Neural Machine Translation

Nagamese Translation

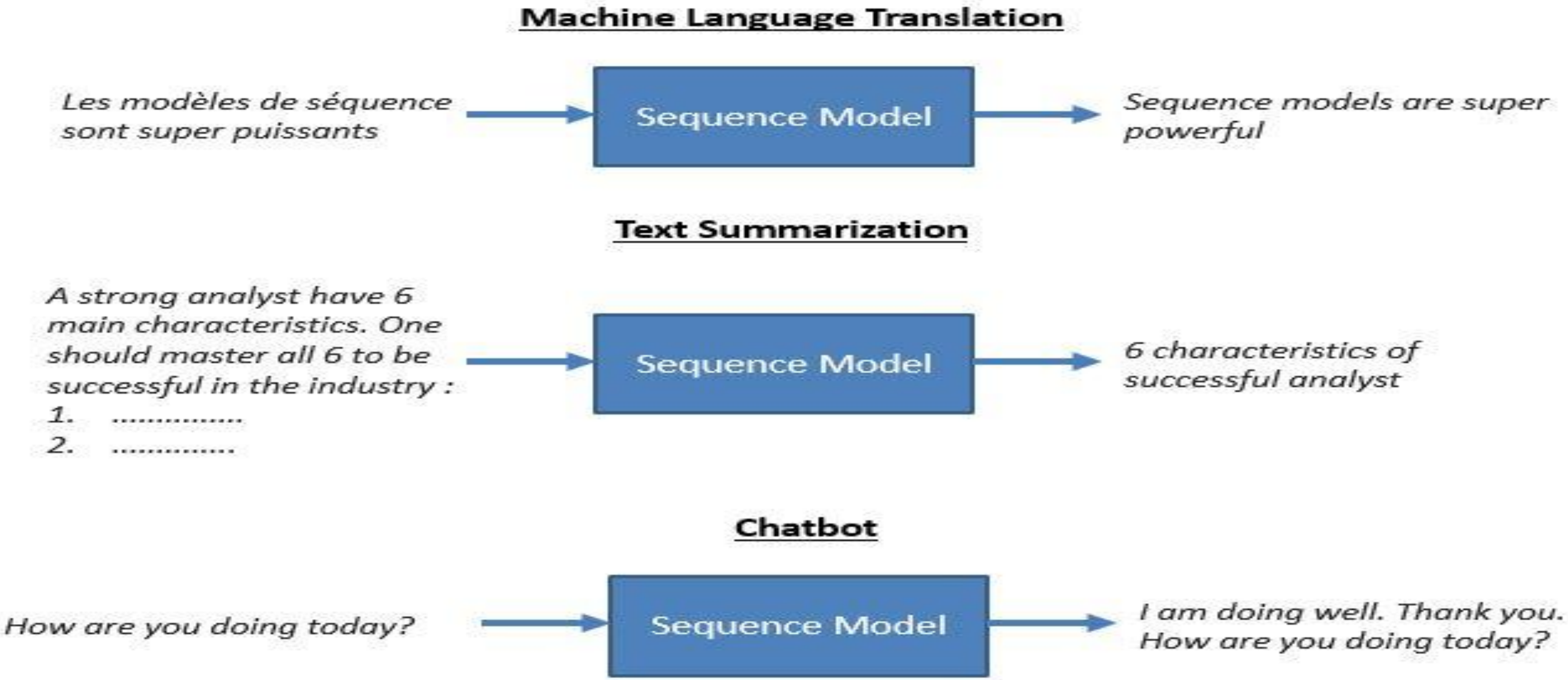
What is Neural Machine Translation or NMT ?

Neural Machine Translation(NMT) is a cutting-edge approach to automated language translation that utilizes artificial neural networks to enhance the accuracy and fluency of translation.NMT processes entire sentences rather than breaking then down into smaller components,allowing for a more holistic understanding of context and meaning.

Architecture for Nagamese Translation

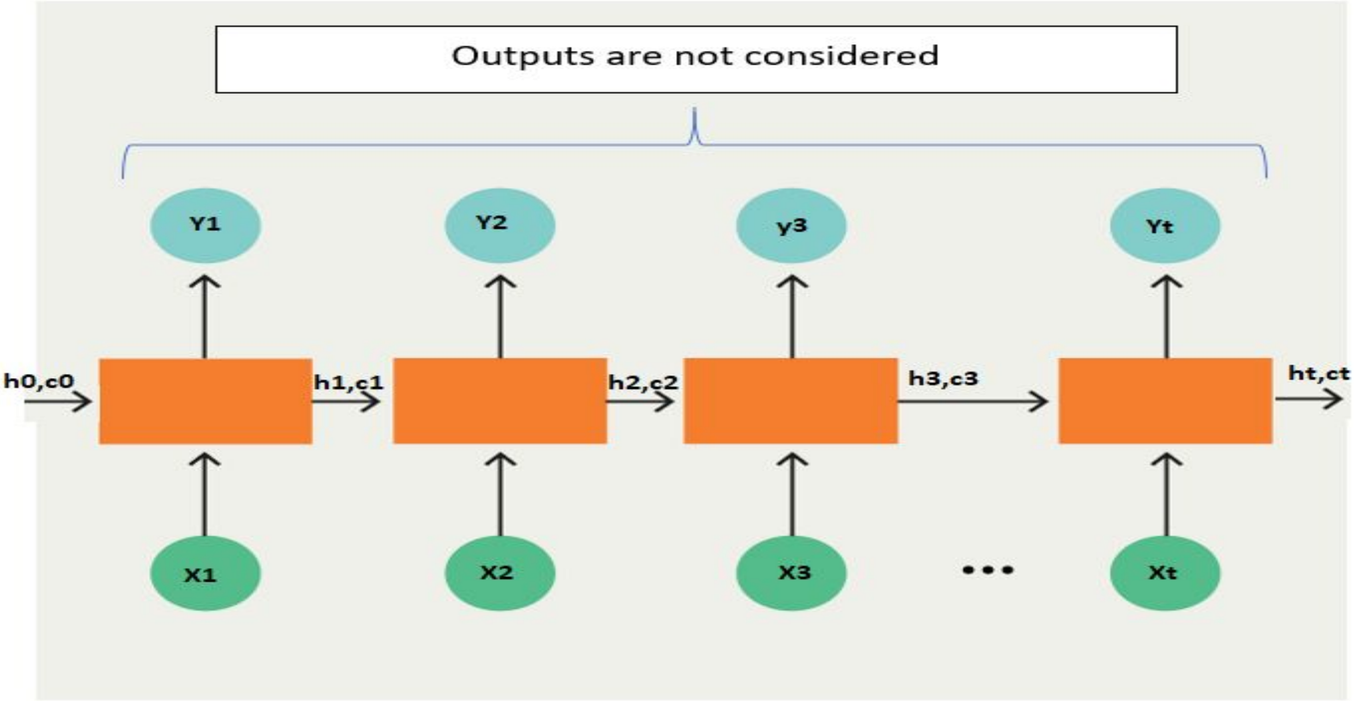
Seq2seq is a foundational architecture in NMT that consists of two main components:

- **Encoder:** Processes the input sequence (source language) and converts it into a fixed-length context vector.
- **Decoder:** Takes the context vector and generates the output sequence (target language). This model is particularly effective for handling variable-length input and output sequences, making it versatile for different languages and sentence structures.

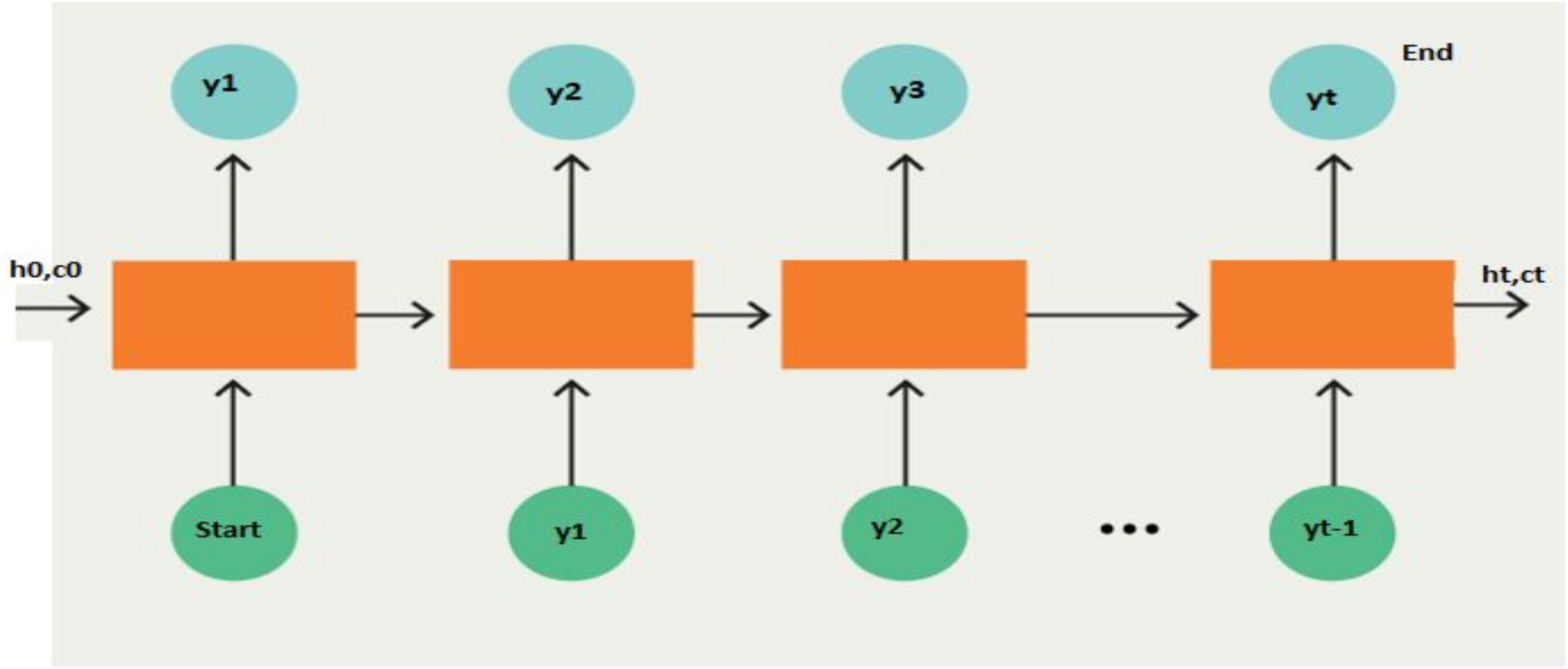

Machine Language Translation
Les modèles de séquence sont super puissants
Sequence Model
Sequence models are super powerful
Text Summarization
A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :
1. ....
2. ....
Sequence Model
6 characteristics of successful analyst
Chatbot
How are you doing today?
Sequence Model
I am doing well. Thank you. How are you doing today?

source

Encoder Architecture



Decoder Architecture



Challenges/Problems We Faced During Our Development

- One of the greatest challenges in Neural Machine Translation (NMT) development is obtaining sufficient amount of high-quality datasets to train the model to achieve optimal accuracy. Nagamese, a language spoken primarily in the northeastern Indian state of Nagaland, is a prime example of this challenge.

  As a regional language with limited native speakers, resources for Nagamese are scarce and difficult to find online. Even in print, literature and linguistic materials are limited, making it exceptionally challenging to gather the data needed for comprehensive language analysis and NMT model training.

Challenges/Problems We Faced During Our Development

- Training our NMT models on Google Colab poses challenges due to its limited computational resources and session time constraints. Colab offers single GPU instances with capped memory, which often proves insufficient for large models, leading to memory overflows and session timeouts.

  NOTE:It is recommended to use- Google Colab PRO.

Project-WorkFlow

- ***Data Gathering/Data Collection***
- ***Data-PreProcessing***
- ***Research & Model Selection***
- ***Training The Model***
- ***Model Evaluation and Testing***
- ***Fine-Tuning The Model***

Data Gathering / Data Collection

Collecting a large, high-quality dataset is foundational to NMT success. For low-resource languages like Nagamese, this step can be particularly challenging due to the scarcity of available linguistic resources. Sources may include parallel texts, bilingual dictionaries, and online resources, or may even require custom data gathering through interviews and transcriptions.

Scraping the Bible verses both for Nagamese and English turns out to be very effective for high-quality dataset.

NOTE: Scraping data from local news channel ,youtube comments and creating a custom datasets are the another ways to collect datasets.

Data Gathering / Data Collection

```python
# Define the array with file names
files = ['41-MAT.html', '42-MRK.html', '43-LUK.html', '44-JHN.html', '45-ACT.html',
         '46-ROM.html', '47-1CO.html', '48-2CO.html', '49-GAL.html', '50-EPH.html',
         '51-PHP.html', '52-COL.html', '53-1TH.html', '54-2TH.html', '55-1TI.html',
         '56-2TI.html', '57-TIT.html', '58-PHM.html', '59-HEB.html', '60-JAS.html',
         '61-1PE.html', '62-2PE.html', '63-1JN.html', '64-2JN.html', '65-3JN.html',
         '66-JUD.html', '67-REV.html']

# Initialize lists to collect English and Nagamese verses
all_eng_verses = []
all_naga_verses = []

# Iterate through each file in the array
for div in files:
    eng = extract_text(bs_eng, div)  # Fetch English text for the given div
    naga = extract_text(bs_naga, div)  # Fetch Nagamese text for the given div

    eng_verses = split_corpus(eng)  # Split English text into verses
    naga_verses = split_corpus(naga)  # Split Nagamese text into verses

    # Ensure both languages have the same number of verses
    if len(eng_verses) == len(naga_verses):
        all_eng_verses.extend(eng_verses)
        all_naga_verses.extend(naga_verses)
    else:
        print(f"Verse count mismatch in {div}: English({len(eng_verses)}) != Nagamese({len(naga_verses)})")

# Save all collected verses to a single CSV file
save_to_csv("/content/bible_verses.csv", ["English", "Nagamese"], all_eng_verses, all_naga_verses)
```

```
Verse count mismatch in 41-MAT.html: English(1072) != Nagamese(1075)
Verse count mismatch in 42-MRK.html: English(667) != Nagamese(679)
Verse count mismatch in 45-ACT.html: English(1011) != Nagamese(1009)
Verse count mismatch in 48-2CO.html: English(258) != Nagamese(257)
Verse count mismatch in 67-REV.html: English(412) != Nagamese(414)
Data saved successfully to /content/bible_verses.csv
```

Data Gathering / Data Collection

Code + Text All changes saved

RAM Disk Gemini

bible_verses.csv

1 to 50 of 4564 entries Filter

English	Nagamese
1	1
Many have taken on the work of putting together an account of the things that have been accomplished among us, 2	Bisi manukhan pora kotha aru kaam khan laga khisa joma kori kena likhibo nimite kosis korise, juntu amikhan majot te pura hoise 2
just as they were passed down to us by those who from the first were eyewitnesses and servants of the word. 3	jineka pora amikhan laga hath te dise jun khan shuru te nijor suku pora dikhise aru kotha laga sewak khan asele. 3
So it seemed good to me also, because I have accurately investigated everything from the beginning, to write an orderly account for you, most excellent Theophilus, 4	Itu nimite ami itu bisi bhal nisena dikhi ase, bhal pora bisari kena sob kotha aru kaam khan to shuru pora apni ke hisab-kitab pora likhi ase mohan Theophilus, 4
so that you might know the certainty of the things you have been taught. 5	eneka hoile tumi janibo itu bostu laga asol kotha juntu apni ke sikhai dise. 5
In the days of Herod king of Judea there was a certain priest named Zechariah from the division of Abijah; his wife Elizabeth was also a descendant of Aaron. 6	Aru Herod pora Judea laga raja thaka somoite Abijah laga dol pora Zachariah naam laga ekjon purohit thakise. Tai laga maiki to Aaron laga khandan pora asele aru tai laga naam Elizabeth asele. 6
They were both righteous before God, walking blameless in all the commandments and statutes of the Lord. 7	Aru tai duijon Isor usorte bisi bhal thakise, aru Probhu pora ki koi sob kotha to sapha pora mani thake. 7
But they had no child because Elizabeth was barren, and they were both advanced in their days. 8	Aru tai khan bacha thaka nai, kilemane Elizabeth to bacha pabole napara maiki thakise, aru tai duijon laga umor bi bisi hoi jaise. 8
Now it came about that Zechariah was in God's presence, carrying out the priestly duties in the order of his division. 9	Etiya eneka hoise, jitia Zechariah tai Isor usorte tai laga purohit kaam khan kori asele, juntu kaan khan tailaga dol khan pora kore. 9
According to the customary way of choosing which priest would serve, he had been chosen by lot to enter into the temple of the Lord to burn incense. 10	Titia, purohit laga niom hisab te, tai ke Isor laga pobitro ghor bhitorte jaikena dhup aru dhuna julabole chithi pora basi se. 10
The whole crowd of people was praying outside at the hour when the incense was burned. 11	Jitia dhuna juli jaise itu homoite sob manukhan bahar te prathana kori thakise. 11
Now an angel of the Lord appeared to him and stood at the right side of the incense altar. 12	Titia Probhu laga sorgoduth tai usorte ahise, aru jun jagate dhuna julai se itu laga dyna phale khara korise. 12
When Zechariah saw him, he was troubled, and fear fell on him. 13	Jitia Zechariah itu dikhise tai bisi chinta hoise aru tai uporte bisi bhoi ahi jaise. 13
But the angel said to him, "Do not be afraid, Zechariah, because your prayer has been heard. Your wife Elizabeth will bear you a son. You will call his name John. 14	Hoilebi, sorgoduth koise, "Bhoi nakoribi Zechariah, tumi laga prathana huni loise. Tumi laga maiki Elizabeth pora ekjon chokra pabo. Aru tumi tai laga naam John rakhibo. 14
You will have joy and gladness, and many will rejoice at his birth. 15	Aru tumi khan anond aru bhal lagibo, aru bisi manukhan tai laga jonom te khushi koribo. 15
For he will be great in the sight of the Lord. He must never drink wine or strong drink, and he will be filled with the Holy Spirit from his mother's womb. 16	Aru tai Probhu usorte bisi dangor hobo. Tai kitia bi draikha-ros nakhabole lage, tai ama laga peth te thaka homoi pora he Pobitro Atma pora bhorta hoijabo. 16
Many of the descendants of Israel will be turned to the Lord their God. 17	Aru tai Israel laga bacha khan ke Probhu Isor logote wapas anibo. 17
He will go before the face of the Lord in the spirit and power of Elijah, to turn the hearts of the fathers to the children and the disobedient to the wisdom of the righteous—to make ready for the Lord a people prepared for him." 18	Elijah laga atma aru takot pora Probhu laga kaam te tai he age korikena loi jabo. Aru jun dharmikta laga kotha namani kena ase taikhan laga mon ghuraidibo nimite aru baba khan laga mon taikhan laga bacha khan phale ghurai dibole, aru jun Probhu ahibole ase Tai nimite rasta taiyar koribole." 18
Zechariah said to the angel, "How can I know this? For I am an old man and my wife is advanced in her days." 19	Zechariah pora sorgoduth ke koise, "Moi itu kineka janibo? Kilemane moi ekta bura manu hoise aru moi laga maiki bi tai laga umor to bisi hoise." 19
The angel answered and said to him, "I am Gabriel, who stands in the presence of God. I was sent to speak to you, to bring you this good news. 20	Aru sorgoduth tai ke jawab di kena koise, "Moi Gabriel ase, jun Isor usorte khara kore. Ami ke tumi logote kotha kobole pathaise, aru itu bhal khobor to tumi ke kobole nimite. 20
Behold! You will be silent, unable to speak, until the day these things take place. This is because you did not believe my words, which will be fulfilled at the right time." 21	Aru itu nimite, sob kaam to nohua tak tumi chup thakibo aru kotha ulabo naparibo, kilemane tumi moi laga kotha biswas kora nai, itu sob to tai laga thik homoi te pura hobo." 21
Now the people were waiting for Zechariah. They were surprised that he was spending so much time in the temple. 22	Aru manukhan Zechariah nimite rukhi thakisele, aru taikhan chinta-bhabona korise jitia tai mondir ghor bhithor te deri korise. 22
But when he came out, he could not speak to them. They realized that he had seen a vision while he was in the temple. He kept on making signs to them and remained silent. 23	Kintu jitia tai baharte ulai ahise tai kotha kobole para nai, titia taikhan bhabise taike Isor pora kiba ekta dikhaise; aru tai manukhan ke ishara pora dikhai di thakise, aru kotha koribo naparikena he thaki jaise. 23
It came about that when the days of his service were over, he went to his house. 24	Aru itu sob to tai laga sewa kora homoi pura aru khotom hoise, titia tai ghor te jai jaise. 24
After these days, his wife Elizabeth conceived and for five months she kept herself hidden. She said, 25	Aru itu din piche, tai laga maiki Elizabeth bacha bukhi se, aru pans mohina tak tai nije ke lukai kena rakhise, aru koi thakise, 25
"This is what the Lord has done for me when he looked at me with favor in order to take away my shame before people." 26	"Isor ami uporte eneka korise, aru ami uporte Probhu he daya korise, manu usorte sorom nakhilabo nimite." 26

Data-PreProcessing

1. **Cleaning and normalization**
    - Removing duplicates, unwanted characters, HTML tags, and special symbols.
    - Normalizing punctuation marks and spacing.

2. **Removing Noise & Filtering**

    - Filtering out incomplete sentences or mismatched pairs that could reduce the quality of alignment in translation pairs.
    - Removing extremely long or short sentences that could skew the model training.

3. **Tokenization/Subwording**

    - Splitting text into smaller units, such as words, subwords, or characters, depending on the model requirements.
    - SentencePiece or Byte-Pair Encoding (BPE) can be used for subword tokenization, especially useful for handling rare or unseen words in the target language.

Data-PreProcessing

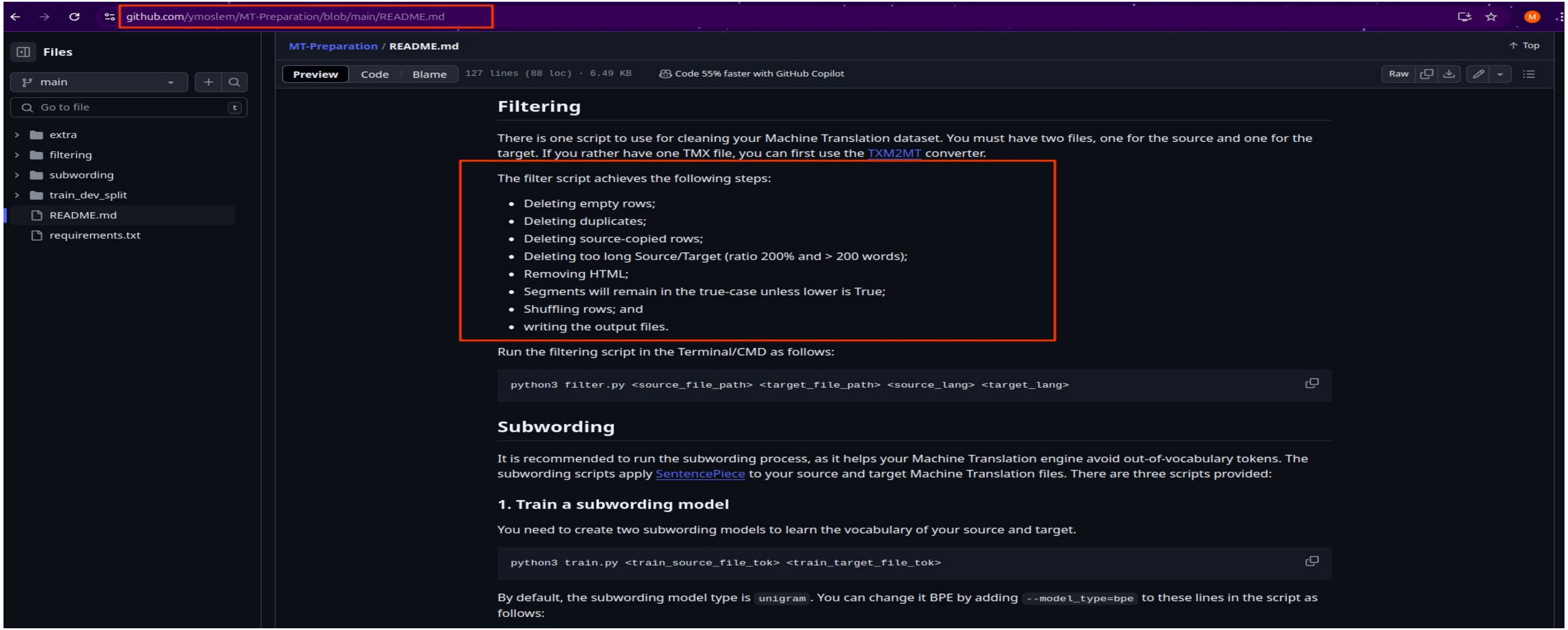
4. Data Splitting

- The Total number of datasets after preprocessing (cleaning, filtering, removing noise & tokenization) is been divided into train, test & dev datasets for Training ,Testing & Development of the Model.

5. Vocabulary

- A specific set of vocabulary is extracted from the training datasets since for a huge datasets it is not feasible to use all tokens/words/subwords.

Data-PreProcessing



Data-PreProcessing

```
[ ] unigram_model_trainer.cc(269) LOG(INFO) Extracting frequent sub strings... node_num=284838
    unigram_model_trainer.cc(312) LOG(INFO) Initialized 14555 seed sentencepieces
    trainer_interface.cc(598) LOG(INFO) Tokenizing input sentences with whitespace: 4420
    trainer_interface.cc(609) LOG(INFO) Done! 9420
    unigram_model_trainer.cc(602) LOG(INFO) Using 9420 sentences for EM training
    unigram_model_trainer.cc(618) LOG(INFO) EM sub_iter=0 size=5351 obj=10.2813 num_tokens=18886 num_tokens/piece=3.52943
    unigram_model_trainer.cc(618) LOG(INFO) EM sub_iter=1 size=4461 obj=8.21106 num_tokens=18967 num_tokens/piece=4.25174
    trainer_interface.cc(687) LOG(INFO) Saving model: target.model
    trainer_interface.cc(699) LOG(INFO) Saving vocabs: target.vocab
    Done, training a SentencepPiece model for the Target finished successfully!
```

```
!ls
```

```
bible_verses.csv  english.csv-filtered.en  nagamese.csv              source.model  target.model
english.csv       MT-Preparation           nagamese.csv-filtered.ng  source.vocab  target.vocab
```

```
[ ] # Subword the dataset
    !python3 MT-Preparation/subwording/2-subword.py source.model target.model "/content/machine_translation/nagamese.csv-filtered.ng" "/content/machine_translation/english.csv-filtered.en"
```

```
Source Model: source.model
Target Model: target.model
Source Dataset: /content/machine_translation/nagamese.csv-filtered.ng
Target Dataset: /content/machine_translation/english.csv-filtered.en
Done subwording the source file! Output: /content/machine_translation/nagamese.csv-filtered.ng.subword
Done subwording the target file! Output: /content/machine_translation/english.csv-filtered.en.subword
```

```
[ ] # Split the dataset into training set, development set, and test set
    # Development and test sets should be between 1000 and 5000 segments (here we chose 2000)
    !python MT-Preparation/train_dev_split/train_dev_test_split.py 500 500 "/content/machine_translation/nagamese.csv-filtered.ng.subword" "/content/machine_translation/english.csv-filtered.en.subword"
```

```
Dataframe shape: (4420, 2)
--- Empty Cells Deleted --> Rows: 4420
--- Wrote Files
Done!
Output files
/content/machine_translation/nagamese.csv-filtered.ng.subword.train
/content/machine_translation/english.csv-filtered.en.subword.train
/content/machine_translation/nagamese.csv-filtered.ng.subword.dev
/content/machine_translation/english.csv-filtered.en.subword.dev
/content/machine_translation/nagamese.csv-filtered.ng.subword.test
/content/machine_translation/english.csv-filtered.en.subword.test
```

Research & Model Selection

One of the most used Model for Neural Machine Translation which also enable us to work with seq2seq model & with highly configurable model architectures and training procedures is **OpenNMT**.

OpenNMT is an open source ecosystem for neural machine translation and neural sequence learning.Started in December 2016 by the Harvard NLP group and SYSTRAN, the project has since been used in several research and industry applications. It is currently maintained by SYSTRAN and Ubiqus.

Training The Model

YAML Configuration File for OpenNMT(ONMT)

The YAML configuration file for OpenNMT (ONMT) contains essential parameters that define the dataset paths, model parameters, and training procedures. This file is crucial for orchestrating training, validation, and checkpointing workflows in NMT projects. For environments with large-scale data, certain parameters must be scaled accordingly to enhance performance and accuracy.

Key Configuration Parameters

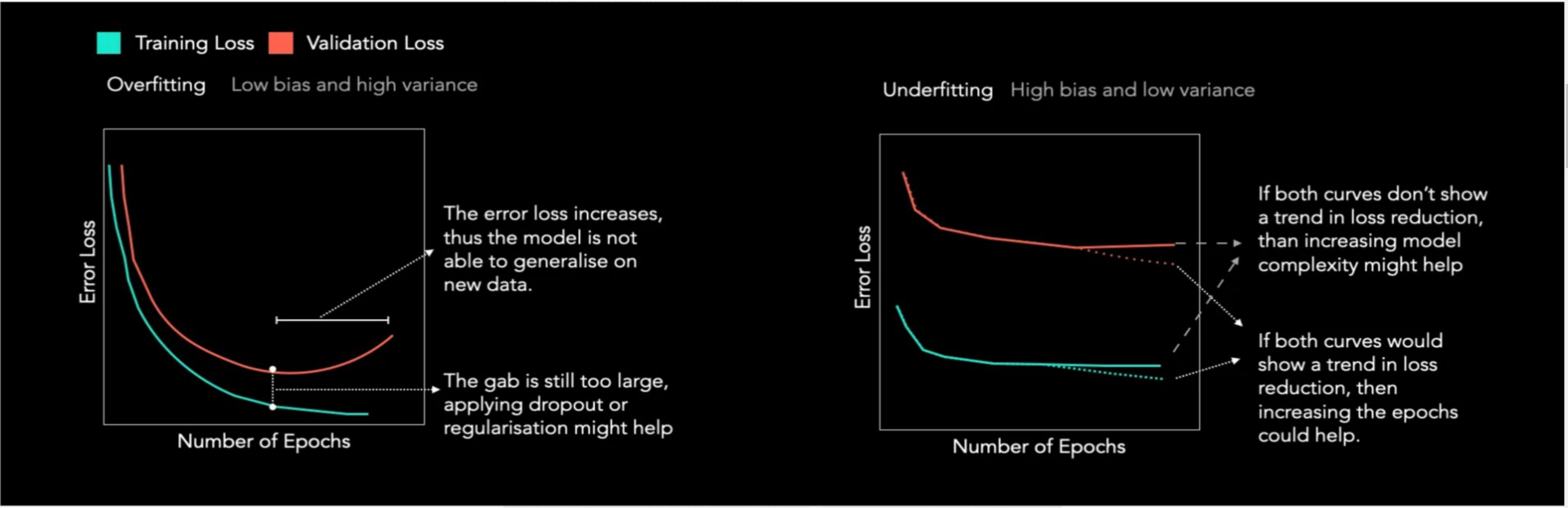
1. **Define Configuration**: Create a YAML configuration file to specify model parameters, such as `src_vocab_size`, `tgt_vocab_size`, `batch_size`, `learning_rate`, `train_steps`, and `valid_steps`. These parameters need adjustment depending on dataset size and computational resources.
2. **Hyperparameter Tuning**:
   a. **Batch Size**: Choose a batch size that balances between memory usage and gradient stability. A larger batch size may require gradient accumulation for stability.
   b. **Learning Rate**: Start with a low learning rate (e.g., 0.0001–0.001) for fine-tuning; this helps avoid drastic changes in pre-trained weights.
   c. **Training Steps**: For small datasets, limit training steps (e.g., 3,000–5,000 steps) to avoid overfitting, increasing steps for larger datasets.
   d. **Checkpoints**: Set `save_checkpoint_steps` and `keep_checkpoint` to periodically save model checkpoints, aiding in recovery and stability during longer training sessions.

Training Loss & Validation Loss

Training Loss and **Validation Loss** are metrics used to evaluate a machine learning model's performance during training:

1. **Training Loss**: This is the error calculated on the training data after each iteration of model optimization. It shows how well the model fits the training dataset. A declining training loss over time indicates that the model is learning and adapting to the data.
2. **Validation Loss**: This error is measured on a separate validation dataset not used in training. It evaluates how well the model generalizes to new, unseen data. Ideally, validation loss should decrease alongside training loss. If validation loss increases while training loss decreases, it suggests overfitting, where the model learns the training data too closely but fails to generalize.

Training Loss & Validation Loss



Training The Model

+ Code + Text All changes saved

```
                 * corpus_1: 271
[2024-11-06 19:49:49,124 INFO] Weighted corpora loaded so far:
                 * corpus_1: 272
[2024-11-06 19:49:49,386 INFO] Weighted corpora loaded so far:
                 * corpus_1: 273
[2024-11-06 19:49:49,636 INFO] Weighted corpora loaded so far:
                 * corpus_1: 274
[2024-11-06 19:49:56,152 INFO] Weighted corpora loaded so far:
                 * corpus_1: 275
[2024-11-06 19:49:56,300 INFO] Weighted corpora loaded so far:
                 * corpus_1: 276
[2024-11-06 19:49:58,437 INFO] Step 2400/ 4000; acc: 99.5; ppl:   3.4; xent: 1.2; lr: 0.00090; sents:   17223; bsz: 1556/1453/43; 19629/18333 tok/s;    781 sec;
[2024-11-06 19:50:02,525 INFO] Weighted corpora loaded so far:
                 * corpus_1: 277
[2024-11-06 19:50:02,794 INFO] Weighted corpora loaded so far:
                 * corpus_1: 278
[2024-11-06 19:50:09,338 INFO] Weighted corpora loaded so far:
                 * corpus_1: 279
[2024-11-06 19:50:09,491 INFO] Weighted corpora loaded so far:
                 * corpus_1: 280
[2024-11-06 19:50:09,645 INFO] Weighted corpora loaded so far:
                 * corpus_1: 281
[2024-11-06 19:50:15,867 INFO] Weighted corpora loaded so far:
                 * corpus_1: 282
[2024-11-06 19:50:16,125 INFO] Weighted corpora loaded so far:
                 * corpus_1: 283
[2024-11-06 19:50:22,496 INFO] Weighted corpora loaded so far:
                 * corpus_1: 284
[2024-11-06 19:50:22,846 INFO] Weighted corpora loaded so far:
                 * corpus_1: 285
[2024-11-06 19:50:22,992 INFO] Weighted corpora loaded so far:
                 * corpus_1: 286
[2024-11-06 19:50:28,940 INFO] Weighted corpora loaded so far:
                 * corpus_1: 287
[2024-11-06 19:50:29,461 INFO] Weighted corpora loaded so far:
                 * corpus_1: 288
[2024-11-06 19:50:30,400 INFO] Step 2500/ 4000; acc: 99.5; ppl:   3.4; xent: 1.2; lr: 0.00088; sents:   17446; bsz: 1557/1457/44; 19479/18236 tok/s;    813 sec;
[2024-11-06 19:50:31,042 INFO] valid stats calculation
                  took: 0.6403071880340576 s.
[2024-11-06 19:50:31,044 INFO] Train perplexity: 5.84653
[2024-11-06 19:50:31,044 INFO] Train accuracy: 88.6603
[2024-11-06 19:50:31,044 INFO] Sentences processed: 433228
[2024-11-06 19:50:31,044 INFO] Average bsz: 1557/1455/43
[2024-11-06 19:50:31,044 INFO] Validation perplexity: 166.196
[2024-11-06 19:50:31,044 INFO] Validation accuracy: 38.0781
[2024-11-06 19:50:31,045 INFO] Stalled patience: 0/4
[2024-11-06 19:50:31,045 INFO] Training finished after stalled validations. Early Stop!
[2024-11-06 19:50:31,045 INFO] Best model found at step 500
[2024-11-06 19:50:31,045 INFO] earlystopper has_stopped!
[2024-11-06 19:50:31,120 INFO] Saving checkpoint models/model.fren_step_2500.pt
```

Model Evaluation and Testing

1. **BLEU Score Evaluation**: Assess the model's performance using the BLEU score on the validation and test sets. A higher BLEU score generally indicates better translation accuracy.
2. **Error Analysis**: Examine translations for common errors, such as mistranslations or repetitive phrases. Error analysis can provide insights for further fine-tuning or augmenting the training dataset.
3. **Iteration**: Based on evaluation results, adjust hyperparameters, and re-train if necessary. Iterative fine-tuning may yield incremental improvements.

Model Evaluation and Testing

The BLEU Score of the model is ***low*** because of lack of resources.But, can be enhanced with large amount high-quality datasets & fine-tuning, in future.

```
Successfully installed colorama-0.4.6 portalocker-2.10.1 sacrebleu-2.4.3

# Evaluate the translation (without subwording)
!python3 compute-bleu.py english.csv-filtered.en.subword.test.desubword en.translated.desubword

Reference 1st sentence: They were not able to give an answer to these things. 7
MTed 1st sentence: The thief does not come except to steal. 7
BLEU:  8.027037299990369
```

Fine-Tuning The Model

Fine-tuning a model refers to the process of taking a pre-trained machine learning model and adapting it to a specific task or dataset. Fine-tuning optimizes the model for specialized use cases, allowing it to leverage prior knowledge while adjusting to new data, leading to better performance without needing to train from scratch.

STEPS:

Initialize Training: Load pre-trained weights and start the training process using the fine-tuning dataset. Track key metrics, especially BLEU scores, which assess the translation quality.

Checkpoints and Early Stopping: Implement checkpoints and early stopping to avoid unnecessary training when validation accuracy plateaus. This saves computational resources and avoids overfitting.

Logging: Enable detailed logging for loss, accuracy, and BLEU scores at regular intervals (`report_every` parameter). This feedback loop allows monitoring of model progress and timely interventions if issues arise.

NOTE:we used Hugging Face & Wandb for fine-tuning our model which requires to create an account on those platforms for Api-key and validation-token to use its services.

+ Code + Text Connect T4 Gemini

FINE-Tuning The Model

```
[ ] from transformers import Seq2SeqTrainingArguments

    args = Seq2SeqTrainingArguments(
        f"mbart-50-finetuned-eng-to-naga",  # mBART-50 is a multilingual Sequence-to-Sequence model. It was introduced to show that multilingual translation models can be created through multilingual fine-tuning.
        evaluation_strategy="no",
        save_strategy="epoch",
        learning_rate=2e-5,
        per_device_train_batch_size=1,
        per_device_eval_batch_size=1,
        weight_decay=0.01,
        save_total_limit=3,
        num_train_epochs=3,
        predict_with_generate=True,
        fp16=True,
        push_to_hub=True,
    )
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1525: FutureWarning: `evaluation_strategy` is deprecated and will be removed in version 4.46 of 🤗 Transformers. Use `eval_strategy` instead
  warnings.warn(
```

```
[ ] from transformers import Seq2SeqTrainer

    trainer = Seq2SeqTrainer(
        model,
        args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=tokenized_datasets["validation"],
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
    )
```

```
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:494: FutureWarning: `torch.cuda.amp.GradScaler(args...)` is deprecated. Please use `torch.amp.GradScaler('cuda', args...)` instead.
  self.scaler = torch.cuda.amp.GradScaler(**kwargs)
```

```
# A wandb.ai profile is alos needed to create for API Key....

trainer.evaluate(max_length=max_length)
```

```
[ ]
translator("The boat was in the sea")
```

```
[{'translation_text': 'Aru naw to samundar te thakise'}]
```

```
[ ]
translator(
    "When he was walking beside the Sea of Galilee, he saw Simon and Andrew the brother of Simon casting a net in the sea, for they were fishermen."
)
```

```
[{'translation_text': 'Jitia Tai Galilee laga samundar kinar te berai thakise, Tai Simon aru tai laga bhai Andrew ke samundar te jal phelai thaka dikhise, kilemane tai khan maas dhora manu asele.'}]
```

```
[ ] translator(" eating fish")
```

```
[{'translation_text': 'maas kha luwa kora'}]
```

Fine-Tuning The Model

```
[ ] # A wadb profile is alos needed to create for API Key....

    trainer.evaluate(max_length=max_length)
```

```
[1590/1590 24:44]
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please specify a different run name by setting the `TrainingArguments.run_name` parame
wandb: Using wandb-core as the SDK backend. Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: ERROR API key must be 40 characters long, yours was 39
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.5
Run data is saved locally in /content/wandb/run-20241107_090357-ehv333w0
Syncing run mbart-50-finetuned-eng-to-naga to Weights & Biases (docs)
View project at https://wandb.ai/sabirasdev-royal-school-of-engineering-technology/huggingface
View run at https://wandb.ai/sabirasdev-royal-school-of-engineering-technology/huggingface/runs/ehv333w0
{'eval_loss': 10.652088165283203,
 'eval_model_preparation_time': 0.0078,
 'eval_bleu': 0.20485897823256105,
 'eval_runtime': 1487.8241,
 'eval_samples_per_second': 1.069,
 'eval_steps_per_second': 1.069}
```

```
[ ] trainer.train()
```

[10644/19080 1:12:39 < 57:35, 2.44 it/s, Epoch 1.67/3]

Step	Training Loss
500	4.327400
1000	2.828800
1500	5.228100

Fine-Tuning The Model

```
In [ ]: trainer.evaluate(max_length=max_length)
```

[1590/1590 45:05]

```
Out[ ]: {'eval_loss': 1.4960336685180664,
 'eval_bleu': 20.9247827422808,
 'eval_runtime': 2709.558,
 'eval_samples_per_second': 0.587,
 'eval_steps_per_second': 0.587,
 'epoch': 3.0}
```

Tools & Repositories

- Hugging Face
- Google Colab
- Jupyter NoteBook
- OpenNMT
- MT-Preparation
- Wandb ai