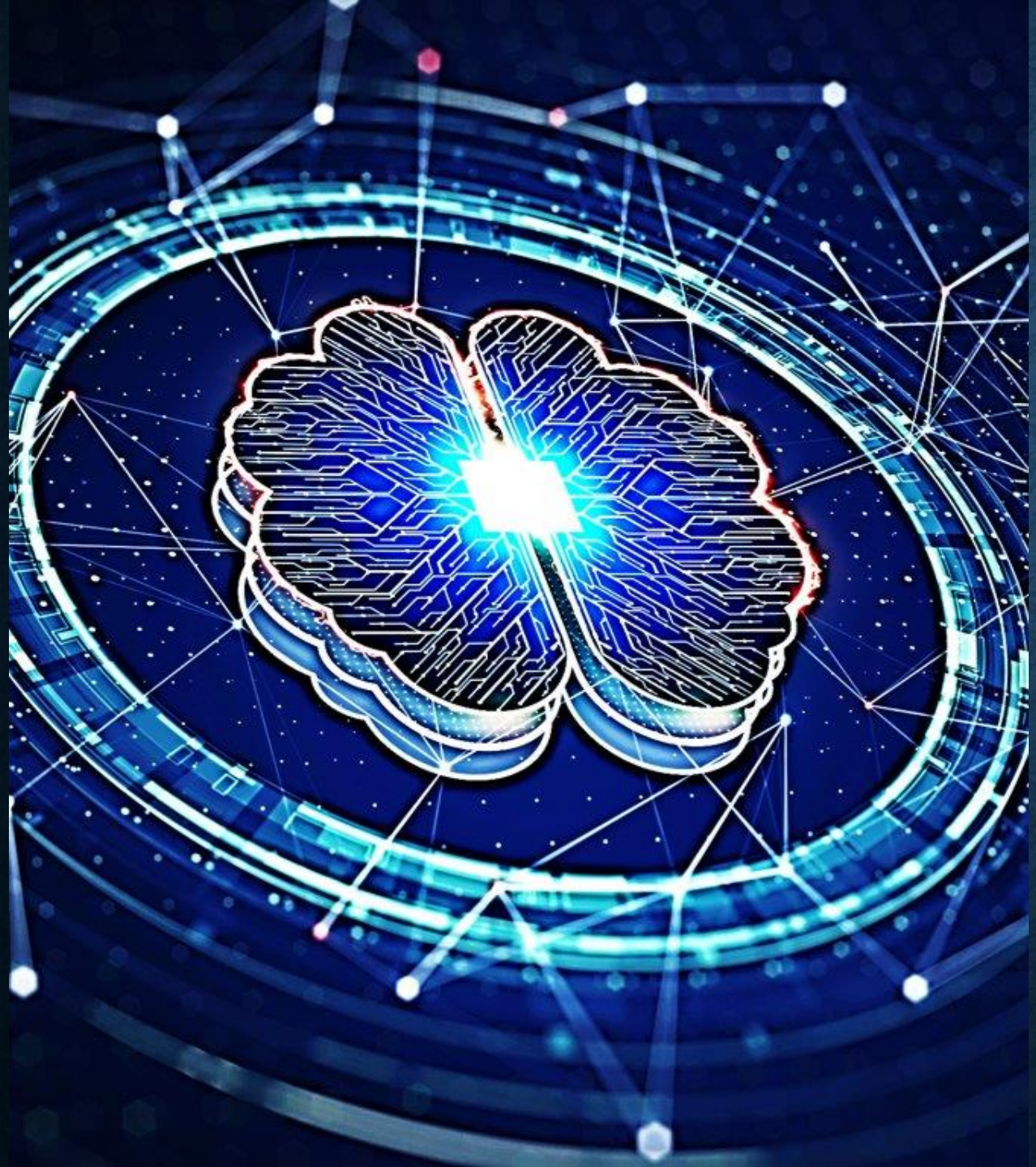


NEURAL MACHINE TRANSLATION FROM ENGLISH TO DRAVIDIAN LANGUAGES

VISHNU CHOUNDUR

NARESH BOLISHETTY

VISHNU TEJA YERRAPRAGADA

AGENDA

- Introduction
- Objective
- Methodology
- Dataset
- Challenges Faced
- Results and Analysis
- Conclusion
- Reference

INTRODUCTION

- Neural Machine Translation (NMT) has revolutionized the field of automated translation by providing more accurate and context-aware translations compared to traditional rule-based and statistical methods.
- Advancing machine translation for Dravidian languages.
- Novel methodologies in neural machine translation (NMT).
- The development of multilingual models like the MT5, a variant of the T5 (Text-to-Text Transfer Transformer) model designed to handle multiple languages, has further expanded the potential of NMT systems.
- Focus on low-resource language translation
- This introduction explores the application of the MT5 model for translating English into Dravidian languages—a group of languages primarily spoken in South India, including Tamil, Telugu, Kannada, and Malayalam.

OBJECTIVE

The key objectives of the paper as identified from the document:

1. **Address Limited Resources**: Tackle the challenge of developing robust bilingual machine translation systems, particularly under limited resource constraints which are common with low-resource languages like the Dravidian family .
2. **Focus on Dravidian Languages**: Specifically target the translation of English into other Dravidian languages such as Tamil, Telugu. This includes the development of machine translation systems tailored to these languages .
3. **Experiment with Pre-trained and Fine tuning**: Apply pre-trained model to finetuning techniques as a method of data augmentation to enrich training data, particularly for languages with limited monolingual data .
4. **Evaluate with BLEU Scores**: Use the BLEU score metric to evaluate the accuracy of the translation models, providing a quantitative measure of translation quality across different language pairs.

NEURAL MACHINE TRANSLATION

NMT is a type of machine translation that relies on artificial neural networks to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model.

NMT utilizes artificial neural networks to predict word sequences, offering an integrated model for entire sentences.

Compared to rule-based and statistical methods, NMT yields more fluent and accurate translations by capturing linguistic nuances.

While superior in output quality, NMT requires substantial data and computing power for training.

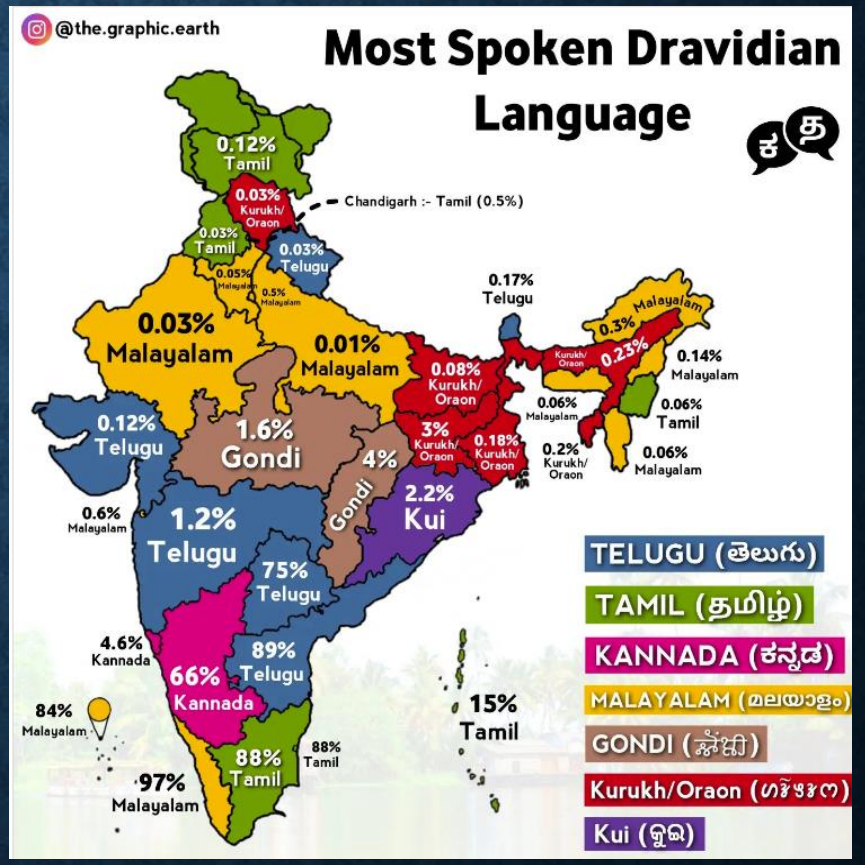
Significantly advances machine translation technology, underpinning many online translation services today.

IMPORTANCE OF NMT FOR DRAVIDIAN LANGUAGES

Facilitates seamless communication across Dravidian languages, vital for the linguistic diversity in South Asia.

Acts as a digital preservation tool for the literary and cultural heritage of these languages.

Ensures linguistic representation in the digital age, providing access to information and technology in native languages.



CHALLENGES IN NMT FOR LOW-RESOURCE LANGUAGES

Some languages support Homograph words whose meaning changes with context.

- E.g.: In Heart Attack and Dog attacks cat , attack is the homograph word.

Some languages have multiple scripts.

- E.g.: Punjabi (Gurmukhi, Shahmukhi).

Grammatical variations between languages lead to ambiguity.

Judging of speakers intention is difficult. Meanings of sentences or words vary with the speakers intention (like sarcasm, sentiment, metaphor, etc.).

Code-Mixed language processing is challenging as user uses multiple languages in a sentence or an utterance.

- E.g.: User tweet : “listening to Bombae Haelutaitae from Rajakumara”

DATA SET

```
DatasetDict({
    train: Dataset({
        features: ['translation'],
        num_rows: 4946035
    })
    validation: Dataset({
        features: ['translation'],
        num_rows: 1000
    })
    test: Dataset({
        features: ['translation'],
        num_rows: 2390
    })
})
```

Te

```
dataset

DatasetDict({
    train: Dataset({
        features: ['translation'],
        num_rows: 1000000
    })
    validation: Dataset({
        features: ['translation'],
        num_rows: 1000
    })
    test: Dataset({
        features: ['translation'],
        num_rows: 2390
    })
})
```

```
dataset

DatasetDict({
    train: Dataset({
        features: ['translation'],
        num_rows: 5264867
    })
    validation: Dataset({
        features: ['translation'],
        num_rows: 1000
    })
    test: Dataset({
        features: ['translation'],
        num_rows: 2390
    })
})
```

Ta

```
dataset['train'][0]

{'translation': {'en': 'Have you heard about Foie gras?',
  'te': 'ఇక ఫ్రూట్ ఫ్లైస్ గురించి మీరు విన్నారా?'}}
```

```
dataset['train'][0]

{'translation': {'en': 'Some 14 months later, the second calf is born.',
  'ta': 'சுமார் 14 மாதங்கள் கழித்து, இரண்டாம் கன்றை ஈனுகிறது.'}}
```

	source	target
518947	Russia then followed suit.	அதை ரஷ்யாவும் பின்பற்றத் தொடங்கி விட்டது.
67190	What do you do with a naughty child?	பீற்றரின்ட மகனோட உனக்கென்னகூட்டு?
385463	What is your mantra?	முகென் என்ன உன் சொத்தா?
221209	"They said, ""You are only of those affected b...	"""நீரும் எங்களைப் போன்ற ஒரு மனிதரேயன்றி (வேற...
536990	Resolutions were passed in this regard.	இதைத் தொடர்ந்து தீர்மானங்கள்முன்மொழியப்பட்டன.
...	...	...
442223	Disaster relief teams are working towards resc...	காணாமல் போனவர்களை மீட்கும் நடவடிக்கையில் பேரிட...
967952	The lagoon communities, however, rebelled and ...	ஒரு குடியரசின் பிறப்பும் எழுச்சியும்
792224	America first	அமெரிக்கா முதலிடத்தில்

PROPOSED METHODOLOGY

Pre-Trained, Fine-Tuning

METHODOLOGY

1. Environment Setup and Data Preparation

- ✓ Installation of Necessary Libraries: The script begins by installing Python libraries such as transformers, sentencepiece, datasets, and sacrebleu that are essential for working with the MT5 model and handling language data.
- ✓ Data Loading: Data files for training, validation, and testing are specified. Each file pair consists of parallel texts in English and Tamil. The script reads these files and creates a datasets.Dataset object for each set, ensuring the number of lines in English files matches those in Tamil to maintain accurate translation pairs.

2. Model and Tokenizer Initialization

- ✓ MT5 Model Loading: The MT5 model and its corresponding tokenizer are loaded from Hugging Face's transformers library. This setup is critical for processing text in a format suitable for the model.
- ✓ CUDA Configuration: If a GPU is available (checked via torch.cuda.is_available()), the model is configured to use it, enhancing the training speed and efficiency.

METHODOLOGY

3. Data Encoding and Preparation

- ✓ Text Encoding: The English and Tamil texts are tokenized using the MT5 tokenizer. The tokenized texts are then converted into tensors that the model can process. This step includes setting a maximum sequence length and ensuring the data fits this constraint by truncation or padding.
- ✓ Data Batching: The tokenized data is divided into batches. This method improves the model's learning by not overwhelming it with too much data at once and helps in managing memory usage effectively.

4. Model Training

- ✓ Optimization Setup: An AdamW optimizer and a learning rate scheduler are configured to update the model's weights efficiently during training.
- ✓ Training Loop: The model undergoes training over multiple epochs, where it learns to translate from English to Tamil. Training involves feeding batches of data into the model, calculating the loss (how far the model's output is from the actual translation), and adjusting the model parameters to minimize this loss.
- ✓ Loss Tracking and Evaluation: The script tracks the training loss to monitor the model’s performance. Periodically, the model's translation ability is evaluated using a validation set to ensure that it generalizes well to new data.

METHODOLOGY

5. Model Saving and Loading

- ✓ Saving the Model: After training, the model and tokenizer are saved to disk. This step allows the model to be reused later without retraining from scratch.
- ✓ Loading the Model: The script includes functionality to reload the model and tokenizer from the saved files for further use or evaluation.

6. Evaluation and Translation

- ✓ BLEU Score Calculation: The model's translation quality is quantitatively evaluated using the BLEU score, a common metric in machine translation that compares machine-generated text to one or more reference translations.
- ✓ Translation Demonstration: The script showcases the model's translation capability by inputting English sentences and displaying the model-generated Tamil translations.

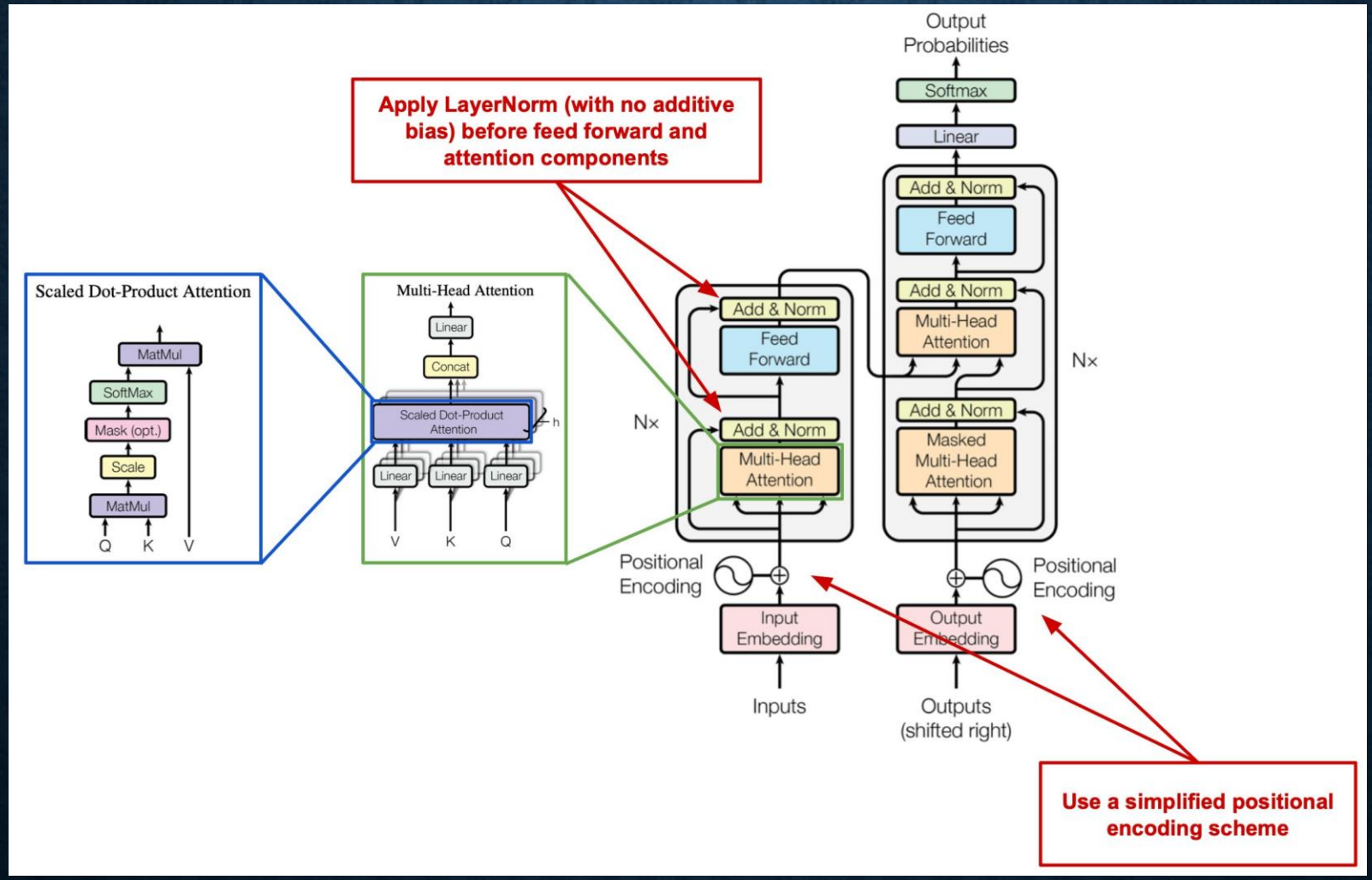
7. Visualization

- ✓ Loss Visualization: Plots the training loss over time to visually assess how the model's performance improved during training.

PRE-TRAINED MODEL

- **Introduction to mT5**:
  - ✓ Explain that mT5 is a multilingual extension of the T5 (Text-to-Text Transfer Transformer) model, developed by Google Research.
  - ✓ It is designed to handle tasks across multiple languages, capable of understanding and generating text in over 100 languages.
- Model Architecture:
  - ✓ Describe the architecture of mT5, which follows the Transformer model framework with an encoder-decoder structure.
  - ✓ Highlight modifications made from the original T5 to better support multilingual capabilities.

PRE-TRAINED MODEL



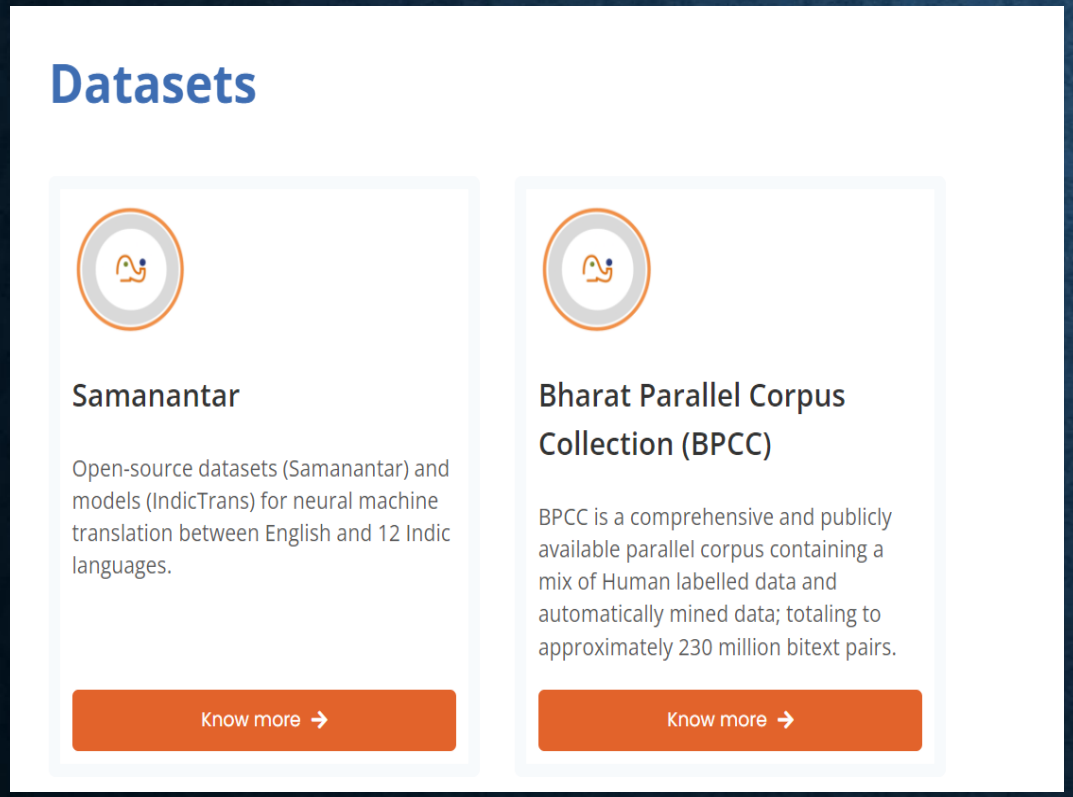
```
model_repo = 'google/mt5-small'
tokenizer = AutoTokenizer.from_pretrained(model_repo)
model = AutoModelForSeq2SeqLM.from_pretrained(model_repo)
```

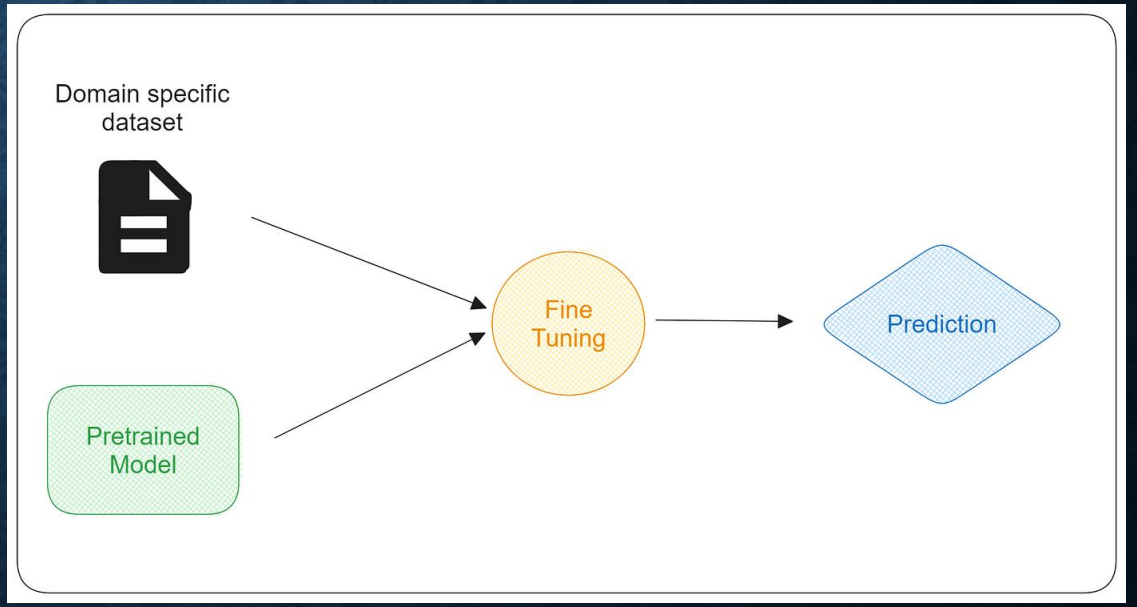
```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 82.0/82.0 [00:00<00:00, 6.84kB/s]
config.json: 100% 553/553 [00:00<00:00, 48.0kB/s]
spiece.model: 100% 4.31M/4.31M [00:01<00:00, 3.58MB/s]
special_tokens_map.json: 100% 99.0/99.0 [00:00<00:00, 7.53kB/s]
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, and simply means that the `legacy` (previous) behavior will be us
/usr/local/lib/python3.10/dist-packages/transformers/convert_slow_tokenizer.py:560: UserWarning: The sentencepiece tokenizer that you are converting to a fast tokenizer uses the byte fallback opt
  warnings.warn(
pytorch_model.bin: 100% 1.20G/1.20G [00:13<00:00, 95.0MB/s]
generation_config.json: 100% 147/147 [00:00<00:00, 13.5kB/s]
```

Attribute	T5 (Text-to-Text Transfer Transformer)	MT5 (Multilingual Text-to-Text Transfer Transformer)
Language Support	Primarily English, with potential adaptation to other languages.	Supports over 100 languages, inherently multilingual.
Training Data	C4 (Colossal Clean Crawled Corpus), predominantly English content.	mC4, a multilingual variant of C4, covering about 101 languages.
Task Flexibility	Versatile within English language tasks; requires fine-tuning for others.	Highly flexible across multiple languages without extensive retraining.
Use Cases	English text-based tasks such as summarization, question answering, text classification.	Multilingual tasks including translation, multilingual summarization, question answering across languages.
Model Complexity	Available in various sizes (small to XXL) tailored to computational needs and task requirements.	Similar variety in sizes, potentially higher resource requirements due to multilingual processing.
Ideal Environment	English-centric applications, academic research, single-language projects.	Global applications, multilingual environments, international businesses.

FINE TUNING

- Fine-tuning was key to leveraging resources for high translation accuracy.
- Adjusted hyperparameters, including learning rate, and used label smoothing to enhance model performance.





```
        avg_loss = np.mean(losses[-print_freq:])
        print('Epoch: {} | Step: {} | Avg. loss: {:.3f} | lr: {}'.format(
            epoch_idx+1, batch_idx+1, avg_loss, scheduler.get_last_lr()[0]))

    if (batch_idx + 1) % checkpoint_freq == 0:
        test_loss = eval_model(model, test_df)
        print('Test loss of {:.3f}'.format(test_loss))
        # torch.save(model.state_dict(), model_path)
```

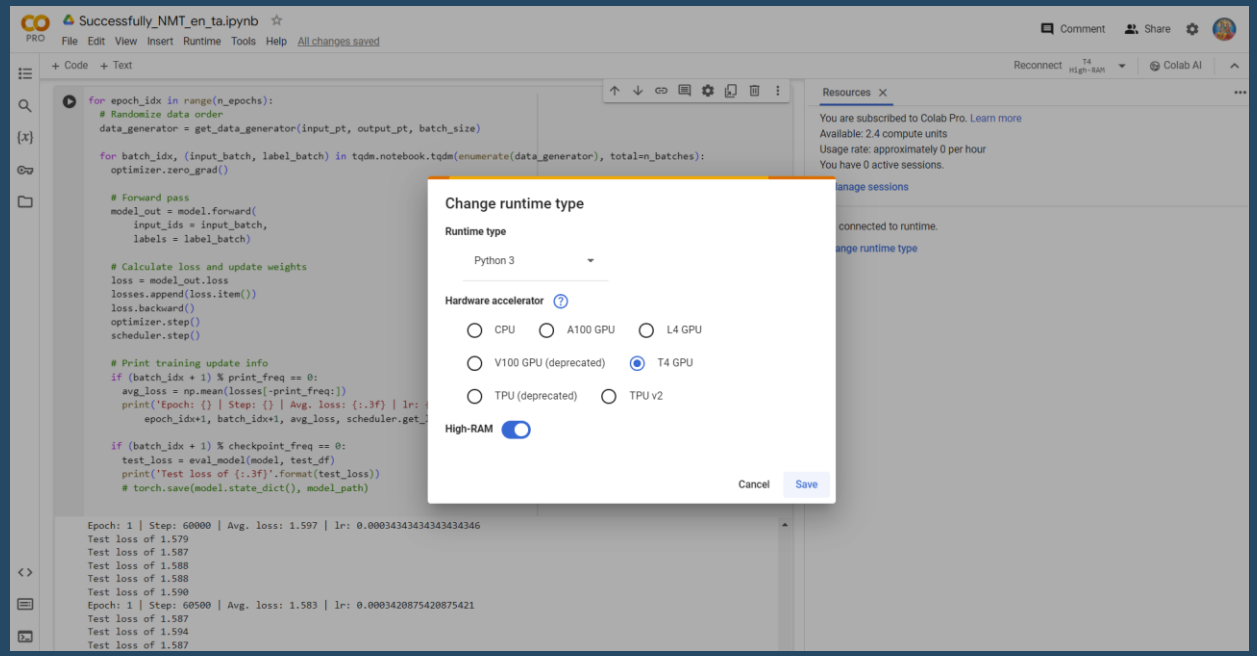
```
100% 62500/62500 [3:42:35<00:00,  2.91it/s]
Test loss of 24.793
Test loss of 14.088
Test loss of 8.770
Test loss of 6.631
Epoch: 1 | Step: 500 | Avg. loss: 15.634 | lr: 0.00013333333333333334
Test loss of 5.133
Test loss of 3.973
Test loss of 3.469
Test loss of 3.223
Test loss of 3.022
Epoch: 1 | Step: 1000 | Avg. loss: 3.559 | lr: 0.0002666666666666667
Test loss of 2.865
Test loss of 2.733
Test loss of 2.620
Test loss of 2.538
Test loss of 2.508
Epoch: 1 | Step: 1500 | Avg. loss: 2.602 | lr: 0.0004
Test loss of 2.468
Test loss of 2.440
Test loss of 2.410
Test loss of 2.393
Test loss of 2.377
Epoch: 1 | Step: 2000 | Avg. loss: 2.411 | lr: 0.0004996632996632996
Test loss of 2.351
Test loss of 2.312
```

HYPER-PARAMETER

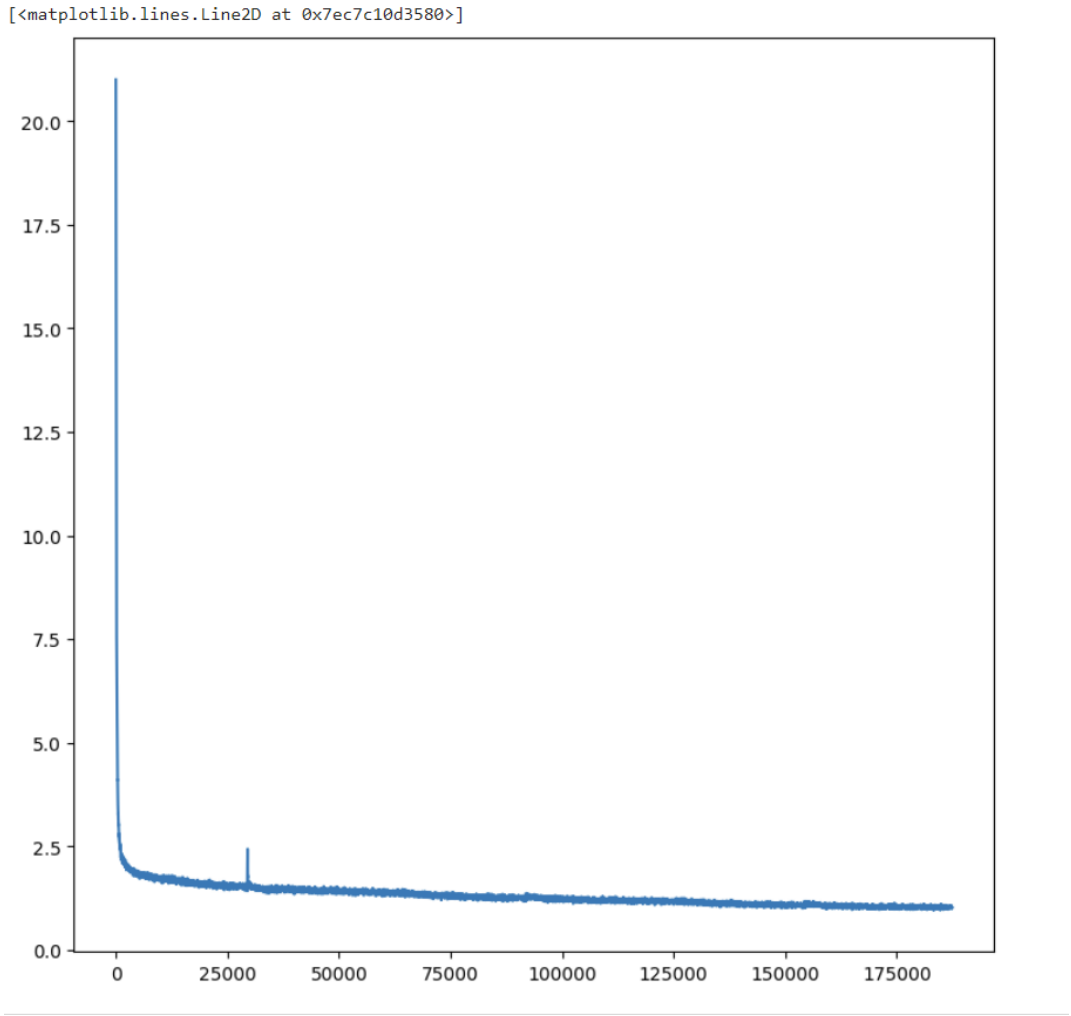
- Model Selection
  - ✓ Model_repo = 'google/mt5-small'
- Tokenizer
- Loss and Evaluation
  - ✓ Cross Entropy loss
- Additonal parameter
  - ✓ Num_beams = 10
  - ✓ Max_length = 20
- Optimizer and Scheduler
  - ✓ Optimizer = AdamW(model.parameters(), lr=lr)
  - ✓ Scheduler = get_linear_schedule_with_warmup(optimizer, n_warm_steps, total_steps)
- Device Management
  - ✓ CUDA
- Training parameter
  - ✓ N_epochs = 3
  - ✓ Batch_size = 16
  - ✓ Lr = 5e-4
  - ✓ N_warmup_steps = int(total_steps * 0.01)
  - ✓ Total_steps = n_epochs * n_batches
  - ✓ N_batches = int(ceil(len(train)/batch_size))

REQUIREMENTS

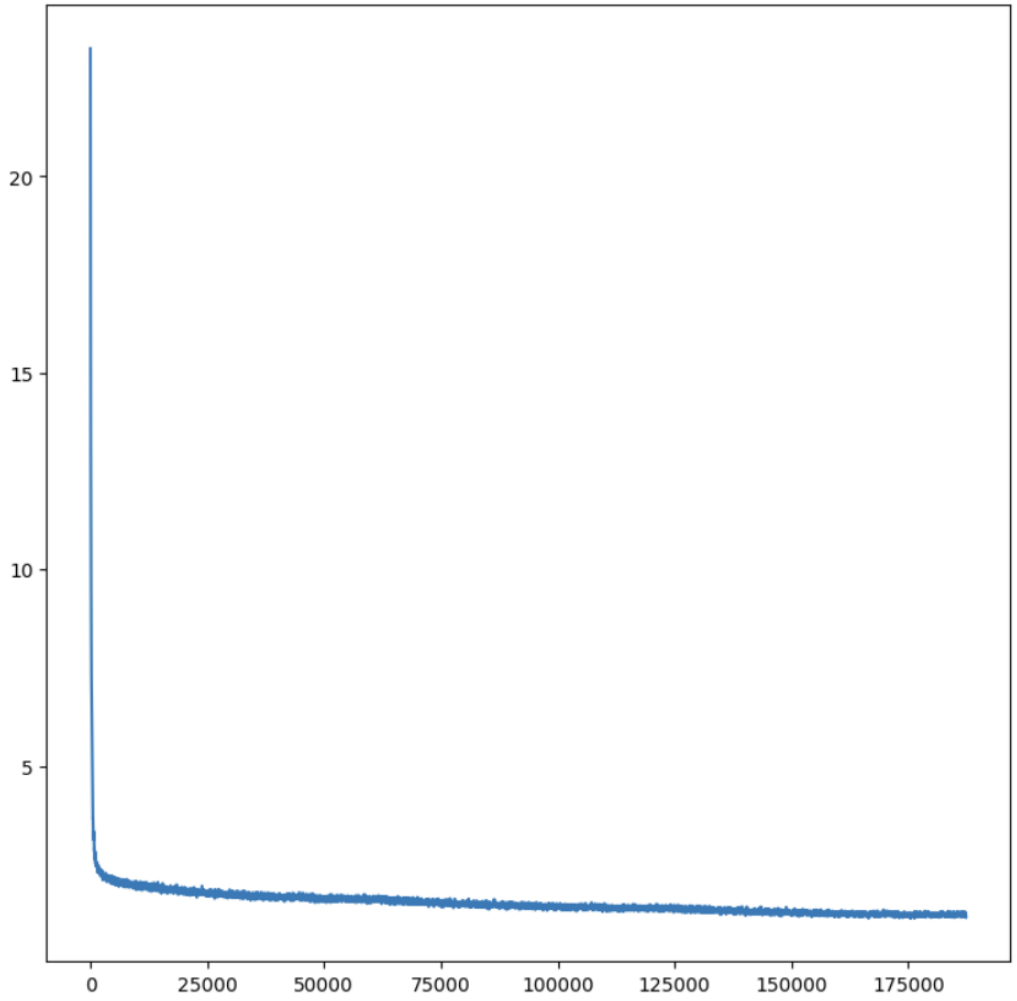
- Google Colab Pro
- GPU Module (T4 GPU) or L4 GPU
- Google Drive



English – Telugu Losses



English – Tamil Losses



RESULTS

TAMIL

```
# Evaluation
test_bleu_score = evaluate_bleu(model, tokenizer, test_df.sample(1), model.config.max_length)
print(f"BLEU score on test data: {test_bleu_score.score}")
```

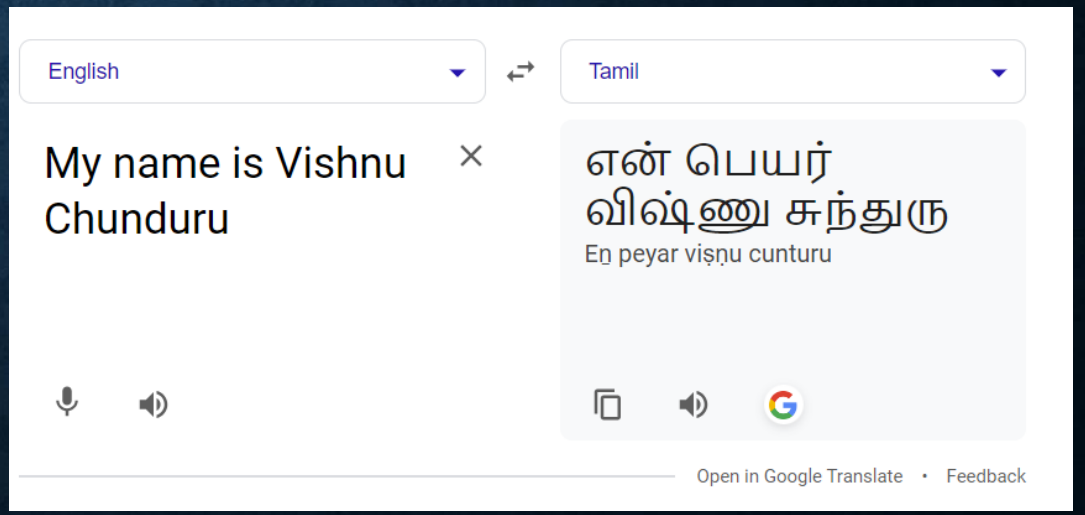
```
BLEU score on test data: 36.55552228545123
```

```
Input:
Raw input text in English: She took pictures and posted it on her Instagram profile.
Expected text in Tamil: அதனை புகைப்படமாக எடுத்து தன்னுடைய இன்ஸ்டாகிராம் பக்கத்தில் பதிவிட்டிருக்கிறார்.

Output:
அந்த புகைப்படங்களை தனது இன்ஸ்டாகிராம் பக்கத்தில் பதிவிட்டுள்ளார்.
```

```
print(f"English Sentence: {english_sentence}")
print(f"Translated Sentence: {translation}")
```

```
English Sentence: My name is Vishnu Chunduru
Translated Sentence: “விஷ்ணு சஞ்சுரு என்னுடைய பெயர்.
```

```
Input:
Raw input text in English: We can use that.
Expected text in Tamil: இதை நாம் பயன்படுத்தி நன்மை பெறலாம்.

Output:
அதை நாம் பயன்படுத்திக் கொள்ளலாம்.
```



TELUGU

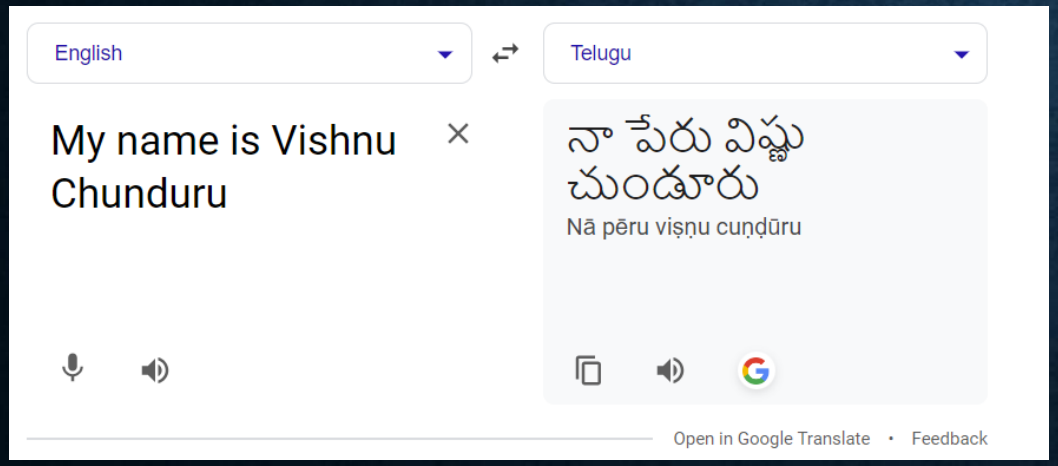
```
# Evaluation
test_bleu_score = evaluate_bleu(model, tokenizer, test_df.sample(1), model.config.max_length)
print(f"BLEU score on test data: {test_bleu_score.score}")
```

```
BLEU score on test data: 31.356006812336346
```

```
print(f"English Sentence: {english_sentence}")
print(f"Translated Sentence: {translation}")
```

```
English Sentence: My name is Vishnu Chunduru
Translated Sentence: నా పేరు విష్ణు చుండూరు.
```

```
Input:
Raw input text in English: No stopping anywhere.
Expected text in Telugu: బాలయ్య ఎక్కడా ఆగడం లేదు.

Output:
ఎక్కడా ఆగిపోలేదు.
```



CONCLUSION

- This project explored the application of the MT5 model in translating English into Dravidian languages, with a focus on Tamil and Telugu.
- Our results demonstrated a notable difference in translation quality between the two languages.
- The BLEU score for Tamil translations was 0.36, indicating a relatively higher translation accuracy compared to Telugu, which had a BLEU score of 0.31.
- This disparity highlights the challenges and varying degrees of difficulty in machine translation across different languages within the Dravidian family.
- The findings underscore the importance of continuous improvements and adaptations in machine translation models to enhance their effectiveness across diverse linguistic landscapes.

REFERENCE

- PICT@DravidianLangTech-ACL2022: Neural Machine Translation On Dravidian Languages - Aditya Vyawahare, Rahul Tangsali, Aditya Mandke, Onkar Litake, Dipali Kadam
- Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). "Attention is All You Need." In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
- This paper introduces the Transformer model, which is the basis for many NMT systems, including MT5.
- Raffel, C., Shazeer, N., Roberts, A., et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." Journal of Machine Learning Research, 21(140), 1-67.
- Bharathi Raja Chakravarthi, Ruba Priyadharshini, Navya Jose, et al. (2020). "A Sentiment Analysis Dataset for Code-Mixed Malayalam-English." Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020), Marseille, France.
- Although focused on sentiment analysis, this paper provides insights into handling code-mixed text in Dravidian languages, relevant to NMT for these languages.
- Papineni, K., Roukos, S., Ward, T., et al. (2002). "BLEU: a Method for Automatic Evaluation of Machine Translation." Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, USA.
- Introduces the BLEU score, a standard metric for evaluating machine translation quality.
- J. Albert, V. Pandian (2019). "Neural Machine Translation of Dravidian Languages by Incorporating Linguistic Resources." Neural Computing and Applications.

THANK YOU

VISHNU CHOUNDUR

NARESH BOLISHETTY

VISHNU TEJA YERRAPRAGADA