Machine Translation Model Comparisons for English to French

Aylin Elmali
aelmali@umass.edu

Sapna Parihar
sparihar@umass.edu

Sarah Boal
sboal@umass.edu

Albert Galimov
agalimov@umass.edu

Deepa Rukmini Mahalingappa
drukminimaha@umass.edu

1 Problem statement

Even though we live in such a connected world, language barriers can prevent us from maximizing cross-cultural communication and collaboration. Luckily, machine translation was created to efficiently address this issue, but syntactical differences in languages as well as cultural and contextual differences still make it difficult to obtain higher-quality machine translations. Our goal for our final project is to compare different translation models that can be utilized to improve machine translation even when cultural, contextual, and syntactical differences are present. Due to time constraints, we decided to focus mainly on machine translation from English to French. Enhancing machine translation can potentially have a positive global impact through advancements in cultural exchange, international business, diplomacy, education, and many other domains.

2 What we proposed vs. what we accomplished

We initially decided to only work on one English to French translation model that uses the T5 architecture. Here is a list of the tasks that we proposed to do in the project proposal to build that specific model:

- Collect a sizable parallel English-French text corpus/ pre-process data from the text corpus to create a dataset
- Build and train a T5 model, and examine output and performance
- Evaluate the model's performance using metrics like BLEU
- Possibly deploy the model for practical use

However, instead of only working on one model that can improve machine translation, we decided to work on six models that use different model architectures and sets of parameters. This approach can help us compare the models and see which model(s) would be more beneficial for the task of English-to-French machine translation. Here are the updated tasks that we worked on after deciding to work on multiple models instead of one:

- ~~Collect a sizable parallel English-French text corpus/ acquire and pre-process data from the text corpus to create a dataset~~
- ~~Build and train various mt5 and LLAMA models on collected dataset, and examine output and performance~~
- ~~Evaluate models using metrics such as BLEU and COMET~~
- ~~Perform human evaluation on outputs and perform in-depth error analysis to compare performance of each model~~

3 Related work

The paper introducing mT5 from (Xue et al., 2021) describes it as a multilingual variant of T5 pretrained on the mC4 dataset for multilingual benchmarks while deviating little from T5. The paper describes the mC4 dataset and that only pretrained models were released. When we were testing zero-shot setting we didn't experience partial translation into the wrong language, but zero-shot was unsuccessful since mT5 needs to be fine-tuned before being used on a downstream task. We also fine-tuned LLAMA from Meta AI, according to (Touvron et al., 2023) released in 2023. Compared to mT5 it has 7B parameters and would be expected to achieve higher BLEU and COMET score metrics for machine translation.

Our human evaluation methodology is firmly based on the MQM (Multidimensional Quality Metrics) Framework, an established tool for setting evaluation benchmarks across various language pairs, including recent applications to English-Korean by (Park and Padó, 2024). This framework was also the focus of an extensive study by (Freitag et al., 2021). The MQM Framework is designed to achieve reliable human evaluations by providing a detailed and systematic approach to error identification. In the MQM Framework, errors are categorized into several main areas: Accuracy, Fluency, Terminology, etc., each containing numerous subcategories to capture the nuances of language quality. In the original MQM scoring system, evaluators score from 0 to 25, where lower scores indicate better quality. However, to simplify the evaluation process and enhance clarity, we adopted a 1-3 scoring metric. In this adapted system, higher scores represent better quality, with 1 indicating significant issues and 3 representing high quality with minimal errors.

According to (Papineni et al., 2002), utilizing an evaluation metric like BLEU is a quick technique that developers can use to measure the quality of their machine translation models, enabling developers to make adjustments to their code much more quickly since they don't need to wait on human evaluation. However, (Wieting et al., 2019) mentions the drawbacks of BLEU and how it's not able to successfully measure sentences that are lexically different even though they have semantic similarities. Instead, the paper introduces another evaluation metric called SIMILE that is able to measure the semantic similarities between machine translations and their corresponding reference translations. We didn't use SIMILE to measure the quality of the translations from our models. However, we could use the metric in the future to better compare our models for higher-quality machine translation.

A neural framework for training multilingual machine translation evaluation models is COMET, used as a metric in addition to BLEU for comparing the translations of our models. According to (Rei et al., 2020), this framework is PyTorch-based for training highly multilingual and adaptable machine translation evaluation models and can be used for training MT evaluation models that serve as automatic metrics, adapted to human judgements of machine translation quality. COMET is more recent (developed in 2020 versus BLEU developed in 2019) and captures semantic similarity between machine-translation-generated and human reference translations, therefore could be a better option as a metric.

The preexisting models that can conduct neural machine translation such as RNNSearch (Bahdanau et al., 2015) have laid the foundation for us, addressing the potential concern of translating long sentences and having to encode them into a fixed-length vector. In the preprocessing phase of our project, we filtered our sentences that were too long or contained corrupted translations.

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation by (Wu et al., 2016) introduces a system which aims to address several challenges such as handling rare words and reducing computational costs. This is a relevant past work that is relevant for our dataset for training or testing as it contains acronyms and names of sites, places, etc. that may have not been in the mC4 dataset.

”Massively Multilingual Neural Machine Translation” by (Aharoni et al., 2019) explores the challenges and opportunities of training a single neural machine translation model on data from multiple languages, including English and French. This work is a predecessor to the mT5 model that we used but was a good reference for how BLEU scores can be used for neural machine translation.

4 Your dataset

Our dataset that we used for model fine-tuning for this project was a large 8.41 GB English-French Translation Dataset from Kaggle created by Chris Callison-Burch, and used in the 2015 Workshop on Statistical Machine Translation (Bojar et al., 2015). The dataset consists of over 22.5 million translation pairs with data crawled from millions of web pages. Some example English/French pairs from the dataset are: ”Site map - Plan du site”, ”Today it is used to show the sky to the general public. - On l'utilise aujourd'hui pour montrer le ciel au grand public.”, ”What is light ? - Qu'est-ce que la lumière?”.

We did identify certain flaws and limitations with the dataset. For instance, the dataset was obtained by using simple heuristics to transform French URLs into English URLs, and it was assumed by the data collector that these documents are translations of each other. However for some instances this assumption may not have been accurate, meaning some of the translations pairs were incorrect. Some of the English sentences were also not in English. Another limitation was that because the dataset consisted of random bits of text crawled from the internet, the sentences included did not always consist of coherent full sentences and could sometimes be short clipped phrases or other web-specific text such as links which were not particularly useful for translation, which may have also placed limitations on the coherency of the model.

4.1 Data preprocessing

As the dataset consisted of text crawled from the internet, preprocessing of data was necessary before it could be fed into a model. We opted for shorter sentences for more effective training, so we filtered the dataset to only use sentences that were 100 characters or less. We also noted that some of the French translations in the dataset appeared to be corrupted and had unusual symbols, so we filtered out any pairs with corrupted translations. Finally, we also reduced the output to limit the size and make training more manageable and to handle compute limits. The final filtered file that we used was 2,452,224 lines in total. The preprocessing code can be found in preprocess.py.

4.2 Data annotation

We were only able to contact one person who was fluent in our target language, so interannotator agreement could not be collected. We asked a human evaluator to evaluate the translations produced by each model, adapting guidelines from the MQM (Multidimensional Quality Metrics) Framework (Freitag et al., 2021). The instructions for the annotator were as follows:

”In each excel file will be two columns: “en” will have the original sentence or phrase in English, and “fr” will contain its French translation produced by the model. You will be assessing the French translations based on three different metrics: Fluency, Adequacy, and Accuracy. The remaining three columns will have room for you to place your evaluations. For each translation, you will rank it on a scale of 1-3 for each metric:

Fluency:

1. The translation is completely incomprehensible or nonsensical
2. Translation is partially understandable but with significant grammatical errors or awkward phrasing
3. Translation is fluent and natural-sounding

Adequacy:

1. The translation completely misses the meaning of the original text
2. The translation captures some meaning but lacks context or details
3. The translation fully captures the meaning and context of the original text

Accuracy:

1. The translation contains numerous mistranslations, omitted or extra words, or words which are not translated
2. Translation contains some errors but the overall meaning is conveyed
3. Fully accurate translation with minimal to no errors, untranslated words, or omitted or extra words.

Read the original text carefully to understand its meaning and context. Then, evaluate the French translation for each metric and decide on your score. For each translation, place your score of 1-3 in the column for each metric.”

30 translation pairs were provided to the annotator for each model.

5 Your approach

For our machine translation task from English to French we decided to especially focus on mT5 model (A massively multilingual pre-trained text-to text transformer) with 300M parameters and Llama 2 pre-trained and fine – tuned Large Language Model with 7B parameters. Our approaches are structured into two parts:

1. Exploration of mT5 model for machine translation task with different fine-tuning methods (full fine-tuning, prompt tuning and Quantized Low-Rank adaptation (QLoRa))
2. Discussion about Llama 2 model for machine translation task with using two methods of fine-tuning: Quantized Low-Rank adaptation and full fine-tuning

Nowadays, transformer-based model plays a significant role in machine translation due to ability to handle complex linguistic parts and provide high-quality translation in real-time. One of the brightest representatives of transformers is the T5 model (Text-to-Text Transfer Transformer), but we decided to move forward with multilingual variant mT5, that was designed to especially handle multiple languages that makes this model perfect to machine translation tasks. We were interested in creating a high-quality model for text translation, so we decided not to stop at one model and our choice fell on the Llama 2 model. This model is known for performance in various natural language processing task.

By performing these models to our downstream task, we would like to compare their performance and accuracy in translating English to French, using different types of fine-tuning techniques to see outputs. For comparative analysis we will use 2 evaluation metrics: BLEU score and COMET score. Additionally, we found an annotator that can evaluate outputs from models.

Fine-tuning mT5: Unlike the T5 model, mT5 (Xue et al., 2021) does not have ”out-of-the-box” functionality and has to be fine-tuned before it can be used for any downstream task. In this section we will build our model from scratch. We explored 3 methods of fine-tuning techniques, each method has unique advantages and disadvantages, and our goal is evaluate each of them.

1) Full fine-tuning

In this approach, we use Google Colab Pro L4 GPU with High RAM, HuggingFace Machine translation Tutorial, HuggingFace libraries for accessing mT5 (MT5For ConditionalGeneration), tokenization (AutoTokenizer), training (Seq2Seq Training Arguments and Seq2Seq Trainer) and data pairing and preprocessing. We trained the model two times with different numbers of lines 100,000 and 200,000. The train-val-test split that we used was 80% for training, 10% for validation, and 10% for testing.

For a model with 100 000 lines we decided to move forward with the following combinations of hyperparameters that gave us a good balance between training time, CUDA memory and quality of translation: learning rate = 2e-5, batch size = 32, number of epochs = 10. Total training took around 3 hours and loss results for model after full training are: training loss - 3.366678 and validation loss - 2.086953. Model can be found under the name ”mT5 fine tuning 100k.ipynb”.

We also wanted to look at the model with a different set up. We used the same libraries, but hyperparameters were changed: learning rate = 2e-5, batch size = 32, number of epochs = 6. We increased the number of lines twice (200 000 lines), but decreased the number of epochs. Total training time took around the same as for 100 000 lines - around 3 hours and losses results are: training loss - 2.625600 and validation loss - 2.039549.

One problem that was during training is that we couldn't add in function Seq2Seq Trainer parameter ”compute metrics”, due to GPU RAM limitations (out of CUDA memory). We created another file in order to make computations separately and called this file ”mt5 fine tuning Load.ipynb” where the model was loaded and evaluation scores were computed.

2) Prompt-tuning mT5: This fine-tuning method for mT5 involves adding prompt embeddings, getting input embeddings, and concatenating prompts. Then, the attention mask is adjusted for the prompt embeddings and passed through the model. All original mT5 parameters are frozen and only the prompt embeddings are trained. The Google Colab T4 GPU with High RAM was mainly used with Hugging Face libraries for accessing mT5 , tokenization (MT5Tokenizer), and training (AutoModelForSeq2SeqLM). The learning rate for all runs was 1e-3, batch size was 10 and number of epochs was 2. The model can be found under the file name ” PromptTuning_model_fine_tuning.ipynb” and the file loading the model was ”mT5 prompt tuning load.ipynb”. The file with the loaded model is where the evaluation scores were computed.

This Prompt Tuned model was initially run on a range of 5,000 sentences to 100,000 sentences with 90 percent of the data for training and 10 percent for testing. From the test output, it was clear that using a dataset size of 5,000 sentences was not enough as the BLEU score was almost 0.0 (very small) and the output was just extra_id tokens. After training and testing a range of dataset sizes from the preprocessed data file, the outputs from the model improved significantly when using dataset sizes of 50,000 sentences and higher. Thus, moving forward the focus of the training and testing of the model was on a dataset of 100,000 sentences and greater, if possible. The amount of time for training 90,000 sentences and testing 10,000 sentences was about 6 hours for each run. One run of the model was conducted with a dataset of 200,000 sentences however due to compute limitations with GPU and also training time taking almost 12 hours on Google Colab using a T4 GPU with High RAM, this was only run once to obtain a BLEU score for comparison with other models.

BLEU score was calculated with SacreBleu and COMET was determined using Unbabel COMET metric. Compute capacity, training time, memory limitations and GPU usage were all factors that limited the amount of training and testing data that could be run through our models for mT5. This is also one major reason that we had to limit the model to mT5-small. One potential project for the future would be to try running our dataset on the larger mT5 models, if we can access multiple GPUs instead of the capabilities of Google Colab or a Single GPU.

3) Fine-tuning mT5 with QLoRA

In this approach, the model was first quantized using BitsAndBytesConfig from the BitsAndBytes library in HuggingFace. The parameters for this were: load in 4bit set to true, which loads the model in 4-bit precision bnb 4bit use double quant set to true to enable double quantization, bnb 4bit quant type set to ”nf4”, and bnb 4bit compute dtype set to ”bfloat16”, a 16-bit floating-point representation. Afterwards, the model was prepared for k-bit training and the LoRA model was set up using the PEFT library. The parameters for LoraConfig were: rank (r) = 8, lora alpha = 32 for scaling, target modules = [”q”, ”v”] for a T5 model, lora dropout = 0.05, bias = ”none”, and task type set to ”SEQ 2 SEQ LM” for sequence-to-sequence modeling. The total dataset used for this was 200k lines. The train-val-test split that we used was 80% for training, 10% for validation, and 10% for testing.

This setup reduced the number of trainable parameters in the model to 344,064 out of a total of 300,520,832 parameters. Other parameters such as learning rate and batch size were kept consistent with the regular mT5 fine-tuning model, however in the Seq2SeqTrainingArguments an additional optimizer ”paged adamw 8bit” was used. These parameters were chosen to make the fine-tuning process more efficient and optimize performance while reducing computational workload. Total training took about 3 hours. The code for running this model can be found in ”mt5_QLoRA.ipynb”

Fine-tuning LLaMA:

1) Fine-tuning with QLoRA Using Unsloth Library

This approach relies on the open-source version of Unsloth, which is a library/tool that can help train models at a faster rate with less memory usage. The open-source version of Unsloth is able to support Llama 2, which is a pretrained large language model created by Meta AI, and 4-bit LoRA or QLoRA (quantized low-rank adaptation), which is a technique that combines quantization and low-rank adaptation. Quantization is used to help further reduce memory usage and make the process of model inference much faster. Low-rank adaptation is used to reduce the number of trainable parameters in a model. Both Llama 2 and QLoRA are used to build the model for this specific approach.

Firstly, a 4-bit prequantized pretrained model, specifically unsloth/llama-2-7b-bnb-4bit, and its corresponding tokenizer are obtained from the Unsloth library by using the FastLanguageModel.from_pretrained() function. A BitsAndBytesConfig was added into the Unsloth pretrained model to make sure that 4-bit quantization was being utilized as well. This pretrained model then acts as a base model that is wrapped around the FastLanguageModel.get_peft_model() function, which allows the addition of more parameters that enable low-rank adaptation (LoRA). Here are some of the parameters used to enable LoRA: r is set to 16, lora_alpha is set to 16, and lora_dropout is set to 0.

After taking the pretrained Llama 2 model and adding parameters that enable 4-bit quantization and low-rank adaptation, the data needs to be formatted for training and evaluation. 100,000 rows are taken from the filtered dataset that we created from the text corpus and are split into a training dataset (80,000 rows), a validation dataset (10,000 rows), and a test dataset (10,000 rows). The training dataset and the validation dataset are formatted with the formattingpromptsfunc function, which prepends the instruction/prompt “Translate from English to French:” and appends a tokenizer.eostoken (a token that represents the end of a sentence) to each English sentence and French sentence pair (each row of data).

The model is then trained on the training dataset alongside the validation dataset with the help of the SFTTrainer. We initially tried to use the Seq2SeqTrainer instead of the SFTTrainer with the model, but we dealt with multiple errors. Based on information from (Han-Chen, 2024), which is a HuggingFace article about Unsloth, the models from Unsloth can be used with the trainer suite from the TRL library, so we decided to stick with SFTTrainer, which is from that TRL trainer suite. We also added additional hyperparameters to the SFTTrainer to customize the training process with the help of TrainingArguments. Here are some of the hyperparameters that were added to the trainer: per_device_train_batch_size is set to 4, per_device_eval_batch_size is set to 4, learning_rate is set to 5e-04, and num_train_epochs is set to 1. The tokenizer is integrated into the SFTTrainer as well. During training, a total of 20,000 steps were taken and 39,976,960 parameters were present. Training only happened with one epoch, and it took 2 hours and 30 minutes to complete training. After running trainer.evaluate(), it returned an eval loss of around 0.6983.

After training was complete, the test dataset was formatted as well by prepending the same instruction/prompt “Translate from English to French:” to the English sentence from each row without adding the corresponding French sentence from the dataset. Not adding the French sentences allows the model to generate outputs on the test dataset that contain the French translations. model.generate() is then used on 200 lines of the test dataset to generate the outputs containing the translated French sentences. Finally, we take the machine translated French sentences from the outputs, the corresponding French sentences from the 200 lines of test data, and the corresponding English sentences from the 200 lines of test data to then compute the average BLEU score and the average COMET score. For this model, the average BLEU score is 0.5205 and the average COMET score is 0.7686.

We were able to complete a working implementation. To accomplish this, we used these libraries: torch, datasets, pandas, csv, drive from google.colab, unsloth, trl, transformers, nltk.translate.bleu_score, evaluate, and numpy. The code for this model took inspiration from an existing implementation from Unsloth. The existing implementation is a free Google Colab Notebook created by Unsloth that contains an example of how to use the Unsloth library with Llama 2, which is linked here.

The code built for this approach can be found in QLoRa_unsloth.ipynb and the computations for the BLEU and COMET score for this approach can be found in QloraLlamaUnslothCOMETandBLEU.ipynb. Another file that is associated with this approach is qlorallamaresults.csv, which contains the 200 machine translated French sentences with their corresponding French sentences and English sentences from the test dataset. The code was built using Google Colab, and the first Colab notebook mentioned above in this paragraph requires A100 GPU to run. Some issues that we dealt with while using Google Colab was CUDA out of memory errors. We would have generated outputs for more than 200 lines of data, but we couldn’t due to the limited memory that exists using Google Colab.

2) Fine-tuning LLAMA with Quantization-aware LoRA Library

This is a technique designed to enhance the efficiency of fine-tuning large language models by integrating quantization directly into the LoRA framework. This approach allows models to be fine-tuned with significantly reduced computational resources while maintaining high accuracy.

QLoRA works by combining low-rank adaptation (LoRA) with quantization-aware training. During fine-tuning, the weights of the LLM are quantized, typically to lower precision such as INT4, which reduces both time and memory usage. The key advantage is that after fine-tuning, the model and its auxiliary weights are fully integrated into a quantized model without requiring post-training quantization. This process helps in maintaining accuracy and efficiency during inference as well.

Firstly, we used transformers a library from Hugging Face that provides pre-trained models and tokenizers.Then QALoRAConfig, QALoRA, Classes from the QA-LoRA library that configure and apply quantization-aware low-rank adaptation. LlamaTokenizer.from_pretrained loads the tokenizer for the specified model. LlamaForSequenceClassification from_pretrained loads the pre-trained LLaMA model for sequence classification. Later, quant_bits=4 specifies 4-bit quantization for the model weights. BitsAndBytesConfig configures the model for 4-bit quantization. TrainingArguments defines various parameters for training. Hyperparameters like learning rate, batch size, and number of epochs are adjusted based on our dataset size and computing resources. After pretrained Llama2 model and adding parameters and using quantization-aware LoRA library, the data needs to be formatted for training and evaluation. 100,000 rows are taken from the filtered dataset that we created from the text corpus and 80% dataset for training and 10% for validation. For each English sentence and each French sentence pair (each row of data), the formattingpromptsfunc function formatting-matters the training and validation datasets by appending a token (tokenizer.eostoken, which indicates the end of a sentence) and prepending the instruction/prompt ”Translate from English to French:”.

After that, the SFTTrainer is used to train the model on both the training and validation datasets. Initially, we attempted to utilize the Seq2SeqTrainer with the model rather than the SFTTrainer, but we encountered several errors. There was only one epoch used for training and completing the training, takes two hours and thirty minutes. Following training, the test dataset was also structured by appending the identical prompt or instruction, ”Translate from English to French:” to each row's English sentence without also adding the dataset's equivalent French text. On the test dataset, the model may provide outputs with the French translations if the French sentences are left out.

For running our models we set up a shared Google account with Colab Pro so we could all have access to more powerful GPUs than the ones on our local machines without individually having to pay for our own accounts. This gave us the benefit of being able to place all our files in a shared location and making it much easier for group members to view and run each other's code, as well as being able to share compute units as easily mount Google Drive. There were some limitations with this approach especially as we were each working on multiple different models, as we occasionally ran into issues with multiple people trying to run several models all at once, and there was a limit as to how many we could run at a time. We would often have to coordinate and schedule who would be able to run their models when, and wait for others to finish. It was also difficult to keep programs running for an extended period of time, as they would sometimes crash in the middle of training.

3) Full fine-tuning LLaMA:

The last model that we tried to perform was LLaMA 2 with 7B parameters with A100 High RAM GPU using the same libraries for tokenization and training as for mT5. After successful processing and tokenization of data we have faced a problem during the training step - out of CUDA memory. Despite our best efforts, this problem didn't disappear even after several code optimizations to reduce memory usage. Our guess why this happened is because the model size is extremely large for the power of our GPU. This model size requires significantly more memory for full fine-tuning.

6 Evaluation scores

For evaluation, we will use 4 models from which we were able to get 200 lines of output/machine translated sentences:

- mT5 with full fine-tuning
- mT5 QLoRA
- mT5 prompt-tuning
- Llama 2 QLoRA using the Unsloth library

Additionally, among the two mT5 full fine-tuning models, one using a dataset of 100000 lines and the other one using a dataset of 200000 lines, we picked the model that has better output translations, which is the model with the dataset of 200000 lines.

In order to compare our models, we used evaluation metrics COMET score and BLEU score that were used in this paper for machine translation (Zhu et al., 2023). These metrics provide an understanding of the quality of translation and performance of each model. The BLEU score uses n-gram overlap (which checks how similar machine-generated translation are to their corresponding references) to show the quality of machine translation. The COMET score is a neural framework evaluation metric of machine translation.

For our experiment, we picked 200 sentences from the test datasets of each of the four models, which are sentences that our models didn't use for training or validation purposes.

In table 1 and figure 1 below, you can see the scores for each of the four models.

As expected, the Llama2 7b model showed better performance than the mT5 models with the highest BLUE and COMET scores. This is due to the fact that these models are much larger than the mT5 300M. As for the mT5 models that use dif-

Translation performance (BLEU/COMET)	
Model	**Scores**
mT5 fine-tuning	44.11/69.35
mT5 QLoRA	26.49/51.7
mT5 Prompt-tuning	42.39/63.89
Llama 2 QLoRA Unsloth library	52.05/76.86

Table 1: Machine translation performance of different models (BLUE/COMET)

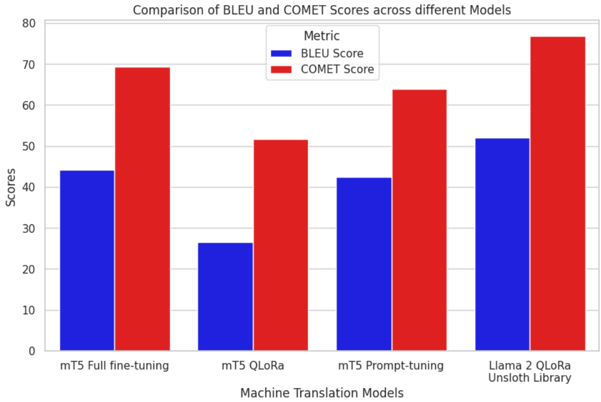

Figure 1: Machine translation performance of different models (BLEU/COMET).

ferent fine-tuning techniques, the model with full fine-tuning and prompt-tuning are the best in this category. While, mT5 QLoRA showed low results compared to other models.

For the human evaluation metrics, we compare the models' performance for each of the three metrics: Fluency, Adequacy, and Accuracy. The highest possible summed score was 90; the lowest was 30. The models compared were: an mT5 model fine-tuned on 200k lines, an mT5 model fine-tuned with QLoRA on 200k lines, a prompt-tuned mT5 model trained on 200k lines, a LLaMA model fine-tuned with QLoRA on 80k lines, and a LLaMA model fine-tuned with QLoRA on 100k lines.

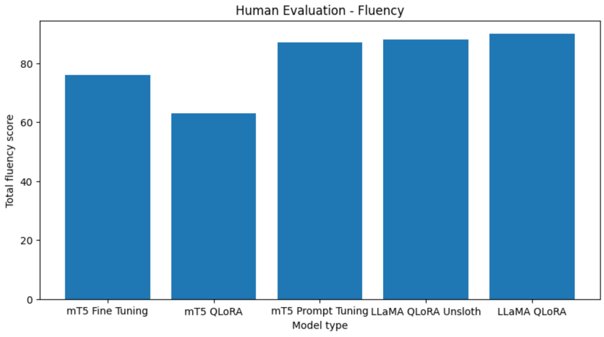

Figure 2: Sum of fluency scores for each model

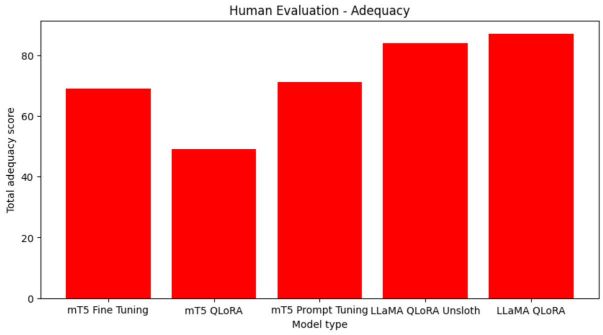

Figure 3: Sum of adequacy scores for each model

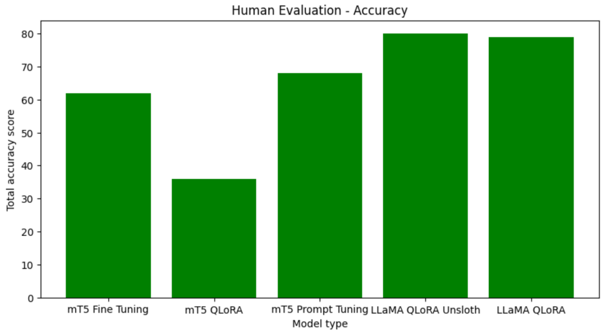

Figure 4: Sum of accuracy scores for each model

We can see from these figures that according to human evaluation the mT5 model pre-trained with QLoRA was easily the lowest-performing model in all metrics, while the LLaMA model pre-trained with QLoRA (not using the unsloth library) performed the highest in every category except accuracy, in which case the LLaMA model using the Unsloth library pre-trained with QLoRA performed slightly better. The prompt-tuned mT5 model was the only mT5 model that came close to the performance of LLaMA, suggesting that prompt-tuning is a good approach to get mT5 to produce competitive results with LLaMA. We can also see that overall the models performed the highest in fluency and the lowest in accuracy, suggesting that these machine translation models generally perform better at creating text that is human-readable but struggle with capturing the exact meaning of the original text or specific grammatical and syntactic rules.

7 Error analysis

In analyzing failure cases for each model with some help from our human evaluator, we noted common mistakes that each model would make and different types of inputs that the model struggled with. For the mT5 models, we noted that longer inputs were more likely to contain mistakes as there was more room for the model to make an error when translating.

One very common problem that we noted was that sequences of words that were either all capitalized or all began with a capital letter were frequently left untranslated. For example, in the fine-tuned mT5 model the sentence ”RIGHT TO APPEAL An appeal shall lie from the present decision.” was translated to ”RIGHT TO APPEAL Un appel doit être lié à la décision présentée.” Here you can see the first part was left in English. In translations, it is often a common practice for proper nouns and other text like titles to be left untranslated, which are often indicated with capitalized letters. However it can sometimes be ambiguous as to what should be left untranslated and what should not. It appeared that the model over-relied on this rule and left most capitalized words untranslated even in cases where it would make more sense for them to be translated, causing loss of context and information.

Another form of text that was often untranslated or sometimes omitted from the translation was acronyms. Many of the input lines from the dataset contained acronyms such as ”• Self-serve and PWGSC in collaboration with employer” which the model translated to ”• Vente et collaboration avec l'employeur”, omitting the acronym. Since acronyms can often be very technical and domain-specific, the model may lack the contextual knowledge of that domain to know how (or if) to translate the acronym and may not know what to do when it encounters one.

The fine-tuned mT5 model also struggled with handling input if it was formatted incorrectly, such in one example when the input text was not spaced correctly: ”AP HA (1995) Stand ard m ethod s for the examination o f water an d wastewater.” the model produced ”AP HA (1995) Stand ard m ethod s pour l'examen o f water an d wastewater.” Most of the text was left untranslated as the model could not correctly parse the text.

A frequently occurring problem with the translations from fine-tuned mT5 models was that they would often produce bizarre sequence with repeating words, most commonly towards the end of the sentence. The issue sometimes occurred in the fine-tuned mT5 model but was even more common in the QLoRA mT5 model which trained fewer parameters. For example the fine-tuned mT5 model took the input ”United States Department of Health and Human Services.” and produced the output ”Les États-Unis ministère des services de santé et des services de santé.” The QLoRA version had more frequent and extreme cases of repeating words in its output, such as such as with one input ”The detainees were forced to dig canals, build dams, and work in the paddies.” and then produced the output ”Les detainees sont forces à créer canals, construire dams, construire dams, construire dams, construire dams, construire dams, construire dams, construire dams.” We hypothesized that the word repetition may be attributed to beam search degeneration in decoding, where the model assigns a high probability to repeating sequences, which can cause the decoder to loop on itself. This issue appeared to be mitigated with training on more lines and more parameters. The prompt-tuned mT5 model also did not run into this issue as often, suggesting prompt tuning may be another method to resolve this.

Another issue with the output of the mT5 QLoRA model was that many outputs often began with a token masking the beginning of the output. Examples of such outputs were ”extra_id_0 est l'adoption d'un code de conducte.” and ”extra_id_0 à voir la page dedication à Ernie Collison.” The other mT5 models did not have this token appear in their outputs once they were trained on a sufficient number of lines, however this token still appeared in the output of the mT5 model fine-tuned with QLoRA, indicating that 200k lines was not sufficient for the model.

Many of the models struggled with the grammatical and semantic rules of French such as gender agreement, proper verb conjugation, and adding special accented characters. The fewer parameters a model was trained on, the more common these errors were. For instance, for the English sentence ”• High energy prices represent an increased cost component which impacts on the steel industry.” the mT5 fine-tuned model produced ”• Les prix élevés représentent une composante élevée qui impacte la industrie de l'acier.” In French, two vowel sounds are almost never placed next to each other and are often connected with a ”liaison”, so ”la industrie” should actually be ”l'industrie”. This is one of many rules that underperforming models can struggle with if they have not been trained on enough examples.

The LLaMA model pretrained with QLoRA

produced much better results, however there were occasionally still some problems with the output. One behavior of the LLaMA model was that it would not always produce literal word-for-word translations but would sometimes produce a sentence that paraphrased or captured the overall idea of the original sentence, but without using the exact wording. This can be a valid approach for translation but can sometimes also cause loss of context or information. One example was for the input sentence ”Dr. Ron wants you – if you have the right stuff.” the LLaMA model output ”Dr Ron est à la recherche de personnes qui ont le gout de la médecine.” Which in English literally translates to ”Dr. Ron is looking for people who have a taste for medicine.” (Note the model also had a minor spelling error - ”gout” should contain the circumflex character: goût). This is similar to the idea of the original sentence but not exactly the same and may cause some miscommunication. One reason for this may be that common English slang terms like ”have the right stuff” may not have exact translations in other languages like French, and may cause confusion if translated literally. The model may not know how to translate this phrase and will instead use a similar but different phrase. The mT5 prompt-tuned model has a similar problem- the model would sometimes ”hallucinate” and add extra information or text that was not present in the original text, often changing the meaning of the sentence and creating a mistranslation. We were unsure why this was; it could possibly be attributed to the dataset containing incorrect pairs or an issue with decoding.

The models also sometimes lacked cultural knowledge needed to create accurate translations. For example, for the sentence ”Some colleges also offer summer sessions.” the prompt-tuned mT5 model generated ”Certains collèges offrent également des sessions d’été.” This may look correct on the surface, however in France ”collège” does not refer to higher education like ”college” in English, but is actually middle/high school for students between the ages of 11-15. A more correct translation here would be ”université”. If the model does not have details about French culture such as the French education system, it may struggle with translations like these.

There were many other types of issues that we noticed; the full error analysis can be found in ”error analysis.xlsx” under the ”Evaluation files” folder on GitHub. The human evaluator feedback files can also be found in this folder.

8 Contributions of group members

- Aylin Elmali: did data preprocessing, contacted human evaluator and wrote evaluation guidelines, setup shared Colab account, built and trained MT5 model with QLoRA, error analysis, wrote for sections 3, 4 and 7
- Sapna Parihar: scheduled weekly meetings with groupmates to work on final project, built, trained, and tested QLoRa Llama model using Unsloth library, wrote sections 1, 2, part of 3, and part of 5 of final report
- Deepa Rukmini Mahalingappa: Data preprocessing, built and Fine-tuning LLAMA with Quantization-aware LoRA Library, contributed for final report writing in section 3 and 5.
- Sarah Boal: Built and trained fine-tuned mT5-small model for Prompt Tuning with BLEU and COMET evaluations, attended office hours with Professor and TA for overall project guidance and to answer teammates’ questions, writing final report contributing to section 2, 5, and 9.
- Albert Galimov: Build and fine-tuned mT5 models with 100k lines and 200k lines. Collection a scores from each model and plotting a graph and table to compare BLEU scores and COMET scores. Tried to build Llama 2 full fine-tuning, wrote part of 5 and 6 of report.

9 Conclusion

Our first takeaway was that we learned that mT5 was only pretrained without any supervised training, therefore it needed to be fine-tuned before using it on a downstream task unlike T5. We had initially tried using T5 which worked easily when testing zero-shot prompting, however this was not the case for mT5. This is how we knew that it needed to be fine-tuned in order to obtain any meaningful output or translation.

It was surprisingly difficult to fine-tune Llama 2 because of the size of the model, and amount of CUDA memory required to fine-tune this model for our task. This demonstrated to us the importance of fine-tuning with QLoRa for large models in order to be able to run our model without running out of memory. However for mT5 which had a smaller number of parameters, QLoRA did not appear to provide much benefit and actually hindered the performance of the model. Additionally, we did not realize how long it would take to train our models, and how much GPU usage would be required.

This was good experience for us to better understand the capabilities required to fine tune large language models, in this case for English to French translation. The results for Llama 2 fined tuned with QLoRa Unsloth were excellent with BLEU scores in the 50 range, this was surprising at first but then when we considered that Llama 2 has 7B parameters as compared to the much smaller number of parameters in mT5, this provides a reasonable explanation. We could potentially consider trying out the Llama unsloth paid versions and see how they perform with the metrics of BLEU and COMET.

The model mT5 was more difficult than anticipated to achieve a decent translation result, it requires a lot more training data because it didn't have any supervised training. If we continue working on our project in the future, we could try working with the larger models of mT5 and Llama 2, as well as experiment with Llama 3. Also we would want to try training on even larger datasets and better quality datasets, our original dataset from Kaggle had some flaws and were 22 million sentences. To train a dataset of this size we would need significantly more compute than what is available on Google Colab or on a Single GPU. We would most likely need to utilize multiple GPUs and a Cloud resource such as Microsoft Azure, AWS, or Google Cloud.

10 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - No, we didn't

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - N/A
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  - N/A

References

Aharoni, R., Johnson, M., and Firat, O. (2019). Massively multilingual neural machine translation. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., and Macherey, W. (2021). Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Han-Chen, D. (2024). Make llm fine-tuning 2x faster with unsloth and trl.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In Isabelle, P., Charniak, E., and Lin, D., editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Park, D. and Padó, S. (2024). Multi-dimensional machine translation evaluation: Model evaluation and resource for korean.

Rei, R., Chau, A., Linares, J., Stymne, S., Wang, S., Singh, V., Cho, K., Birch, A., and Weston, J. (2020). COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Touvron, H., Alayrac, J.-B., Douze, M., Strub, F., Cord, M., and Jaggi, M. (2023). LLaMA: Open and efficient foundation language models. *arXiv:2302.13971 [cs]*.

Wieting, J., Berg-Kirkpatrick, T., Gimpel, K., and Neubig, G. (2019). Beyond BLEU: Training neural machine translation with semantic similarity. In Korhonen, A., Traum, D., and Màrquez, L., editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4344–4355, Florence, Italy. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mt5: A massively multilingual pre-trained text-to-text transformer.

Zhu, W., Liu, H., Dong, Q., Xu, J., Huang, S., Kong, L., Chen, J., and Li, L. (2023). Multilingual machine translation with large language models: Empirical results and analysis.