

ADL FINAL REPORT - E-MoTChi

Emoji-Mandarin Translation via Computational Homophonic Intelligence

B10902032 李沛宸、B10902033 林祐辰、B10902034 廖奕鈞

B10902041 連奕維、B10902117 范秉逸

Github-link: <https://github.com/MarkymarkLee/ADL-Final-Project>

ABSTRACT

In this work, we introduce a model to deal with a new sort of translation, bi-directional conversion of traditional chinese and emojis, utilizing advanced language models including Taiwan-Llm-2 with Query-based Low-Rank Adaptation (QLoRA) method and fine-tuned massively multilingual pre-trained text-to-text transformer(mt5-large). We trained models using a diverse dataset, ingeniously blending a precise one-to-one dictionary algorithm with creative GPT-3.5 generations, and achieved the goal of offering a novel way, fine-tuned model E-MoTChi, to generate translations.

1. INTRODUCTION

In the realm of social media, emojis become a creative tool for crafting unique and engaging conversations with friends. Rather than straightforwardly translating meanings, a fascinating practice emerges in which Chinese characters leverage homophones to transform sentences into expressive emojis. This method introduces an element of playfulness, allowing both the sender and receiver to share a linguistic shorthand. For instance, the combination “🐦🐼” might signify the Chinese characters “確實”, while “🌀🧵🍵” could cleverly represent “真羨慕”. This linguistic exchange adds a layer of fun and connection, creating a shared understanding for those who happen to share the same "brain cells" in decoding these symbol-laden messages.

This work aims to create a unique form of digital communication mentioned above, blending traditional Chinese with the expressiveness of emojis by first creating an innovative algorithm that translates chinese to emojis character by character and then training both a transformer model and a LLM model with the combined dataset generated by our algorithm and GPT.

During this process, we also discovered a research problem of how a LLM interacts with 2 slightly different tasks.

1.1 Inspiration & Observation

Inspired by the creative usage of emojis on social media, particularly on platforms like YouTube and TikTok, we were motivated to explore this concept further. However, after experimenting on GPT-4, we found that some information in a sentence will be ignored, and it generally directly translates without using homophones.

1.2 Significance

The project is significant as it offers a novel and engaging way to communicate. It transforms standard conversations into an interactive game of interpreting emojis, enhancing digital interactions with a blend of fun and insight. It also provides an opportunity to explore the distinct training processes of transformers and large language models (LLMs), as well as to evaluate their unique abilities in tackling tasks they have never encountered before. Moreover, this task introduces a new problem for LLMs that can be researched further.

2. METHOD

2.1 Step-by-Step Project Overview

The project starts with **(1)Dataset Collection:** Initiate the project by gathering a comprehensive dataset of Chinese sentences. This followed by the **(2)Initial Emoji Translation:** Develop an emoji translation of the collected Chinese sentences, based on our understanding and interpretation of their meanings. In the next phase, **(3)Translation Using GPT-4:** Employ GPT-4 to translate the Chinese sentences into

emojis, focusing on capturing the essence and meaning of each single sentence. Similarly, we set **(4)Character-by-Character Translation Algorithm:** Design an algorithm to translate each character of the Chinese sentences into their corresponding emojis. With two dataset from (3) and (4), We **(5)Integrate Dataset:** Combine the two datasets, resulting in a mixed dataset containing both emoji and Chinese sentences, mixing the properties of two datasets.

Next, we start to train two models. First, we **(6)Implement Model Training on Taiwan Llama:** Train both directions of tasks respectively on the Taiwan Llama model using the mixed dataset. Experiment with various training arguments to identify the most effective ones. Further, We **(7)Train on Varied Datasets:** Repeat the training process with the identified arguments, but apply them to different datasets to test their effectiveness. Following the training stage, We conduct **(8)Model Evaluation:** Assess the performance of all trained models to determine their accuracy and efficiency. To compare with the first model, we develop **(9)Adaptation to Google MT5-Large:** Train and evaluate Google MT5-Large model. Finally, **(10)Model Comparison and Selection:** Evaluate the performance of both Taiwan Llama and MT5-Large models. Select the final model based on comparative analysis.

2.2 Learning Settings:

2.2.1 Models:

(1)Taiwan Llama v2 and **(2)Google mT5 Large**

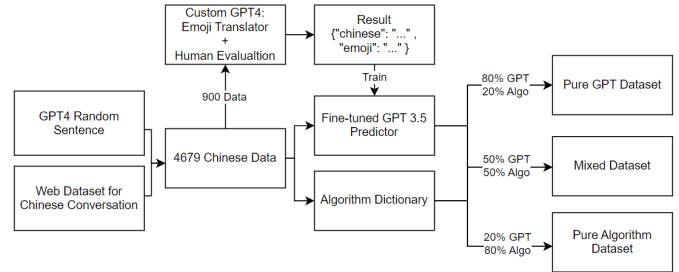
2.2.2 Evaluation Metrics:

(1)METEOR and **(2)Human Evaluation** gives total scores on understandability, and adaptability (How well it uses the knowledge in the dataset to generate.

2.3 Data Generation

2.3.1 Selection of Sentences for Translation

We selected conversation datasets and random



GPT generated sentences for the training dataset.

2.3.2 Dataset Generation Methods

Figure1: workflow of dataset generation

2.3.2.1 GPT

We utilize GPT-4 to create an initial set of sample data. This process involves translating sentences into emojis, ensuring that the emojis accurately reflect the core meaning of each sentence. Subsequently, this initial dataset, generated by GPT-4, is used to further train GPT-3.5-turbo. The purpose of this training is to enable GPT-3.5-turbo to expand the dataset by producing more translations. Through human evaluation, it has been observed that the GPT-generated dataset tends to favor translations where emojis correspond closely to the semantic content of the sentences.

2.3.2.2 Algorithm

We sourced critical data from a Chinese website (zidian.18dao.net), where we scraped two types of information: mappings from each Chinese character to a corresponding emoji, and a list of homophonic Chinese characters. This data was then subjected to human evaluation for preprocessing, ensuring its suitability and accuracy for our purposes. Following this, we developed an algorithm to create a specialized dataset. The full algorithm pseudocode is shown in [Appendix A](#).

2.4 Model selection and training

2.4.1 Taiwan Llama2

We trained the Taiwan Llama with Axolotl.

2.4.1.1 Training Args

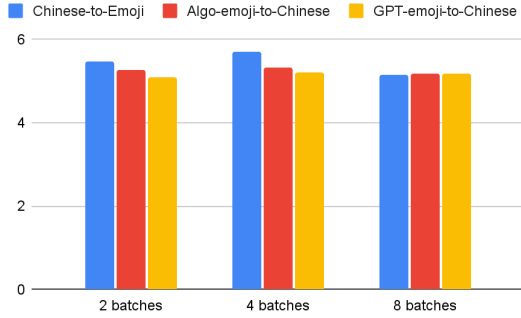


Figure 2: Human Evaluation on Llama model

In our experiments, we tested batch sizes of 2, 4, and 8, each paired with 8, 16, and 24 epochs to evaluate their impact on model performance. We chose these sizes based on promising results from smaller tests with a batch size of 4. The best training arguments are outlined in [Appendix B](#).

2.4.1.2 Score

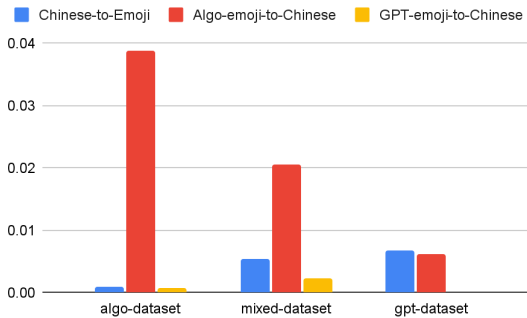


Figure 3: METEOR Score on Llama prediction with different dataset models

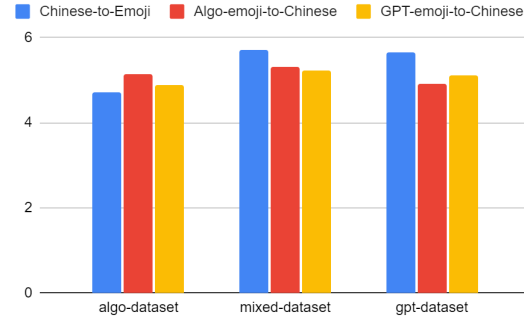


Figure 4: Human Evaluation on Llama model trained with different datasets

Three models were trained using different datasets: Algo-dataset, Mixed-dataset, and GPT-dataset, with dataset ratios of 80-20, 50-50, and 20-80, respectively. Classifying the model, Chinese-to-Emoji, Algo-emoji-to-Chinese, GPT-emoji-to-Chinese, depends on the input dataset used for prediction. Despite the algo-dataset showing a higher METEOR score visually than the mixed-dataset, The human evaluation score shows that training with the mixed dataset is better than with the other 2 datasets.

2.4.2 Google-MT5-Large

2.4.2.1 Training Args

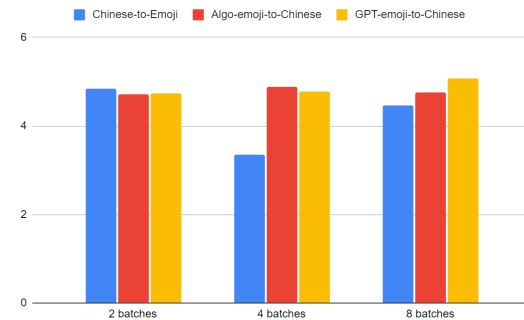


Figure 6: Human Evaluation on mT5 models trained with different batch sizes

Similarly, we explored batch sizes of 2, 4, and 8 with 5, 10, and 15 epochs respectively. Even though the model seems to perform better on batch size 2 and 8, batch size 4 is still chosen to compare the results with Llama. The more complete training arguments will be shown in [Appendix C](#).

However, a problem occurred when training on the task Chinese-to-Emoji and will be discussed more in [Section 5.2.3](#).

2.4.2.2 Score

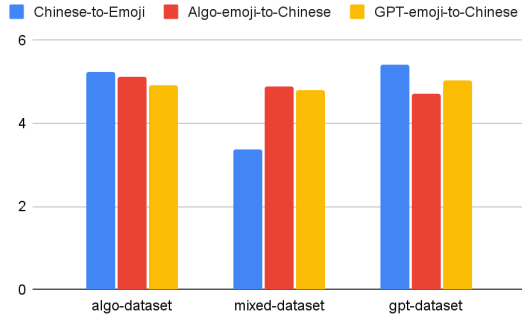


Figure 7: Human Evaluation score on mT5 models trained with different dataset

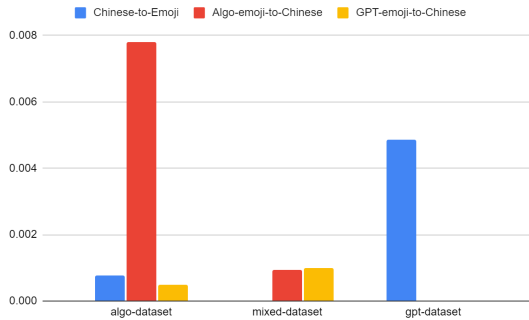


Figure 8: METEOR score on mT5 models trained with different dataset

From Figure 7, we can slightly learn that the task Algo-to-Emoji performs better if more Algorithm data are in the training dataset. However, the task GPT-to-Emoji may not perform better even if more GPT data are in the training dataset for mT5, which is an interesting discovery.

2.4.3 Final Decision

Based on both the meteor metric and the human evaluations, the Taiwan Llama model trained on the mixed dataset with batch size 4 and 16 epochs outperforms every other model, and thus it will be the final model that we decide to use.

3. Experimental Details

3.1 Sentence Selection Process

E-MoTChi is determined to primarily focus on processing conversation-based content. To facilitate this, we sourced an English conversation dataset from Kaggle, and subsequently translated it using Google's Translation API. Additionally, we employed GPT-4 to generate a variety of random sentences, aiming to enhance the model's adaptability to more diverse and general datasets. The combined sentence dataset is all the sentences that we want to translate.

3.2 Reasons for Selecting GPT as Initial Translators

We selected GPT as the initial translator for its ability to understand both Chinese and emojis effectively. GPT excels at providing straightforward translations and can creatively interpret Chinese into emojis, particularly when direct translations are not feasible. This results in translations that are generally more readable than those produced by our algorithm. However, GPT's limitations include occasionally overcomplicating meanings, leading to awkward emoji grammar, and a lack of understanding of homophonic Chinese characters in relation to emojis. These considerations were crucial in our decision-making process.

3.3 Reasons for Selecting Our Algorithm as Initial Translators

Our custom algorithm is chosen for its thoroughness in translating every Chinese character to an emoji, ensuring no loss of original meaning, and its proficiency in handling homophones. However, it falls short in understanding the varied meanings a Chinese word might have in different contexts, often using the same emoji repetitively. This approach can lead to less intuitive translations compared to the context-sensitive results from GPT-4, making them harder for humans to comprehend.

3.4 Selection and Adaptation of Models for Translation

Our initial choice was **(1)Taiwan Llama v2**, selected for its adeptness in understanding Traditional Chinese and a broad array of emojis. To train this large language model (LLM), we employed Low-Rank Adaptation (LoRA), a technique tailored for efficient training of expansive models.

However, early experiments with Taiwan Llama v2 indicated limitations in learning from our dataset, often resulting in outputs suggesting a lack of understanding. This prompted us to explore methods to enhance performance. One approach was to increase the LoRA rank, 'r'. Remarkably, at 'r = 64', the model began to effectively learn from the dataset, leading to the improved performance we observed with Taiwan Llama v2.

Simultaneously, we considered an alternative strategy involving a different pre-trained model. We chose **(2)Google mT5 large** for its multilingual capabilities and encoder-decoder architecture. Our hypothesis was that mT5, being a transformer model with less inherent pre-knowledge, would be more adept at quickly assimilating the training dataset. This decision to experiment with multiple models reflects our commitment to finding the most effective translation tool, taking into account both linguistic understanding and learning efficiency.

3.5 Selection of Evaluation Metrics for Model Assessment

Initially, we planned to use Perplexity Scores for training evaluation and ROUGE scores for generation strategy. However, given the mT5 model's incompatibility with perplexity scoring, we opted to standardize our generation strategy across all models. Consequently, we chose the METEOR score as our primary evaluation metric, due to its comprehensive approach in assessing translation quality, which includes factors like accuracy, word order, and synonym

use. This metric aligns more closely with human judgment in translation quality assessment.

Despite selecting the METEOR score for its comprehensive approach to assessing translation quality, we recognized that no existing evaluation metric perfectly aligns with the unique demands of our new task. Given this, while we will utilize the METEOR score as a part of our evaluation process, our primary focus will be on human evaluation. This approach ensures a more nuanced and context-sensitive assessment, crucial for the innovative nature of our translation task.

3.6 More on Human Evaluation

We randomly checked 5 data points for each model. And for each data point each person gives between 0 to 10. We then calculate the standard deviation score based on each task for each person.

The pros for this implementation is that it removes most bias since we show each data point randomly and calculate the per task standard deviation.



4. Result

4.1 Final model











After the above evaluation, we eventually choose Llama-2 with mix datasets and batch size 4 as our final model.

4.2 Demo

We have experimented with some instructions, conducting multiple trials for each. On task "Chinese to Emoji" with the instruction "他們沒有說", E-MoTChi generated the outputs "👤 🏠 🎩 有 🗣️。" and "👤👤👤👤👤👤👤👤👤👤", This serves as compelling evidence of its capability to perform translations in both algorithmic and GPT styles. On the other hand, on task "Emoji to Chinese", given the instruction "🐦 🍌", the generated output is "當然", and with instruction "🍌 🍌 🍌", E-MoTChi outputs "一個果實". It's obvious that E-MoTChi can understand homophonic and is able to translate it, but we are

wondering how it come up with “當然” when seeing “ ” even though “確實” and “當然” are synonyms.\

4.3 Other

There are some unexpected performances in our experiment. When we input “白雪公主”, E-MoTChi outputs “  ”. This is beyond our expectation since GPT-3.5 would output “  ”, algorithm would output “  ”, and Taiwan Llama would output “”. The output of E-MoTChi seems to be a brand new way of translation.

5. Discussion

5.1 Final Results

batch size - performance relation (how changing the batch size change the model’s performance)

dataset - performance relation (how changing the datasets change the model’s performance)

More observations on how the model performs the 2 tasks separately instead of learning the techniques.

5.2 Difficulties and Reflections

5.2.1 Dataset Generation

We believe that data generation is the most challenging part in our project since It’s a brand new idea of combining chinese, homophonic chinese characters, and emojis. Therefore, there aren’t any existing datasets, and we have to create them by ourselves.

5.2.1.1 Improvements

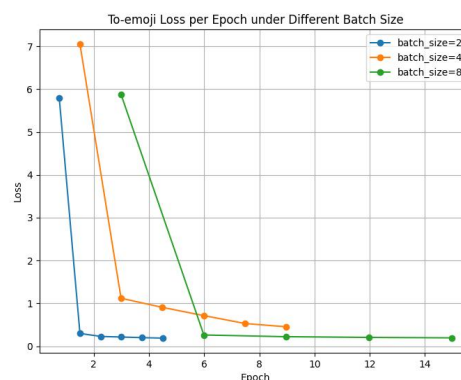
The first thing that could have been done better is scraping more emoji data that is actually used in real world conversations. The second thing that could be done better is adjusting the algorithm. An example for this is to implement sentence segmentation and translate each segment based on both meanings and homophones. Another example is instead of mixing the 2 datasets into one dataset, we can translate some parts of a sentence with GPT and the others with our algorithm. However, we haven’t thought of an

effective way to implement the 2 examples without sacrificing the original algorithm’s ability.

5.2.2 Model Selection

Initially, we wanted to compare the performance on this task between different LLMs. The problem with this is finding an effective model with traditional Chinese support. For example, we wanted to test the newest Mistral model, but only simplified Chinese supported versions were found, and the model performs poorly even if we did translate the inputs and outputs to simplified Chinese before training it. Another attempt was made with the GPT3.5 model. However, fine-tuning cost a lot of money and exceeded our team’s budget. Therefore, we settled with the Taiwan Llama v2 13B model and the Google mT5 model.

5.2.3 Training



A problem occurred during training, and that is when we train the mT5 model on the mix dataset with batch size 4 and 10 epochs. Unlike with batch size 2 or 8, the model didn’t converge even after 10 epochs. As shown in the figure. Another problem that we were met with is overfitting. As shown in the figure above, most of our models overfit and thus might not have the best performance. An improvement that could have been made is to observe the training loss during training instead of training a lot of models at once.

5.3 Future Expectations

First, we plan to test additional models and refine our algorithm for better task alignment. Second, we investigate the impact of a dataset combining Chinese and emojis within sentences to enhance the model's grammatical comprehension of emojis. Third, leveraging our existing datasets for homophones and emojis, we'll direct the model to apply this knowledge to specific tasks. Fourth, we need to explore how the model differentiates between datasets and tasks, as it currently tends to use two methods separately rather than integrating them in a single sentence. Finally, although our group has substantial computational resources, including four A100 GPUs, and isn't constrained by memory limits, we aim to reduce our models' memory requirements to increase their accessibility.

6. Reference

Data Generation:

<https://www.kaggle.com/datasets/kreeshrajani/3k-conversations-dataset-for-chatbot>
<https://chat.openai.com/>
<https://www.emojiall.com/zh-hant>
<https://zidian.18dao.net/>

Models:

<https://huggingface.co/yentinglin/Taiwan-LLM-13B-v2.0-chat>
<https://huggingface.co/google/mt5-large>
<https://huggingface.co/spaces/evaluate-metric/me-teor>

Trainers:

<https://github.com/OpenAccess-AI-Collective/axolotl>

7. Appendix

7.1 Appendix A

`char_to_emoji`: a map from character to its corresponding emoji

```
def set_weighted_emoji(char_to_emoji):
    create a map for each character to its frequency in all sentences
    for each char:
        weight = {}
        get its homophonic characters
        for each homophonic character h:
            if h is in char_to_emoji:
                weight[char_to_emoji[h]] += h's frequency
        set char's weighted score for emoji e to sqrt(weight[e])

def translate_to_emoji(sentence):
    for each character c in sentence:
        if c is not in char_to_emoji:
            do nothing
        elif 50 percent chance:
            replace c with char_to_emoji[character]
        else:
```

```
        retrieve c's weighted score for each emoji
        replace c with random weighted chosen emoji
    return sentence
```

7.2 Appendix B

```
base_model: Taiwan-LLM-13B-v2.0-chat
model_type: LlamaForCausalLM
tokenizer_type: LlamaTokenizer
is_llama_derived_model: true
load_in_4bit: true
datasets:
  - path: data/c2e_train_mix.json
    type: alpaca
dataset_prepared_path:
val_set_size: 0
output_dir: model/c2e/mix_4
adapter: qlora
sequence_len: 2048
sample_packing: true
pad_to_sequence_len: true
lora_r: 64
lora_alpha: 16
lora_dropout: 0.05
lora_target_modules:
lora_target_linear: true
lora_fan_in_fan_out:
gradient_accumulation_steps: 1
micro_batch_size: 4
num_epochs: 16
optimizer: paged_adamw_32bit
lr_scheduler: cosine
learning_rate: 0.0002
bf16: true
fp16: false
tf32: false
```

7.3 Appendix C

```
{
  "_name_or_path": "google/mt5-large",
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "d_ff": 2816,
  "d_kv": 64,
  "d_model": 1024,
```



```
"decoder_start_token_id": 0,  
"dense_act_fn": "gelu_new",  
"dropout_rate": 0.1,  
"eos_token_id": 1,  
"feed_forward_proj": "gated-gelu",  
"initializer_factor": 1.0,  
"is_encoder_decoder": true,  
"is_gated_act": true,  
"layer_norm_epsilon": 1e-06,  
"model_type": "mt5",  
"num_decoder_layers": 24,  
"num_heads": 16,  
"num_layers": 24,  
"output_past": true,  
"pad_token_id": 0,  
"relative_attention_max_distance": 128,  
"relative_attention_num_buckets": 32,  
"tie_word_embeddings": false,  
"tokenizer_class": "T5Tokenizer",  
"torch_dtype": "float32",  
"transformers_version": "4.34.1",  
"use_cache": true,  
"vocab_size": 250112  
}
```