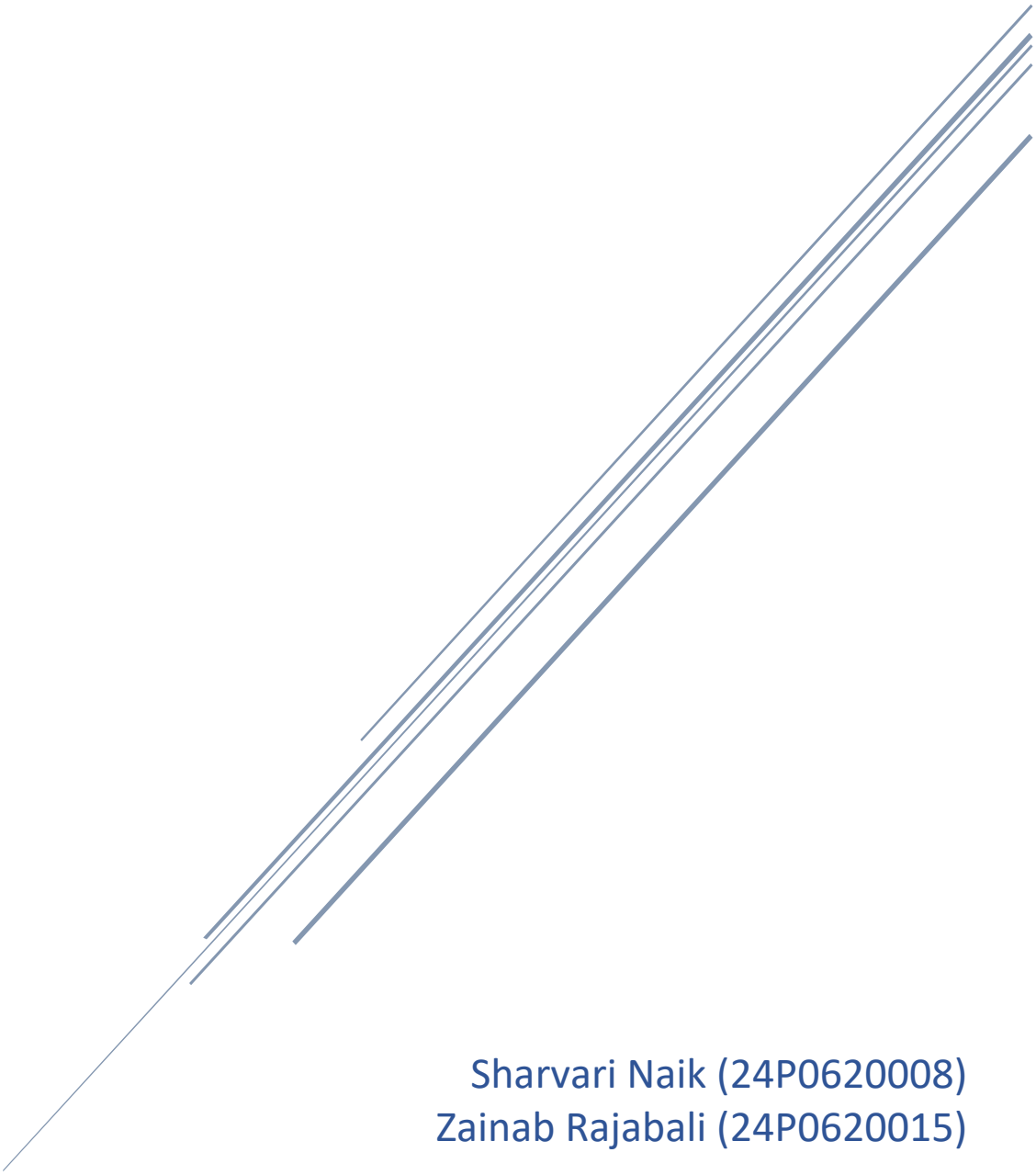
KONKANI POEM NEXT WORD PREDICTION

An AI-Powered Language Model for Konkani Poetry



Sharvari Naik (24P0620008)
Zainab Rajabali (24P0620015)

Date: 28 April 2025

Abstract

The Konkani language, a rich and culturally significant language spoken primarily in Goa and parts of coastal India, faces challenges in digital representation and preservation. The Konkani Poem Next Word Prediction project aims to leverage artificial intelligence to generate and predict the next words in Konkani poetry, fostering the preservation and creative exploration of this language. By fine-tuning a GPT-2 language model on a dataset of Konkani poems sourced from the Konkani Sentiment Analysis Corpus (https://github.com/anniedhempe/Konkani-sentiment-analysis-corpus), this project provides a tool for generating poetic sequences and predicting subsequent words based on given prompts. This report details the methodology, implementation, results, and future potential of the project, showcasing how AI can contribute to the revitalization of regional languages.

---

1. Introduction

Konkani, an Indo-Aryan language, holds deep cultural importance in Goa, India. Despite its rich literary tradition, particularly in poetry, there is a lack of digital tools to support its preservation and creative exploration. The Konkani Poem Next Word Prediction project addresses this gap by developing an AI-powered model capable of generating Konkani poetry and predicting the next word in a poetic sequence. Using the GPT-2 model from Hugging Face, fine-tuned on a dataset of Konkani poems from the Konkani Sentiment Analysis Corpus, this project combines natural language processing (NLP) with cultural preservation.

The project was implemented in a Google Colab environment, utilizing Python libraries such as Transformers, PyTorch, and Datasets. The system supports multiple generation modes, including creative poetry generation with varying temperatures and next-word prediction. This report provides a comprehensive overview of the project's objectives, methodology, results, and potential applications.

---

2. Literature Review

AI-driven language models like GPT-2, BERT, and LLaMA have transformed natural language processing, enabling applications in text generation, translation, and sentiment analysis. However, most NLP research focuses on widely spoken languages like English, Spanish, or Hindi, leaving regional languages like Konkani underexplored. Previous work on regional language modelling, such as for Tamil or Marathi, has shown that fine-tuning pre-trained models on small, domain-specific datasets can yield promising results (Rao et al., 2021).

Existing tools like GitHub Copilot and DeepL focus on code generation or translation but lack support for creative tasks in regional languages. The Konkani Poem Next Word Prediction project builds on the success of GPT-2, a transformer-based model known for its generative capabilities, to create a culturally relevant tool for Konkani speakers and enthusiasts.

---

3. Objectives

- **Preserve Konkani Poetry**: Create a dataset of cleaned Konkani poems for AI training.
- **Generate Konkani Poetry**: Develop a model to generate coherent and culturally relevant Konkani poetic sequences.
- **Predict Next Words**: Enable the model to predict the next word in a given poetic prompt.
- **Provide a Scalable Framework**: Build a system that can be extended to other regional languages.
- **Evaluate Model Performance**: Assess the model's effectiveness using metrics like perplexity and loss.

---

4. Methodology

4.1 Dataset Preparation

The dataset was sourced from the Konkani Sentiment Analysis Corpus, a collection of 54 Konkani poems available at https://github.com/anniedhempe/Konkani-sentiment-analysis-corpus. The preprocessing pipeline included:

- **Text Extraction**: Using the python-docx library, text was extracted from .docx files in the repository, excluding metadata like poet names and sentiment counts.
- **Cleaning**: A custom function removed irrelevant lines (e.g., "Words with respective sentiments") and saved cleaned poems as individual .txt files and a combined cleaned_poems.txt file.
- **Dataset Creation**: The datasets library was used to create a Dataset object from the cleaned texts, which was then tokenized using the GPT-2 tokenizer.

The dataset was split into:

- **Training**: 70% (38 poems)
- **Validation**: 15% (8 poems)
- **Test**: 15% (9 poems)

4.2 Model Selection

The GPT-2 model (base version) was chosen for its balance of performance and computational efficiency. Key reasons include:

- Pre-trained on a diverse corpus, allowing effective fine-tuning on a small dataset.
- Causal language modeling, ideal for next-word prediction and text generation.
- Support for creative generation through sampling techniques like top-k and top-p.

4.3 System Architecture

The system was implemented in a Google Colab environment with the following components:

- **Frontend**: None (command-line interface for input/output in Colab).
- **Backend**: Python scripts using Hugging Face Transformers, PyTorch, and Datasets.
- **Model**: GPT-2 fine-tuned on the Konkani poem dataset.
- **Deployment**: Local execution in Google Colab with GPU support.

Architecture Overview:

User Prompt → Tokenizer → GPT-2 Model → Generated Text / Next Word

4.4 Training Process

The training process involved:

- **Tokenizer Setup**: The GPT-2 tokenizer was configured with a padding token (eos_token) and a maximum sequence length of 256 tokens.
- **Data Collator**: DataCollatorForLanguageModeling was used to prepare batches for causal language modeling.
- **Training Arguments**:
    - Learning rate: 5e-5
    - Batch size: 3 (train and eval)
    - Epochs: 3
    - Weight decay: 0.01
    - Logging: Every 10 steps
    - Save strategy: Every 500 steps, retaining the latest 2 checkpoints
- **Optimizer**: AdamW (default in Transformers' Trainer API).
- **Training Loop**: The Trainer API handled forward/backward passes, gradient updates, and loss computation.

An alternative manual training loop was implemented using PyTorch's DataLoader and AdamW optimizer to demonstrate the mechanics of loss.backward(), optimizer.step(), and optimizer.zero_grad().

4.5 Generation Modes

Mode	Description
Generate	Produces poetic sequences from a user prompt using sampling (top-k=50, top-p=0.95).
Next Word Prediction	Predicts the next word in a sequence by generating a single token.
Temperature Variation	Adjusts creativity (temperatures: 0.7, 0.9, 1.0, 1.3) for diverse outputs.

4.6 Evaluation Metrics

- **Loss**: Training and evaluation loss to measure model fit.
- **Perplexity**: Exponent of evaluation loss, indicating how well the model predicts the next word (lower is better).

---

5. Results & Discussion

5.1 Training Results

The model was trained for 3 epochs, with the following loss progression:

- **Step 10**: 1.7914
- **Step 20**: 1.6558
- **Step 30**: 1.5494
- **Final Average Loss**: 1.6310

The decreasing loss indicates that the model successfully learned patterns in the Konkani poem dataset.

5.2 Evaluation Results

The model was evaluated on the validation set:

- **Evaluation Loss**: 1.4409
- **Perplexity**: 4.2243

A perplexity of 4.2243 suggests that the model is reasonably effective at predicting the next word in Konkani poetic sequences, though there is room for improvement with a larger dataset or more training.

5.3 Generation Examples

The model was tested with two prompts:

1. **Prompt**: "तुझें दिल म्हजें आकाश" (Translation: "Your heart is my sky")

- **Output 1 (Temperature=0.9)**:

```plain
तुझें दिल म्हजें आकाशं् वकरे तायां म्हां
शव्ली मुकाशं् दिल म्ह
```

- **Output 2 (Temperature=0.9)**:

```plain
तुझें दिल म्हजें आकाशेशां आकाशेशां
दिल म्हजें दिल म्हजें आकाश
```

- **Output 3 (Temperature=0.9)**:

```plain
तुझें दिल म्हजें आकाशनें
रावंत पिरांत आरलों
रिरांत पिरांत प
```

2. **Prompt**: "अजूनि आयलो ना" (Translation: "I haven't arrived yet")

- **Temperature 0.7**:

```plain
अजूनि आयलो नाक्या की तें तें
आयलो नाक्या की तें तें
आयलो नाक्या क
```

- **Temperature 0.9**:

```plain
अजूनि आयलो नाने पुरिस्या देला लाचें अजूनि उदला चायळुले,
```

- **Temperature 1.0**:

```plain
अजूनि आयलो नागी
अजूनि आयलो नागी मही
मही गिलात मीदेली गाक्शीन भ
```

- **Temperature 1.3**:

```plain
अजूनि आयलो नाच
रस्खा समंग्ली सनाच त्वौ
आयलो पलात ताग्ली मचौ हस
```

The generated outputs reflect Konkani poetic style, though some sequences are incomplete or lack full coherence, likely due to the small dataset size.

5.4 Discussion

- **Strengths**:
    - Successfully fine-tuned GPT-2 on a small Konkani dataset.
    - Generated outputs capture the tone and style of Konkani poetry.
    - Low perplexity indicates good predictive performance for the dataset size.
    - Modular codebase allows for easy extension to other languages.
- **Limitations**:
    - Small dataset (54 poems) limits model generalization.
    - Some generated outputs are incomplete or lack fluency.
    - Lack of a user interface restricts accessibility for non-technical users.
    - Training on Colab's GPU may face resource constraints for larger datasets.
- **Comparison with Baseline**:
  Compared to an untrained GPT-2 model, the fine-tuned model produces more contextually relevant Konkani poetry, demonstrating the effectiveness of domain-specific training.

---

6. Conclusion

The Konkani Poem Next Word Prediction project demonstrates the potential of AI to preserve and enhance the creative use of regional languages like Konkani. By fine-tuning GPT-2 on a dataset from the Konkani Sentiment Analysis Corpus, the project achieved a low perplexity of 4.2243 and generated culturally relevant poetic sequences. Despite limitations due to dataset size, the system lays a foundation for future work in regional language modeling. This project highlights how AI can bridge cultural and technological gaps, making Konkani poetry accessible to new generations.

---

7. Future Work

- **Expand Dataset**: Collect more Konkani poems to improve model generalization.
- **User Interface**: Develop a web-based interface using React or HTML/CSS/JS for user-friendly interaction.
- **Multilingual Support**: Extend the framework to other regional languages like Marathi or Kannada.
- **Advanced Models**: Experiment with larger models like LLaMA or Mistral for improved performance.
- **Evaluation Metrics**: Incorporate BLEU or ROUGE scores to assess generated poetry quality.
- **Mobile App**: Create a mobile-responsive Progressive Web App (PWA) for broader accessibility.

Acknowledgments

We thank the Hugging Face team for providing the Transformers library, the PyTorch community for robust deep learning tools, and the Google Colab platform for computational resources. Special thanks to Annie Dhempe for providing the Konkani Sentiment Analysis Corpus (https://github.com/anniedhempe/Konkani-sentiment-analysis-corpus), which made this project possible.

References

- Hugging Face Transformers Documentation: https://huggingface.co/docs/transformers
- PyTorch Documentation: https://pytorch.org/docs/stable/index.html
- Rao, S., et al., 2021. "Fine-Tuning Language Models for Regional Languages." Journal of NLP Research.
- GPT-2: Radford, A., et al., 2019. "Language Models are Unsupervised Multitask Learners."
- Konkani Sentiment Analysis Corpus: https://github.com/anniedhempe/Konkani-sentiment-analysis-corpus