UNIVERSITI MALAYA

WID3002 NATURAL LANGUAGE PROCESSING

GROUP ASSIGNMENT REPORT

Simple Malaysian Tweet Normalization System

SEMESTER 2, SESSION 2024/2025

Guided by: Dr. Mohamed N.M. Lubani

Prepared by: Group 5

No	Name	Matric No.
1	Ryan Chin Jian Hwa	23005233
2	Ang Li Jia	23005237
3	Chong Jia Ying	23005232
4	Chua Hui Ying Nicole	23005225

1. INTRODUCTION

X (formerly Twitter) is widely used in Malaysia, where tweets often contain slang, abbreviations and a mix of English and Malay. These noisy, informal and unstructured texts are hard to understand for outsiders and difficult for standard language models to process. There is also a lack of tools supporting this unique language mix. This project aims to build a simple system to clean and normalize Malaysian tweets into proper English or Malay.

2. SOLUTION ARCHITECTURE

2.1 Solution Design

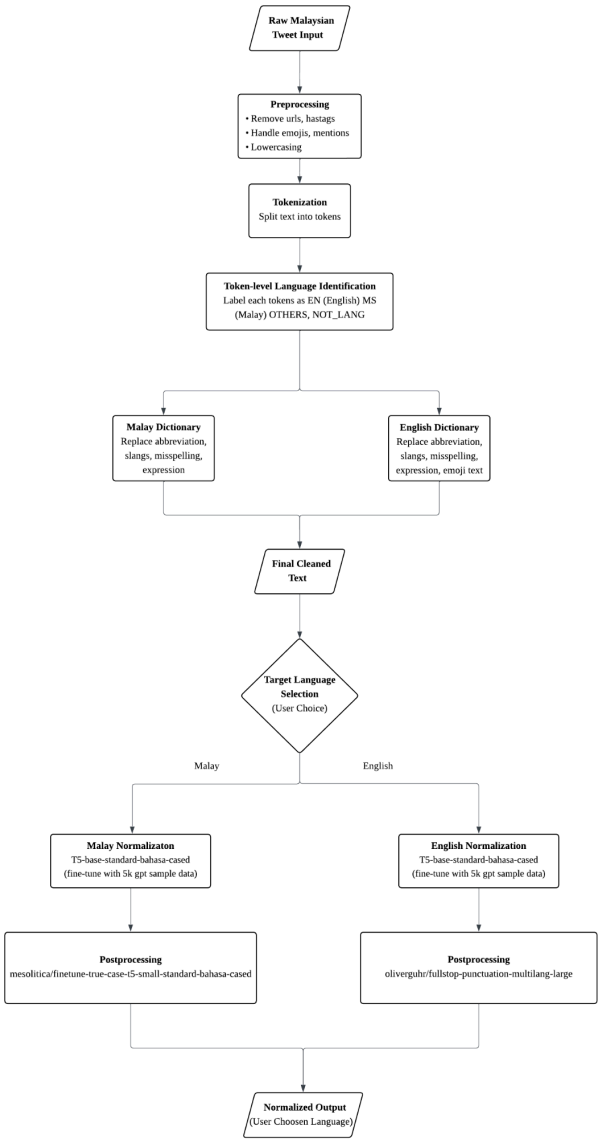

Figure 1: Simple Malaysian Tweet Normalization System Pipeline

To handle the challenge of noisy, mixed-language tweets, this project proposes a normalization pipeline that transforms informal Malaysian tweets into clean, structured text. The solution begins with raw tweet input collected from the platform X (formerly Twitter). The first step is preprocessing, where the system removes URLs, hashtags, emojis and user mentions then it converts the text to lowercase to maintain consistency.

Next is the tokenization stage, where the tweet is broken into smaller parts called tokens. These tokens are passed into a token-level language identification module, which labels each token as English (EN), Malay (MS), Others or Not Language.

Once the language of each token is identified, the system uses two separate dictionaries, one for English and one for Malay. The previous labels help the system choose the correct dictionary for replacing slang, abbreviations, misspelled words and common expressions with their correct or formal versions.

After cleaning the tokens using the dictionaries, the system produces the final cleaned text. At this stage, the user selects the target language they want the tweet to be normalized into (either English or Malay). Based on the user's choice, the cleaned text is passed through a normalization model, a T5-based language model from Mesolitica that has been fine-tuned with 5,000 Malaysian tweet samples to understand local language patterns.

Finally, a postprocessing step ensures the output text has proper punctuation and casing. For Malay, a model from Mesolitica is used, while for English, a multilingual punctuation model by Oliverguhr is applied. The end result is a normalized tweet, cleaned and written in either formal English or Malay.

2.2 Datasets

The dataset used in this project is the ChatGPT 3.5 Noisy Translation Twitter dataset, sourced from *Mesolitica / malaysian-dataset / translation / chatgpt3.5-twitter*. It contains approximately 691,000 cleaned and processed entries, collected from informal Malaysian tweets posted between 2021 and 2022. Each entry in the dataset is stored in JSONL format, where the "left" field contains the original noisy Malay tweet, while "en" and "ms" hold the corresponding high quality translations in English and standard Malay respectively that were generated using ChatGPT 3.5. This dataset was specifically chosen because it focuses on real-world Malaysian language. Moreover, the dual-language translations support multilingual NLP tasks and the large size of the dataset makes it an excellent resource for both training and evaluation.
The translations provided by ChatGPT 3.5 will serve as our reference for fine-tuning and evaluation of our model. This is to support our approach of **model distillation**, since ChatGPT 3.5 is a reliable LLM.

left	en	ms
@mazwinnikanis Dalam hujan lebat some more	In heavy rain some more	Dalam hujan lebat lagi
@is_pelssy @TitiRusdi Wuuiihhh.. Ketemu dimana tuu? Salam ya	@is_pelssy @TitiRusdi Wuuiihhh.. Where did you meet? Say hi for me	@is_pelssy @TitiRusdi Wuuiihhh.. Di mana kamu bertemu? Sampaikan salam untuk saya

Figure 2: Sample Entry from the ChatGPT 3.5 Noisy Translation Twitter Dataset

2.3 Tools

This project uses several tools to support development and experimentation. Visual Studio Code is the main code editor used for writing and managing scripts. The Hugging Face Transformers library is used to load and fine-tune pre-trained language models for normalization tasks. Additionally, Gradio is used to build a simple web interface for testing the system and demonstrating its output.

2.4 Models

For our use case — translation and converting informal text into formal form — T5 is suitable. We chose mesolitica/t5-base-standard-bahasa-cased, a pretrained T5 model available on Hugging Face (mesolitica/t5-base-standard-bahasa-cased). This model is based on Google's T5 (Text-to-Text Transfer Transformer) architecture, which converts all tasks into a text-to-text format. It performs well on sentence-level translation and transformation, making it an appropriate choice for our application.

This is a model which is pretrained on language masking task on bahasa news, bahasa Wikipedia, bahasa Academia.edu, bahasa parliament, EN-MS translation, MS-EN translation, abstractive summarization, knowledge graph triples generation, paraphrase and others. This pretraining coverage gives the model a strong grasp of both formal Malay grammar and common sentence structures, allowing it to generalize well to informal input and generate high-quality output.

T5's bidirectional encoder ensures it fully understands the context of each token in both directions, and its autoregressive decoder makes it capable of generating high-quality, fluent text—something essential when translating or normalizing informal sentences. Furthermore, T5 is highly customizable. Unlike GPT models that rely heavily on prompt engineering, T5 supports full fine-tuning for downstream tasks. This allowed us to adapt the model directly to the structure and quirks of Malaysian tweets through supervised training.

2.5 Training Procedures

2.5.1 Pre-processing

```
tokenizer = TweetTokenizer()
tokenized_tweet = tokenizer.tokenize(text.replace("'", "'").replace("…", "..."))
```

Figure 3: Using TweetTokenizer from TLTK

```
def cleanToken(token):
    lowercase = token.lower()
    if token.startswith("@"):
        return ""
    elif lowercase.startswith("http") or lowercase.startswith("www"):
        return ""
    elif len(token) == 1:
        return demojize(token)
    else:
        return token
```

Figure 4: Clean token

```
normTweet = (
    normTweet.replace("cannot ", "can not ")
    .replace("n't ", " n't ")
    .replace("n 't ", " n't ")
    .replace("ca n't", "can't")
    .replace("ai n't", "ain't")
)
normTweet = (
    normTweet.replace("'m ", " 'm ")
    .replace("'re ", " 're ")
    .replace("'s ", " 's ")
    .replace("'ll ", " 'll ")
    .replace("'d ", " 'd ")
    .replace("'ve ", " 've ")
)
normTweet = (
    normTweet.replace(" p . m .", "  p.m.")
    .replace(" p . m ", " p.m ")
    .replace(" a . m .", " a.m.")
    .replace(" a . m ", " a.m ")
)

return normTweet
```

Figure 5: Token normalization

1. We started off with tokenization using TweetTokenizer from NLTK.
2. We then clean the token.
   - ➔ lowercase the token
   - ➔ remove @, http and www
   - ➔ remove the one character word (emoji)
3. We then normalize the common contractions and time format.
4. We used FastText to identify the language of each word. Each word was labeled as:
   - ➔ 'EN': English
   - ➔ 'MS': Malay
   - ➔ 'OTHERS': Other languages
   - ➔ 'CAPITAL': All capitalized words
   - ➔ 'NOT_LANG': Symbols, numbers, or unknown tokens
5. Using the language tags from the previous step, we applied dictionary replacements to handle:
   - ➔ Abbreviations: Replaced by their full forms
   - ➔ Slang: Standardized to proper words
   - ➔ Misspellings: Corrected to their proper spelling
   - ➔ Expressions: Transformed into their canonical form

We maintained separate dictionaries for English and Malay, consisting of abbreviations, slangs, misspellings and expressions.

2.5.2 Fine Tuning

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

Figure 6: Using Trainer = Seq2SeqTrainer

```
training_args = Seq2SeqTrainingArguments(
    output_dir=output_dir,
    eval_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    num_train_epochs=5, # Increased epochs slightly for smaller dataset
    weight_decay=0.01,
    save_strategy="epoch",
    fp16=torch.cuda.is_available(),
    load_best_model_at_end=True,
    metric_for_best_model="eval_bleu",
    greater_is_better=True,
    report_to="tensorboard",
    gradient_accumulation_steps=2,
    push_to_hub=False,
    predict_with_generate = True
)
```

Figure 7: Setting up training_args

1. We first split the dataset. The sample size is 5000 and testing 10% and validation 90%.
2. For fine tuning, we set up training_args
   a. learning_rate=2e-5
   b. num_train_epochs=5
   c. per_device_train_batch_size=4
   d. fp16=torch.cuda.is_available()
   e. metric_for_best_model="eval_bleu"
   f. gradient_accumulation_steps=2
3. Then, we tokenize dataset
4. Use Trainer = Seq2SeqTrainer to train

Note: Sample size of 5000 was used due to limitations in resources for fine-tuning.

2.5.3 Post-processing

```python
def postprocess_ms(text):

    # Remove ALL @USER and HTTPURL patterns
    text = re.sub(r'HTTPURL', '', text)
    text = re.sub(r'@USER[.,;:!?]*', '', text)

    # Clean extra spaces
    text = re.sub(r'\s+', ' ', text).strip()

    if not text:
        return text

    # Remove excessive punctuation
    text = re.sub(r'([.!?])\1{2,}', r'\1', text)
    text = re.sub(r'\.{2,}', '.', text)
    text = re.sub(r'\?{2,}', '?', text)
    text = re.sub(r'!{2,}', '!', text)

    # Add punctuation
    inputs = tokenizer_ms(text, return_tensors="pt", truncation=True)
    with torch.no_grad():
        output_ids = model_ms.generate(**inputs, max_length=512)
    output = tokenizer_ms.decode(output_ids[0], skip_special_tokens=True)
```

Figure 8: Clean the text

```python
    # Clean spacing around punctuation
    text = re.sub(r'\s+([.,!?])', r'\1', text)
    text = re.sub(r'\s+', ' ', text).strip()

    # Capitalize first letter of sentences
    text = re.sub(r'(^|[.!?]\s+)([a-z])', lambda m: m.group(1) + m.group(2).upper(), text)

    # Ending punctuation
    if text:
        text = re.sub(r'[^\w\s.!?]*$', '', text)
        text = re.sub(r'([.!?])[.!?]*$', r'\1', text)

        if not text.endswith(('.', '!', '?')):
            text += '.'

    return text
```

Figure 9 : Punctuation

1. Clean Text.
   Remove @USER, HTTPURL, and unwanted tokens
   Fix extra spaces and repeated punctuation
   Ensure sentence ends with proper punctuation
2. Capitalization & Casing
   Capitalize first letters after [.], [!], [?]
   Apply true casing using: mesolitica/finetune-true-case-t5-small-standard-bahasa-cased
3. Punctuation Restoration
   Malay: mesolitica/finetune-true-case-t5-small-standard-bahasa-cased
   English: oliverguhr/fullstop-punctuation-multilang-large

3. RESULTS

3.1 Comparative Analysis

Model	T5	BERTweet	GPT
Model Type	Encoder-Decoder	Encoder only	Decoder only
Directional	Bidirectional encoder + autoregressive decoder	Bidirection	Unidirectional (left-to-right)
Fine-tuning	- Supports full fine-tuning for custom tasks - Can be trained on summarization, translation, etc.	- Task-specific layer added on top of the pre-trained BERT model	- Providing task-specific prompts using few-shot or one-shot adaptation and adapting the model's parameter
Use Case	Text-to-text tasks: - Translation - Summarization - Question answering - Formalization of informal text	- Sentiment analysis - Named entity recognition - Word classification	- Text generation - Text completion - Creative writing

Table 1: Comparison of different models

BERTweet was initially tested for our use case, as its pretraining on tweet data appeared to be perfect for tweet normalization. However, upon extensive experimentation, we identified two key issues, namely:

- BERTweet specialises in tasks like sentiment analysis, word classification, etc., but is not well suited for text generation tasks. There is difficulty in decoding the word embeddings produced by the model.
- BERTweet is only trained almost exclusively on English only, making it unsuitable for our scenario of multiple languages.

Following this, other attempts were also made to handle the complexity of mixed-language coded sentences. One of the approaches was to treat the other language as noise and remove it completely, and rely on models like BERT to predict the missing words. However, we found that the words in the other language contained semantic meanings that were too important to ignore. There were also common cases where a sentence almost evenly consisted of both languages. In these cases, the end result was too unsatisfactory, and the meaning completely deviated from the original sentence.

Word-level translation was considered to overcome this obstacle, but due to its weakness in capturing the overall context, it was rejected. We finally settled with the T5 architecture we have currently, which is pretrained on both English and Malay languages, and can perform sentence-level translation as well, while also being suited for text generation.

Table 1 shows the comparison of different models that we considered for our project. In the end, T5 transformer was chosen as it fitted our use case the best, whereas GPT (ChatGPT 3.5) was used as an LLM to provide reference translations for model distillation. While there are existing tweet normalization systems, there is very little work on Malaysian tweets specifically, particularly without the use of LLMs. Thus, our project provides a more lightweight model that is fine-tuned to serve this niche.

3.2 Output

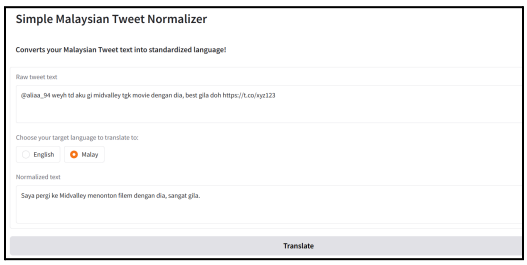

Figure 10.1: Normalized Output in Malay

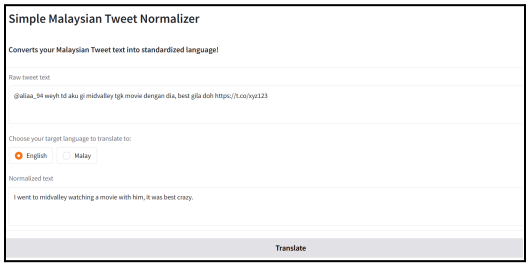

Figure 10.2: Normalized Output in English

Input Raw Text	Normalized Output (Malay)	Normalized Output (English)
@aliaa_94 weyh td aku gi midvalley tgk movie dengan dia, best gila doh https://t.co/xyz123	Saya pergi ke Midvalley menonton filem dengan dia, sangat gila.	I went to midvalley watching a movie with him, it was best crazy.
@ahmed weh serious ke exam next week?? aku blm study apa2	Adakah anda serius untuk peperiksaan minggu depan? Saya sedang mengkaji apa- apa.	Is it serious for the next week' s exam? I don' t study what.

Table 2: Example of Raw Malaysian Tweets and Their Corresponding Normalized Outputs in Malay and English

Table 2 presents sample outputs generated by the Malaysian Tweet Normalization System, showcasing the transformation of raw informal tweets into their normalized forms in both Malay and English. Each raw tweet underwent a comprehensive preprocessing stage where noise such as URLs (e.g., https://t.co/xyz123) and user mentions (e.g.,@aliaa_94, @ahmed) were successfully removed. These elements are not linguistically relevant and are typically excluded from formalized or summarized representations. Additionally, the input text was lowercased, emojis were removed or ignored, and informal words were cleaned before being passed to the language identification and normalization pipeline.

The examples in *Figure 10.1* demonstrate the system's ability to handle informal tweet inputs, including abbreviations such as "td" (tadi), "gi" (pergi), and "tgk" (tengok), which were successfully normalized into standard Malay and English. However, the English output in the first example includes the phrase "best crazy", a literal translation of "best gila" that does not reflect natural English usage, highlighting a limitation in capturing cultural context. Similarly, in the second example, while the Malay sentence is grammatically correct, the English version, "I don't study what" lacks semantic clarity due to the informal phrasing of the input. The time taken for the normalization is negligible (~1 second).

Overall, the system demonstrates effective handling of informal vocabulary, code-switching, and syntactic structure. However, further refinement is needed in semantic accuracy during translation and in improving the formality of the final output, especially in handling non-informative elements such as laughter expressions or filler text.

3.3 Evaluation

Epoch	Training Loss	Validation Loss	Bleu
1	0.508100	0.286034	0.226774
2	0.257800	0.278445	0.240525
3	0.213600	0.279649	0.239281
4	0.191500	0.287934	0.240476
5	0.167800	0.292187	0.240996

Figure 11.1 Malay Translation Model Training Summary

Epoch	Training Loss	Validation Loss	Bleu
1	0.551200	0.331424	0.187028
2	0.309700	0.320883	0.190653
3	0.258800	0.320234	0.197320
4	0.231000	0.325980	0.195752
5	0.208500	0.330423	0.195093

Figure 11.2 English Translation Model Training Summary

The evaluation of our Malaysian tweet normalization system was conducted using BLEU (Bilingual Evaluation Understudy) scores, which measure the quality of machine-generated translations compared to reference translations through n-gram overlap analysis. In our use case, the reference translations are translations generated by ChatGPT 3.5, as we aim to do model distillation from the LLM that is significantly more capable in this translation task. This was an acceptable substitute for human-translations, which we do not have.

For translation to Malay language, the initial baseline before fine-tuning was a score of 0.1796. This showed the weakness of the initial model in normalizing tweet text, despite being pretrained on Malay and English language from various Malaysian sources. This is inferred to be due to the unique and much more noisy structures of lingo used on Twitter/X. Therefore, fine-tuning was a reasonable strategy to overcome this. According to Figure 11.1, the training summary reveals consistent improvement across all five epochs, demonstrating stable learning without significant overfitting. The final overall BLEU score after fine-tuning was 0.4003, which demonstrated more than 2x improvement in translation quality. This is attributed to the T5 transformer effectively learning the patterns and nuances of the Malay text normalization from the dataset.

For translation to English language, the results were similar, improving from 0.1796 to 0.3259 after fine-tuning, with similar performance patterns. This score still represents acceptable translation quality for the English normalization task, with a significant increase still over the baseline results. However, the lower scores in translation to English can be attributed to the abundance of Malay text in the original dataset. Many of the original tweets in the dataset we used consisted predominantly of Malay language, which means the normalization to Malay language yielded better and more consistent results with the reference translation, as there were less cross-language translation. Translation to English however involved much more translating tasks from Malay to English, which leaves more room for variances and errors, thus resulting in lower scores due to its differences with the reference translation. This shows that the limitations of our dataset may be a factor to consider in future improvements.

Both models achieved BLEU scores that indicate meaningful improvements in text normalization quality, with the corpus BLEU scores reflecting the overall translation quality achieved after the complete training process. From the results discussed in the previous section as well, though there may at times be some variances and odd sentence patterns, the overall semantic meaning is still preserved and understandable. The model only encounters more problems in more niche lingo and sentence patterns that are less commonly used, which results in odd formalized normalizations.

4. DISCUSSION

4.1 Strengths

One of the main strengths of our approach is that it is able to capture the general semantic meaning of the original text consistently. In normalization tasks, we think that preserving the semantic meaning should be the highest priority, such that it will remain understandable and readable to the end user. After the fine-tuning of our model, our model is able to successfully adapt to the structure and vocabulary of informal tweets, which typically contain a mix of casual phrasing, spelling variations, and grammar inconsistencies. This targeted training enables the model to generate outputs that are not only grammatically correct but also contextually appropriate. Variations in the wording or structure of our output text still preserve the semantic meaning of the original.

Another notable strength is the system's versatility in handling dual-language (code-switched) tweets, a common phenomenon in Malaysian social media where users often mix English and Malay in a single sentence. The integration of token-level language identification using FastText allows the system to correctly label and process mixed-language input. With appropriate identification and normalization steps, this can be transformed into a more formal and coherent sentence. This ability to intelligently handle and separate languages at the token level makes the system robust and applicable in real-world, multilingual environments.

Additionally, the system includes a post-processing stage that restores punctuation and capitalization, which enhances the fluency and readability of the output. Many informal tweets lack proper punctuation or are written entirely in lowercase. The model also shows the ability to capture the general semantic meaning of input text, even when the original tweet contains informal or incomplete expressions. This is particularly useful in understanding the intent behind short or fragmented phrases and reformulating them into full, meaningful sentences.

4.2 Weakness

Despite the promising results and multiple strengths of the system, there are several notable weaknesses that currently limit its overall accuracy, generalization, and scalability.

One significant limitation is the system's inability to recognize and preserve named entities, such as personal names, geographic locations, and culturally important terms. In its current form, the model treats all input tokens equally and lacks an explicit mechanism to identify proper nouns. As a result, essential entities may be mistakenly translated, replaced with generic terms, or completely omitted during the normalization process. This is especially true for more local Malaysian names, like local landmarks.

Another major weakness lies in the system's limited ability to handle informal language variations such as slang, abbreviations, misspellings, and stylized expressions, which are common in Malaysian tweets. Users often employ creative spellings and regional short forms like "tp" (tapi), "xleh" (tak boleh), or "camtu" (macam itu), many of which are not present in the training data or the fixed dictionaries used for correction. Because of this, the model may either fail to normalize these tokens or apply incorrect replacements, resulting in poor quality outputs. This is made worse from our low sample size in fine-tuning, causing many language variations to not be learned.

4.3 Future Improvement

To further enhance the accuracy, adaptability, and robustness of the Malaysian Tweet Normalization System, several future improvements are proposed. First, the current approach of using separate models for English and Malay output can be improved by implementing joint normalization using multilingual transformer models such as mBART or mT5. These models are trained on multiple languages and are capable of handling code-switched inputs more naturally. By removing the need for users to pre-select a language, this method can produce more coherent and semantically consistent outputs for mixed-language tweets, which are common in Malaysian social media contexts.

Second, the system's performance can benefit from back-translation as a form of data augmentation. This technique involves translating a sentence into another language and then back to the original language, producing multiple paraphrased versions of the same content. By training on these variants, the model is exposed to more diverse sentence structures and vocabulary, which improves its fluency and generalization when normalizing informal or creatively phrased tweets.

Lastly, to address the current weakness in handling proper nouns, the integration of a Named Entity Recognition (NER) module is recommended. Tweets often include names of people, places, or organizations, and misprocessing these entities can result in semantic loss or confusion. A dedicated NER component can identify and preserve these tokens before the normalization stage, ensuring that important contextual information remains intact in the output. Alongside this, a larger dictionary can be kept to handle more variations of local slang, lingo and abbreviations that are commonly yet inconsistently found in Malaysian tweets. Alternatively, a larger sample size and more datasets can be sourced for fine-tuning for this purpose.