format format of the code is qAfB(_id), where A represents the number of bits for storing weights and B represents the number of bits for storing activations. The _id is an integer identifier to distinguish different quantization algorithms (e.g. symmetric, non-symmetric, AWQ, etc). We've performed q4f16_1 quantization of our models **converting model** weights from float32 format to 4 bit format. This reduced the model size from 9GB to 1.5GB. This size reduction made it possible to be deployed in mobile device.

5.2.2 Deployment on Android

For conversion of mlc-llm 4 bit format model to Android-platform native runtime we've used RUST script provided in mlc-llm environment. Thereafter we've deployed model on a average smartphone: Motorola G35 mobile which costs around Rs.10,000. The screenshot of results are as shown in figure4



बेंगलुरु के बारे में एक रिपोर्ट लिखे।

1. परिचय

बेंगलुरु, जिसे पहले बैंगलोर के नाम से जाना जाता था, भारत के कर्नाटक राज्य का एक प्रमुख शहर है। यह शहर कर्नाटक के केंद्र में स्थित है और कर्नाटक का सबसे बड़ा शहर है। बेंगलुरु को भारत के "सिलिकॉन वैली" के रूप में भी जाना जाता है, जो देश के तकनीकी उद्योग के केंद्र के रूप में इसकी स्थिति को दर्शाता है।

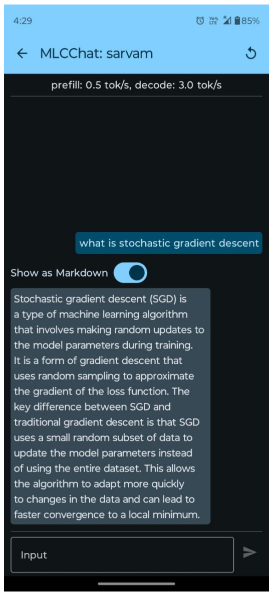
2. इतिहास

बेंगलुरु का इतिहास 7वीं शताब्दी का है जब इसे "पुराना बैंगलोर" के रूप में जाना जाता था। 19वीं शताब्दी में, यह ब्रिटिश शासन के दौरान एक महत्वपूर्ण प्रशासनिक केंद्र बन गया। 1956 में, बेंगलुरु कर्नाटक की राजधानी बन गया, और 1965 में, इसे एक अलग राज्य के रूप में स्थापित किया गया।

3. भूगोल

बेंगलुरु कर्नाटक के केंद्र में स्थित है और 10

(a) Hindi Prompt results



prefill: 0.5 tok/s, decode: 3.0 tok/s

what is stochastic gradient descent

Stochastic gradient descent (SGD) is a type of machine learning algorithm that involves making random updates to the model parameters during training. It is a form of gradient descent that uses random sampling to approximate the gradient of the loss function. The key difference between SGD and traditional gradient descent is that SGD uses a small random subset of data to update the model parameters instead of using the entire dataset. This allows the algorithm to adapt more quickly to changes in the data and can lead to faster convergence to a local minimum.

(b) English Prompt results

Figure 4: Quantized model generation on Motorola G35 Android smartphone

5.3 Other attempt of deploying LLMs on mobile

Initially we tried with the traditional methods of deploying LLMs to mobile which includes the conversion of the model to TFLite format, and TFLite format of the model is compatible with the mobile device. The steps involved in this method are:

1. Model Selection
2. Truncate and quantize the model
3. Convert the model to ONNX format
4. Convert ONNX to TFLite format using tool onnx2tf
5. Deploy the TFLite model on Mobile

In this method, the major issue that we found was, when converting the model to ONNX format using `torch.onnx.export`(step 3), this onnx is supported in the torch version lesser than 2.0, so we installed torch 1.13 stable version then after that the issue arised of that the numpy should also be version lesser than 2, after resolving all this issue one by one at the end we get that this version of pytorch is not supported by your CUDA version, we need to downgrade the version of CUDA that we are using, and there we got stuck because we don't have the permission to downgrade the CUDA version of our server. After this, we tried the completely different approach using MLC-LLM and using that we were able to run and deploy the model locally on Mobile device.

6 What are benefits of LLM's deployed on mobile.

Here are key benefits of running LLMs on mobile phones :

1. **Local Language Support:** Helps families interact in their native languages like Hindi, Tamil, or Bengali, enabling voice assistance, translations, and easy access to information in regional languages.
2. **Offline Accessibility:** Works without internet, reducing data costs and making it usable in areas with poor connectivity or for families on tight budgets.
3. **Elderly and Illiteracy-Friendly:** Simplifies technology for elderly family members and those who cannot read or write, using voice-based interactions in their preferred language.
4. **Privacy and Security:** Keeps sensitive information like health questions or financial details private by processing data directly on the device.
5. **Educational and Daily Assistance:** Supports children with homework, explains government schemes, and helps with daily tasks like writing grocery lists or finding recipes.

References

[1] Dhamala, Jwala (2021), BOLD: Dataset and Metrics for Measuring Biases in Open-Ended Language Generation. Link:https://arxiv.org/abs/2209.03661

[2] Neeraja Kirtane, V Manushree, Aditya Kane (2022), Efficient Gender Debiasing of Pre-trained Indic Language Models. Link:https://arxiv.org/pdf/2101.11718

[3] Marion Bartl, Malvina Nissim, Albert Gatt (2020), Unmasking Contextual Stereotypes: Measuring and Mitigating BERT's Gender Bias. Link: https://aclanthology.org/2020.gebnlp-1.1/

[4] Su Lin Blodgett, Solon Barocas, Hal Daumé III, Hanna Wallach (2020), Language (Technology) is Power: A Critical Survey of "Bias" in NLP.

[5] Neeraja Kirtane, Tanvi Anand (2022), Mitigating Gender Stereotypes in Hindi and Marathi. Link: https://aclanthology.org/2022.gebnlp-1.16.pdf

[6] Pushpdeep Singh (2023), Don't Overlook the Grammatical Gender: Bias Evaluation for Hindi-English Machine Translation. Link: https://arxiv.org/pdf/2312.03710.

[7] Jinman Zhao1, Yitian Ding2, Chen Jia3, Yining Wang1, Zifan Qian4Gender (2024), Bias in Large Language Models across Multiple Languages. Link: https://arxiv.org/pdf/2403.00277v1

[8] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, Kai-Wei Chang (2018), Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods. Link: https://arxiv.org/abs/1804.06876

[9] Tim Dettmers, Luke Zettlemoyer (2022), The case for 4-bit precision: k-bit Inference Scaling Laws.Link: https://arxiv.org/abs/2212.09720

[10] Yilong Zhao 1 2, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, Baris Kasikci (2024), ATOM: LOW-BIT QUANTIZATION FOR EFFICIENT AND ACCURATE LLM SERVING. Link: https://arxiv.org/pdf/2310.19102

[11] Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant (2023),Accurate and efficient post-training quantization for large language models.Link: https://arxiv.org/abs/2211.10438

[12] Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., Gan, C., and Han, S.(2023). Awq: Activation-aware weight quantization for llm compression and acceleration. Link: https://arxiv.org/abs/2306.00978