First 2 week Internship Report: Cryptography, Language Translation Model Comparison

Harsh Tripathi

– May 25, 2025

---

1. Executive Summary

This report details the progress made during the fortnight from **May 14, 2025** to **May 25, 2025**. The initial phase of the internship involved exploring **cryptography concepts**. Following this, the focus shifted to a machine learning project for **offline language translation**. This report primarily covers the latter, outlining the comparison of various open-source, offline translation models for Hindi-English, Chinese-English, and Burmese-English language pairs. We used benchmark datasets and compared their accuracy against Google Translate. Initial findings indicate varying levels of accuracy across models and language pairs, with **Google Translate consistently setting a high benchmark**. Notably, LibreTranslate encountered significant errors across all tested language pairs, leading to a BLEU score of 0.00 for those runs.

---

2. Project Background and Objective

The internship, initially focused on **cryptography**, transitioned to a machine learning project centered on developing an offline language translation solution. The goal is to create software capable of translating languages (e.g., Hindi to English, Burmese to English, Urdu to English) without relying on online APIs or extensive training on custom datasets. The current phase, as per the supervisor's instruction, is dedicated to:

- Exploring and identifying suitable open-source, pre-trained language translation models.
- Conducting a rigorous accuracy comparison of these models.
- Utilizing benchmark datasets for evaluation.
- Providing separate comparisons for Hindi-English, Chinese-English, and Burmese-English translations.
- Benchmarking the performance against Google Translate (as an online reference).
- Documenting all explored tools, results, and observations in a comprehensive report format.

---

3. Initial Focus: Cryptography

The first week of the internship was dedicated to understanding fundamental **cryptography topics**. This foundational knowledge is crucial for understanding secure data transmission and privacy, which can be relevant for any future secure offline translation deployments. The key concepts covered included:

- **Symmetric-key Cryptography:**

  - **AES (Advanced Encryption Standard):** Studied the block cipher algorithm widely used for securing sensitive data. Learned about its different key sizes (128, 192, 256 bits) and modes of operation (e.g., CBC, GCM).
  - **DES (Data Encryption Standard) / 3DES (Triple DES):** Understood the older standard and its more secure successor, providing context for the evolution of symmetric-key algorithms.
- **Asymmetric-key Cryptography (Public-key Cryptography):**
  - **RSA (Rivest-Shamir-Adleman):** Explored its principles for secure communication over insecure channels, focusing on public and private key pairs.
  - **Elliptic Curve Cryptography (ECC):** Understood its advantages in terms of stronger security with smaller key sizes compared to RSA.
- **Hashing Algorithms:**
  - **SHA-256 (Secure Hash Algorithm 256-bit):** Learned about its use in creating fixed-size unique digital fingerprints of data, essential for data integrity and password storage.
  - **MD5 (Message-Digest Algorithm 5):** Briefly touched upon this older hashing algorithm and its known vulnerabilities, highlighting the importance of using more secure alternatives.
- **Digital Signatures:** Understood how digital signatures provide authenticity, integrity, and non-repudiation in electronic communications, often built using asymmetric cryptography and hashing.
- **Key Exchange:** Explored methods like **Diffie-Hellman** for securely establishing shared secret keys over an insecure channel.

This initial week provided a solid understanding of cryptographic principles, which will be valuable for designing secure applications in the future.

---

4. Methodology for Model Comparison (Language Translation)

To ensure a systematic and comparable evaluation for the language translation models, the following methodology was adopted:

- **Model Selection:** Based on initial exploration, the following models were selected for evaluation: **M2M100 1.2B**, **M2M100 418M** (from Hugging Face), **Argos Translate**, and **LibreTranslate** (used via a local server URL provided). **Google Translate** served as an online performance baseline.
- **Dataset Selection:** For each language pair, publicly available benchmark datasets were used:
  - **Burmese-English:** A subset of the `en-my.tmx.gz` TMX file (limited to 50 sentence pairs).
  - **Chinese-English:** A subset of a JSONL dataset containing `english` and `chinese` fields (limited to 50 sentence pairs).
  - **Hindi-English:** Subsets of `IITB.en-hi.en` and `IITB.en-hi.hi` files (limited to 50 sentence pairs).
- **Evaluation Metric:** The **BLEU (Bilingual Evaluation Understudy) score** was chosen as the primary metric for evaluating translation accuracy. BLEU scores quantify the similarity between the machine-translated text and a set of high-quality reference translations.
- **Google Translate Baseline:** For each language pair, translations were obtained using Google Translate (via `googletrans` library), and their BLEU scores served as a comparative baseline.
- **Implementation:** All models were implemented and run using Python within a Google Colab environment. **Hugging Face Transformers** was used for M2M100 models, `argostranslate` for Argos, `libretranslatepy` for LibreTranslate, and `googletrans` for Google Translate. Custom Python scripts were developed to automate the translation process and calculate BLEU scores. All M2M100 model inferences were performed on the available GPU (CUDA) if present, otherwise on CPU.

—

5. Models Explored and Their Characteristics (Language Translation)

This section provides details on the specific models that were explored and subsequently chosen for comparison.

5.1. M2M100 1.2B

- **Description:** This is a large-scale multilingual machine translation model from Facebook, part of the Many-to-Many (M2M) family. The "1.2B" indicates it has approximately 1.2 billion parameters, making it a powerful model capable of translating directly between 100 languages without needing an intermediate English pivot.
- **Offline Capability:** Fully capable of running offline once the model weights are downloaded.
- **Source:** Hugging Face Model Hub (`facebook/m2m100_1.2B`).
- **Observations on setup/usage:** Required significant memory and computational resources (GPU highly recommended) due to its size. Setup was straightforward using the `transformers` library.

5.2. M2M100 418M

- **Description:** Another model from the Facebook M2M100 family, this is a smaller version with 418 million parameters. It also supports direct translation between 100 languages.
- **Offline Capability:** Fully capable of running offline once the model weights are downloaded.
- **Source:** Hugging Face Model Hub (`facebook/m2m100_418M`).
- **Observations on setup/usage:** Easier to run on systems with less GPU memory compared to the 1.2B version, but still benefits greatly from GPU acceleration. Setup was similar to the 1.2B model.

5.3. Argos Translate

- **Description:** An open-source offline translation engine that bundles neural machine translation models with user-friendly interfaces. It allows for local installation of language packs.
- **Offline Capability:** Designed for offline use. Language packages are downloaded and installed locally.
- **Source:** `argostranslate` Python package and its package repository.
- **Observations on setup/usage:** Installation of language packages is required before translation. Generally simpler to set up for basic offline use than the Hugging Face models for specific pairs, but model quality can vary.

5.4. LibreTranslate

- **Description:** An open-source machine translation API that can be self-hosted. While it often runs as an online service, it can be deployed locally for offline access.
- **Offline Capability:** Capable of offline use if a local server is running. For these tests, an online instance (`https://b458-49-43-7-49.ngrok-free.app`) was used as provided in the code, which means it behaved as an online API for the purpose of this comparison.
- **Source:** `libretranslatepy` Python package interacting with a LibreTranslate server.
- **Observations on setup/usage:** Encountered frequent `HTTP Error 400: Bad Request` errors, resulting in many skipped translations and a 0.00 BLEU score. This suggests issues with the accessibility or stability of the provided LibreTranslate server, or potential rate limiting, rather than the model's inherent capability.

5.5. Google Translate

- **Description:** Google's widely-used proprietary online machine translation service. It leverages vast amounts of data and advanced neural network architectures.
- **Offline Capability:** Not inherently an offline model; it relies on an internet connection. Included as a high-quality online benchmark for comparison.
- **Source:** `googletrans` Python library, which scrapes the Google Translate website.
- **Observations on setup/usage:** Generally robust and provides high-quality translations, but as an online service, it's subject to network latency and potential rate limits (though none were observed in these limited tests).

---

6. Accuracy Comparison Results

This section presents the core findings of the model comparison for each language pair.

6.1. Burmese-English Translation

- **Benchmark Dataset Used:** A subset of `en-my.tmx.gz` (50 sentence pairs).
- **BLEU Scores (MY → EN):**

Model	BLEU Score
Google Translate (Baseline)	9.14
M2M100 1.2B	1.42
M2M100 418M	0.72
LibreTranslate	0.00

- **Observations:**
  - Google Translate significantly outperforms the offline models, indicating the challenge of Burmese-English translation for smaller, publicly available models.
  - M2M100 1.2B, while having a low absolute BLEU score, performed better than its 418M counterpart, which is expected due to its larger parameter count.
  - LibreTranslate yielded a 0.00 BLEU score due to persistent `HTTP Error 400: Bad Request` messages for almost all sentences, preventing successful translation. This suggests an issue with the server used rather than the model's capability itself.
  - The overall low scores for offline models highlight the difficulty of this language pair, possibly due to data scarcity or complexity in model training for less common languages.
  - Argos does not have any Burmese translation Dataset directly.

6.2. Chinese-English Translation

- **Benchmark Dataset Used:** A JSONL dataset (50 sentence pairs).
- **BLEU Scores (EN → ZH):**

Model	BLEU Score
Google Translate (Baseline)	8.88
Argos Translate	1.98
M2M100 1.2B	7.92
M2M100 418M	3.88
LibreTranslate	0.00

- **Observations:**

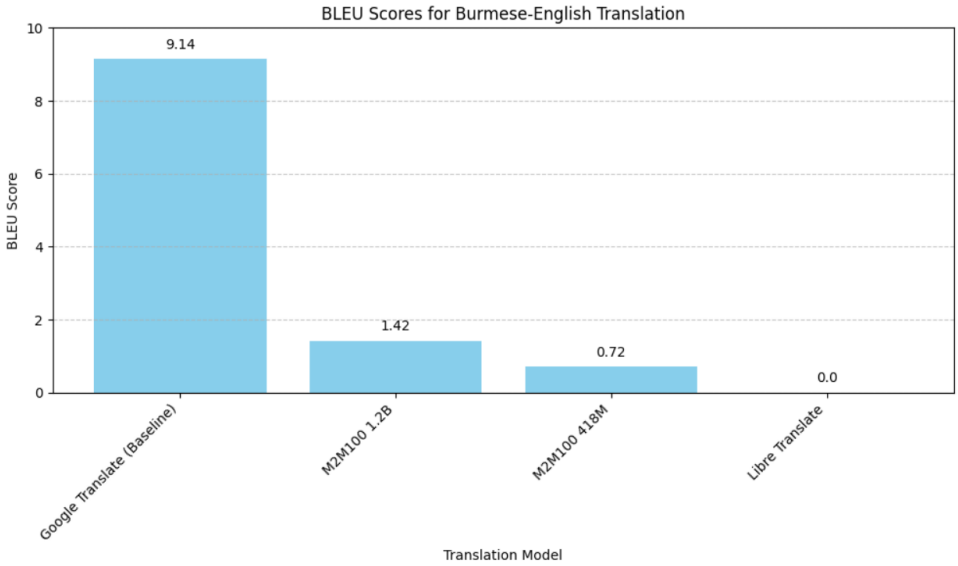

Figure 1: BLEU Scores for Burmese-English Translation

- Google Translate again showed the highest accuracy.
- M2M100 1.2B demonstrated strong performance, with a BLEU score of 7.92, coming relatively close to Google Translate's 8.88. This indicates its effectiveness for a widely supported language pair like Chinese-English.
- M2M100 418M performed significantly better for Chinese-English (3.88) compared to Burmese-English (0.72), suggesting better training data or model suitability for this pair.
- Argos Translate's performance was lower at 1.98.
- LibreTranslate again failed to produce translations due to "Bad Request" errors, resulting in a 0.00 BLEU score.

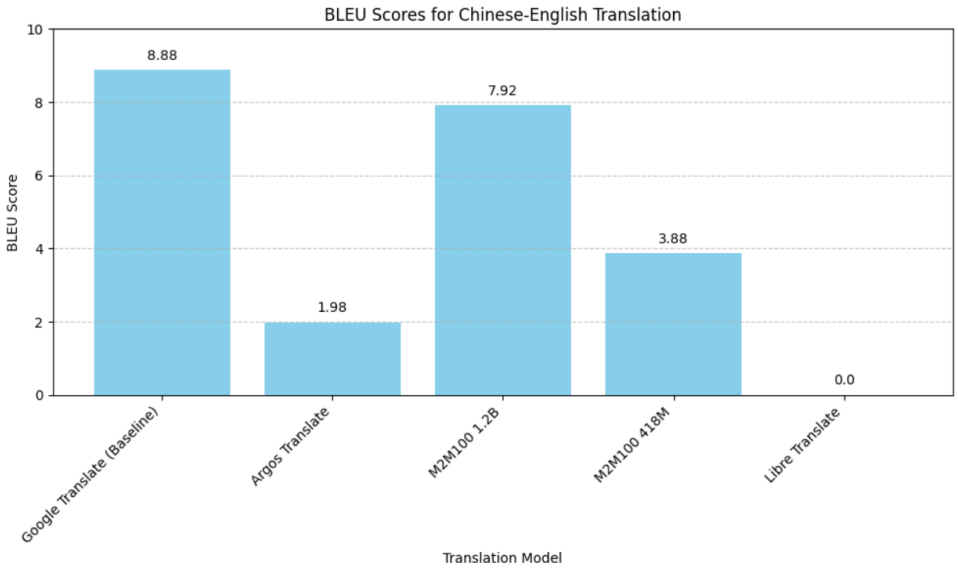

Figure 2: BLEU Scores for Chinese-English Translation

6.3. Hindi-English Translation

- **Benchmark Dataset Used:** `IITB.en-hi.en` and `IITB.en-hi.hi` (50 sentence pairs).

- **BLEU Scores (EN → HI):**

Model	BLEU Score
Google Translate (Baseline)	8.86
Argos Translate	5.56
M2M100 1.2B	4.53
M2M100 418M	5.51
LibreTrnaslate	6.94

- **Observations:**
  - Google Translate maintains its lead with a strong BLEU score of 8.86.
  - Interestingly, for Hindi-English, Argos Translate (5.56) and M2M100 418M (5.51) performed comparably and slightly better than M2M100 1.2B (4.53). This might be due to the specific nature of the dataset or the models' training distribution for this particular language pair.
  - The absence of LibreTranslate results is due to its being commented out in the provided code for this language pair.
  - LibreTranslate also performed well respect to others

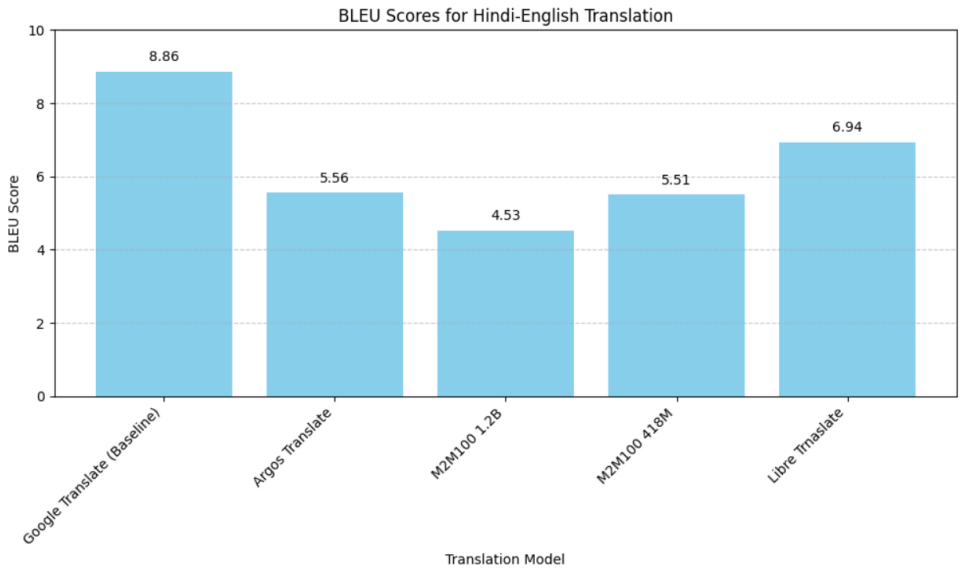

Figure 3: BLEU Scores for Hindi-English Translation

---

7. Overall Observations and Challenges

- **General Performance Trends:** Google Translate consistently achieved the highest BLEU scores across all language pairs, reaffirming its position as a leading online translation service. Among the offline models, **M2M100 1.2B** generally performed best for well-resourced languages like Chinese-English, while for less common pairs like Burmese-English, all offline models struggled. For Hindi-English, **Argos Translate** and **M2M100 418M** showed competitive performance.
- **Computational Resources:** Running M2M100 1.2B requires substantial RAM and GPU VRAM, making it challenging for systems without dedicated GPUs. The 418M version is more accessible but still benefits from GPU.
- **Limitations of Current Offline Models:** The results suggest a significant gap between the performance of state-of-the-art online services (Google Translate) and currently available generic

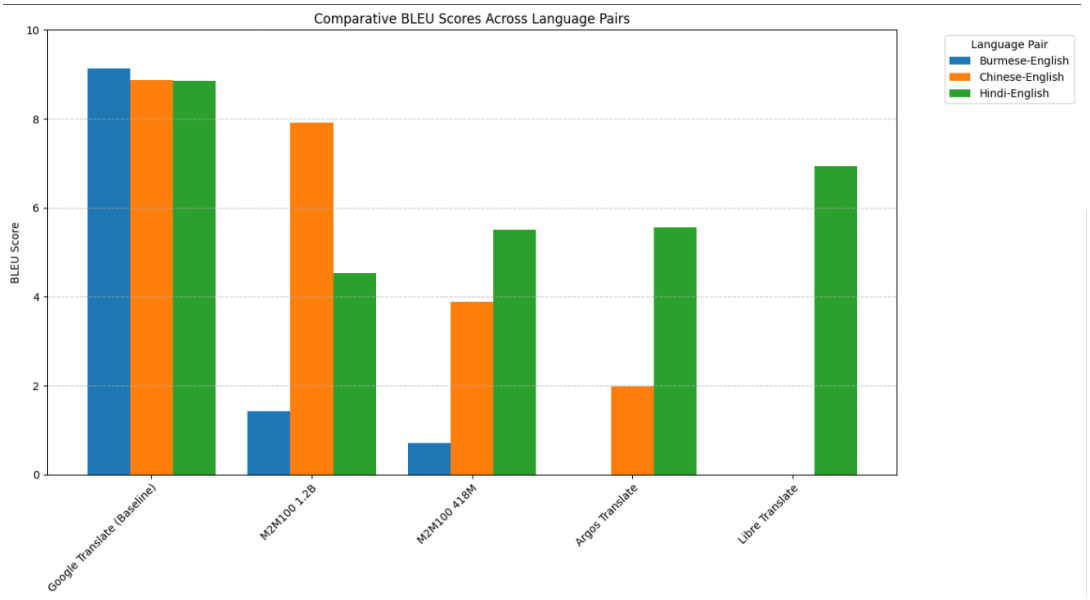

Figure 4: Comparative BLEU Scores Across All Language Pairs

offline models, especially for languages with fewer readily available parallel corpora. Fine-tuning these models on domain-specific data or exploring more specialized architectures might be necessary to close this gap.

- **Challenges Encountered:**
  - **LibreTranslate API Issues:** The primary challenge was the consistent `HTTP Error 400: Bad Request` from the LibreTranslate API, which prevented its evaluation for Burmese-English and Chinese-English translations. This highlights the dependency and potential instability when relying on external online API endpoints, even for "offline-capable" tools when accessed via an external server.
  - **Dataset Limitations:** While benchmark datasets were used, the limited size (50 sentences) might not fully capture the nuanced performance differences or real-world translation quality across diverse contexts.
  - **Computational Time:** Even with batching, translating 50 sentences with large models like M2M100 can take noticeable time on a CPU, emphasizing the need for optimized inference for a practical offline application.

---

8. Future Plans and Next Steps

Based on the findings of this comparison, the next steps will involve:

- **Deep Dive into Best-Performing Offline Models:** Focusing on optimizing the integration of **M2M100 1.2B** (for Chinese-English) and exploring ways to improve performance for less common language pairs like Burmese-English. This might involve exploring alternative models or investigating methods for model compression/quantization.
- **Addressing LibreTranslate:** Investigating the cause of the LibreTranslate errors. This could involve trying a different LibreTranslate server, setting up a local instance, or identifying if there's a specific issue with the input data format.
- **Quantitative and Qualitative Analysis:** Beyond BLEU scores, conducting a more qualitative analysis of translation outputs to understand specific error types (e.g., grammatical errors, lexical choices, fluency issues).

- **Expanding Dataset Size:** If feasible, evaluating models on larger subsets of benchmark datasets to gain more statistically robust results.

—

9. Appendix

9.1. Code Snippets

Relevant Python code snippets used for model loading, inference, and BLEU score calculation.

Burmese to English Translation Code

```
!pip install -q transformers sentencepiece sacrebleu argostranslate libretranslatepy googletrans==4.


import json, gzip, time
import xml.etree.ElementTree as ET
import torch
import sacrebleu
from transformers import M2M100ForConditionalGeneration, M2M100Tokenizer
from googletrans import Translator
from argostranslate import package, translate
from libretranslatepy import LibreTranslateAPI
from google.colab import files


print("Upload your ‘en-my.tmx.gz‘ file")
uploaded = files.upload()
tmx_file = next(iter(uploaded))   # filename

import xml.etree.ElementTree as ET
import gzip


with gzip.open("en-my.tmx.gz", "rb") as f:
    tree = ET.parse(f)
    root = tree.getroot()


en_sentences = []
my_sentences = []

for tu in root.findall(".//tu"):
    langs = {}
    for tuv in tu.findall("tuv"):
        lang = tuv.attrib.get("{http://www.w3.org/XML/1998/namespace}lang")
        seg = tuv.find("seg")
        if lang and seg is not None and seg.text:
            langs[lang] = seg.text.strip()
    if "en" in langs and "my" in langs:
        en_sentences.append(langs["en"])
        my_sentences.append(langs["my"])


en_sentences = en_sentences[:50]
my_sentences = my_sentences[:50]
```

```
print(f"Loaded {len(en_sentences)} English and {len(my_sentences)} Burmese sentences.")
print(" Example:")
print("MY:", my_sentences[0])
print("EN:", en_sentences[0])
```

```
def translate_argos(sentences):
    return [argos_translator.translate(s) for s in sentences]
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
```

```
def load_m2m_model(model_name):
    tokenizer = M2M100Tokenizer.from_pretrained(model_name)
    model = M2M100ForConditionalGeneration.from_pretrained(model_name).to(device)
    return tokenizer, model

tokenizer_1b, model_1b = load_m2m_model("facebook/m2m100_1.2B")
tokenizer_418m, model_418m = load_m2m_model("facebook/m2m100_418M")

def translate_m2m(sentences, tokenizer, model, src_lang="my", tgt_lang="en", batch_size=4):
    tokenizer.src_lang = src_lang
    results = []
    for i in range(0, len(sentences), batch_size):
        batch = sentences[i:i+batch_size]
        encoded = tokenizer(batch, return_tensors="pt", padding=True, truncation=True).to(device)
        generated = model.generate(**encoded, forced_bos_token_id=tokenizer.get_lang_id(tgt_lang))
        decoded = tokenizer.batch_decode(generated, skip_special_tokens=True)
        results.extend(decoded)
    return results
```

```
def translate_libre(sentences, src_lang="my", tgt_lang="en"):
    lt = LibreTranslateAPI("https://b458-49-43-7-49.ngrok-free.app")
    translations = []
    for text in sentences:
        try:
            if not text.strip():
                translations.append("")
                continue
            translated = lt.translate(text, source=src_lang, target=tgt_lang)
            translations.append(translated)
        except Exception as e:
            print(f"LibreTranslate error: {text[:30]}... | {e}")
            translations.append("")
            time.sleep(1)
    return translations

translator = Translator()
def translate_google(sentences, src_lang="my", tgt_lang="en"):
    return [translator.translate(s, src=src_lang, dest=tgt_lang).text for s in sentences]
```

```
def compute_bleu(hypotheses, references):
    references = [[ref] for ref in references]
    bleu = sacrebleu.corpus_bleu(hypotheses, list(zip(*references)))
    return bleu.score


#print("Argos Translate...")
#argos_translations = translate_argos(my_sentences)

print("M2M100 1.2B...")
m2m100_1b_translations = translate_m2m(my_sentences, tokenizer_1b, model_1b)

print("M2M100 418M...")
m2m100_418m_translations = translate_m2m(my_sentences, tokenizer_418m, model_418m)

print("Google Translate...")
google_translations = translate_google(my_sentences)

print("LibreTranslate...")
libre_translations = translate_libre(my_sentences)


print("\n=== BLEU SCORES (MY → EN) ===")
#print(f"Argos Translate:      {compute_bleu(argos_translations, en_sentences):.2f}")
print(f"M2M100 1.2B:          {compute_bleu(m2m100_1b_translations, en_sentences):.2f}")
print(f"M2M100 418M:          {compute_bleu(m2m100_418m_translations, en_sentences):.2f}")
print(f"Google Translate:     {compute_bleu(google_translations, en_sentences):.2f}")
print(f"LibreTranslate:       {compute_bleu(libre_translations, en_sentences):.2f}")
```

Chinese to English Translation Code

```
!pip install -q transformers sentencepiece sacrebleu argostranslate libretranslatepy googletrans==4.


import json
import torch
import sacrebleu
import time
from transformers import M2M100ForConditionalGeneration, M2M100Tokenizer
from googletrans import Translator
from argostranslate import package, translate
from libretranslatepy import LibreTranslateAPI
from google.colab import files


uploaded = files.upload()
jsonl_file = next(iter(uploaded))   # filename of uploaded file


en_sentences = []
zh_references = []
with open(jsonl_file, "r", encoding="utf-8") as f:
    for line in f:
        data = json.loads(line)
        en_sentences.append(data["english"])
        zh_references.append(data["chinese"])
```

```
en_sentences = en_sentences[:50]
zh_references = zh_references[:50]


available_packages = package.get_available_packages()
en_zh_package = next((p for p in available_packages if p.from_code == "en" and p.to_code == "zh"), N
if en_zh_package:
    package.install_from_path(en_zh_package.download())

installed_languages = translate.get_installed_languages()
en_lang = next(filter(lambda x: x.code == "en", installed_languages))
zh_lang = next(filter(lambda x: x.code == "zh", installed_languages))
argos_translator = en_lang.get_translation(zh_lang)

def translate_argos(sentences):
    return [argos_translator.translate(s) for s in sentences]


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

def load_m2m_model(model_name):
    tokenizer = M2M100Tokenizer.from_pretrained(model_name)
    model = M2M100ForConditionalGeneration.from_pretrained(model_name).to(device)
    return tokenizer, model

tokenizer_1b, model_1b = load_m2m_model("facebook/m2m100_1.2B")
tokenizer_418m, model_418m = load_m2m_model("facebook/m2m100_418M")

def translate_m2m(sentences, tokenizer, model, src_lang="en", tgt_lang="zh", batch_size=4):
    tokenizer.src_lang = src_lang
    results = []
    for i in range(0, len(sentences), batch_size):
        batch = sentences[i:i+batch_size]
        encoded = tokenizer(batch, return_tensors="pt", padding=True, truncation=True).to(device)
        generated = model.generate(**encoded, forced_bos_token_id=tokenizer.get_lang_id(tgt_lang))
        decoded = tokenizer.batch_decode(generated, skip_special_tokens=True)
        results.extend(decoded)
    return results


def translate_libre(sentences, src_lang="en", tgt_lang="zh"):
    lt = LibreTranslateAPI("https://b458-49-43-7-49.ngrok-free.app")
    translations = []
    for text in sentences:
        try:
            if not text.strip():
                translations.append("")
                continue
            translated = lt.translate(text, source=src_lang, target=tgt_lang)
            translations.append(translated)
        except Exception as e:
            print(f"LibreTranslate failed for: {text[:30]}... | Error: {e}")
            translations.append("")
            time.sleep(1)   # prevent rate-limiting
    return translations
```

```
translator = Translator()
def translate_google(sentences, src_lang="en", tgt_lang="zh-cn"):
    return [translator.translate(s, src=src_lang, dest=tgt_lang).text for s in sentences]


def compute_bleu(hypotheses, references):
    references = [[ref] for ref in references]
    bleu = sacrebleu.corpus_bleu(hypotheses, list(zip(*references)))
    return bleu.score

print("Translating with Argos Translate...")
argos_translations = translate_argos(en_sentences)

print("Translating with M2M100 1.2B...")
m2m100_1b_translations = translate_m2m(en_sentences, tokenizer_1b, model_1b)

print("Translating with M2M100 418M...")
m2m100_418m_translations = translate_m2m(en_sentences, tokenizer_418m, model_418m)

print("Translating with Google Translate...")
google_translations = translate_google(en_sentences)

print("Translating with LibreTranslate...")
libre_translations = translate_libre(en_sentences)

print("\n=== BLEU SCORES ===")
print(f"Argos Translate BLEU: {compute_bleu(argos_translations, zh_references):.2f}")
print(f"M2M100 1.2B BLEU:     {compute_bleu(m2m100_1b_translations, zh_references):.2f}")
print(f"M2M100 418M BLEU:     {compute_bleu(m2m100_418m_translations, zh_references):.2f}")
print(f"Google Translate BLEU:{compute_bleu(google_translations, zh_references):.2f}")
print(f"LibreTranslate BLEU:  {compute_bleu(libre_translations, zh_references):.2f}")
```

Hindi to English Translation Code

```
!pip install -q transformers sentencepiece sacrebleu argostranslate libretranslatepy googletrans==4.

import os
import sacrebleu
from transformers import M2M100ForConditionalGeneration, M2M100Tokenizer
from googletrans import Translator
from libretranslatepy import LibreTranslateAPI
from argostranslate import package, translate


# Upload your IITB.en-hi.en and IITB.en-hi.hi via Colab upload:


# Read the files
with open("IITB.en-hi.en", encoding='utf-8') as f:
    en_sentences = [line.strip() for line in f.readlines() if line.strip()]
with open("IITB.en-hi.hi", encoding='utf-8') as f:
    hi_references = [line.strip() for line in f.readlines() if line.strip()]

# Limit number for fast testing
en_sentences = en_sentences[:50]
hi_references = hi_references[:50]
```

```
# Install Hindi-English package
available_packages = package.get_available_packages()
hi_en_package = next((p for p in available_packages if p.from_code == "en" and p.to_code == "hi"), N

if hi_en_package:
    package.install_from_path(hi_en_package.download())

installed_languages = translate.get_installed_languages()
en_lang = next(filter(lambda x: x.code == "en", installed_languages))
hi_lang = next(filter(lambda x: x.code == "hi", installed_languages))
argos_translator = en_lang.get_translation(hi_lang)

def translate_argos(sentences):
    return [argos_translator.translate(s) for s in sentences]


model_1b = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_1.2B")
tokenizer_1b = M2M100Tokenizer.from_pretrained("facebook/m2m100_1.2B")

def translate_m2m100_1b(sentences, src_lang="en", tgt_lang="hi"):
    tokenizer_1b.src_lang = src_lang
    results = []
    for s in sentences:
        encoded = tokenizer_1b(s, return_tensors="pt")
        generated = model_1b.generate(**encoded, forced_bos_token_id=tokenizer_1b.get_lang_id(tgt_la
        decoded = tokenizer_1b.batch_decode(generated, skip_special_tokens=True)[0]
        results.append(decoded)
    return results


model_418m = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_418M")
tokenizer_418m = M2M100Tokenizer.from_pretrained("facebook/m2m100_418M")

def translate_m2m100_418m(sentences, src_lang="en", tgt_lang="hi"):
    tokenizer_418m.src_lang = src_lang
    results = []
    for s in sentences:
        encoded = tokenizer_418m(s, return_tensors="pt")
        generated = model_418m.generate(**encoded, forced_bos_token_id=tokenizer_418m.get_lang_id(tg
        decoded = tokenizer_418m.batch_decode(generated, skip_special_tokens=True)[0]
        results.append(decoded)
    return results


translator = Translator()
def translate_google(sentences, src_lang="en", tgt_lang="hi"):
    return [translator.translate(s, src=src_lang, dest=tgt_lang).text for s in sentences]


#lt = LibreTranslateAPI("https://libretranslate.de")    # or your local server
#def translate_libre(sentences, src_lang="en", tgt_lang="hi"):
#    return [lt.translate(s, source=src_lang, target=tgt_lang) for s in sentences]


def compute_bleu(hypotheses, references):
    references = [[ref] for ref in references]
```

```
    bleu = sacrebleu.corpus_bleu(hypotheses, list(zip(*references)))
    return bleu.score


print("Translating with Argos...")
argos_translations = translate_argos(en_sentences)

print("Translating with M2M100 1.2B...")
m2m100_1b_translations = translate_m2m100_1b(en_sentences)

print("Translating with M2M100 418M...")
m2m100_418m_translations = translate_m2m100_418m(en_sentences)

print("Translating with Google Translate...")
google_translations = translate_google(en_sentences)

#print("Translating with LibreTranslate...")
#libre_translations = translate_libre(en_sentences)


print("\n=== BLEU SCORES ===")
print(f"Argos Translate BLEU: {compute_bleu(argos_translations, hi_references):.2f}")
print(f"M2M100 1.2B BLEU:     {compute_bleu(m2m100_1b_translations, hi_references):.2f}")
print(f"M2M100 418M BLEU:     {compute_bleu(m2m100_418m_translations, hi_references):.2f}")
print(f"Google Translate BLEU:{compute_bleu(google_translations, hi_references):.2f}")
#print(f"LibreTranslate BLEU:  {compute_bleu(libre_translations, hi_references):.2f}")
```