AI

MDPI

Article

AppHerb: Language Model for Recommending Traditional Thai Medicine

Thanawat Piyasawetkul [1], Suppachai Tiyaworanant [2] and Tarapong Srisongkram [3,*]

[1] Doctor of Pharmacy Program, Faculty of Pharmaceutical Sciences, Khon Kaen University, Khon Kaen 40002, Thailand; thanawat.piya@kkumail.com
[2] Division of Pharmacognosy and Toxicology, Faculty of Pharmaceutical Sciences, Khon Kaen University, Khon Kaen 40002, Thailand; suptiy@kku.ac.th
[3] Division of Pharmaceutical Chemistry, Faculty of Pharmaceutical Sciences, Khon Kaen University, Khon Kaen 40002, Thailand
* Correspondence: tarasri@kku.ac.th

Abstract

Trust in Traditional Thai Medicine (TTM) among Thai people has been reduced due to a lack of objective standards and the susceptibility of the general population to false information. The emergence of generative artificial intelligence (Gen AI) has significantly impacted various industries, including traditional medicine. However, previous Gen AI models have primarily focused on prescription generation based on Traditional Chinese Medicine (TCM), leaving TTM unexplored. To address this gap, we propose a novel fast-learning fine-tuned language model fortified with TTM knowledge. We utilized textual data from two TTM textbooks, Wat Ratcha-orasaram Ratchaworawihan (WRO), and Tamra Osot Phra Narai (NR), to fine-tune Unsloth's Gemma-2 with 9 billion parameters. We developed two specialized TTM tasks: treatment prediction (TrP) and herbal recipe generation (HRG). The TrP and HRG models achieved precision, recall, and F1 scores of 26.54%, 28.14%, and 24.00%, and 32.51%, 24.42%, and 24.84%, respectively. Performance evaluation against TCM-based generative models showed comparable precision, recall, and F1 results with a smaller knowledge corpus. We further addressed the challenges of utilizing Thai, a low-resource and linguistically complex language. Unlike English or Chinese, Thai lacks explicit sentence boundary markers and employs an abugida writing system without spaces between words, complicating text segmentation and generation. These characteristics pose significant difficulties for machine understanding and limit model accuracy. Despite these obstacles, our work establishes a foundation for further development of AI-assisted TTM applications and highlights both the opportunities and challenges in applying language models to traditional medicine knowledge systems in Thai language contexts.

Keywords: Generative AI; traditional Thai medicine; artificial intelligence; fine-tuning; Low-Rank Adaptation; natural language processing

check for updates

Academic Editor: Gianni D'Angelo

Received: 24 June 2025
Revised: 23 July 2025
Accepted: 24 July 2025
Published: 29 July 2025

Citation: Piyasawetkul, T.; Tiyaworanant, S.; Srisongkram, T. AppHerb: Language Model for Recommending Traditional Thai Medicine. *AI* **2025**, *6*, 170. https://doi.org/10.3390/ai6080170

1. Introduction

Traditional Thai Medicine (TTM) has been the foundation of the healthcare system in Thailand for generations. For centuries, TTM and other traditional medicines (TMs) have held a significant place in global health, and the WHO recognizes their importance by incorporating TM knowledge and technologies in primary healthcare [1]. TTM, like other TM systems, faces the common challenge of being rooted in empirical knowledge with limited objective standards. This results in complex and often ambiguous relationships among formulations, individual herbs, and their associated therapeutic effects [2]. While TTM provides affordable healthcare solutions, patients with limited knowledge are vulnerable to false claims about the efficacy of traditional remedies, which poses significant concerns [3,4]. Therefore, it is essential to address this issue through evidence-based validation and improved public education.

Artificial intelligence (AI) has become a popular tool in various fields, including healthcare. Generative AI (Gen AI) is a subset of AI that has gained significant attention for its capacity to generate new data [5]. However, current models cannot answer TTM-specific questions or perform related tasks correctly. This lack of TTM knowledge will likely provide inaccurate information [6,7] for those who wish to seek answers to health-related questions. Because there is no TTM-related chatbot in Thailand, we aim to fill this gap by developing a fine-tuned TTM language model.

Working with the Thai language for natural language processing (NLP) presents unique technical challenges. Thai is considered a low-resource language due to the limited availability of annotated data sets and domain-specific corpora for training robust models. In addition, Thai text is written without explicit word delimiters or spaces between words, making essential NLP tasks such as word segmentation considerably more challenging compared to languages like English or Chinese. These fundamental linguistic limitations complicate the development of accurate and efficient AI models tailored for Thai, particularly in specialized domains such as TTM.

To guide our investigation, we focus on two core research questions: First, can LoRA-tuned large language models (LLMs) effectively encode and generate accurate TTM recipes and treatment predictions from available data? Second, how does the performance of these TTM-specific models compare to large language models trained or fine-tuned on related domains, such as Traditional Chinese Medicine (TCM)? By explicitly addressing these questions, our study aims to evaluate the potential of advanced language modeling techniques for capturing complex and domain-specific knowledge in a low-resource language setting.

We utilized advanced techniques such as Low-Rank Adaptation (LoRA) [8]. This method could improve the adaptability and efficiency of language models for TTM knowledge adaptation. We developed two variants of language models for different tasks: treatment prediction (TrP) and herbal recipe generation (HRG). Then, we collectively named this framework AppHerb. After this, we tested the performance of the two models using an external test set and evaluated their performance using standard evaluation metrics. Our aim was that these models would ensure that users received accurate and safe information regarding TTM treatment. Furthermore, the model's ability to analyze complex relationships among formulae, herbs, and efficacy could potentially facilitate new insights and discoveries in TTM.

2. Materials and Methods

2.1. Material

We used a cloud computing service called Kaggle, which operates using the Python programming language (version 3.10.12). Kaggle utilized two units of NVIDIA Tesla T4 GPUs as accelerators [9]. It is important to note that deep learning libraries like Unsloth (version 066ec25 on 6 February 2025) and PyTorch (version 2.5.1) require NVIDIA GPUs to perform parallel computing tasks efficiently [10,11]. The versions of Unsloth and PyTorch were the latest at the time of the development.

2.2. Base Model Selection

To select the best model suitable for our development, we refer to the exam scores of the Thai LLM. One study assesses the performance of the Thai LLM with a model size of less than 14 billion parameters, using the M3Exam benchmark [12] and various Thai examination sets, consisting of the Ordinary National Education Test (ONET), the Investment Consultant (IC) exam, the Thai General Aptitude Test (TGAT), the Thai Professional Aptitude Test 1 (TPAT-1), and the Applied Knowledge Level (A-level) [13]. The reason for selecting a model size of less than 14 billion parameters was based on the capacity assessment of our training hardware. The average results are shown in Table 1.

Table 1. Top 10 Thai LLM leaderboard on 30 October 2024 with the number of parameters less than 14 billion parameters.

No.	Model Name	Average Exam Score
1	api/claude-3-5-sonnet-20240620	68.41
2	api/gpt-4o-2024-05-13	66.26
3	api/gpt-4o-mini-2024-07-18	56.18
4	unsloth/gemma-2-9b-it	53.78
5	Tsunami-th/Tsunami-0.5x-7B-Instruct	53.08
6	openthaigpt/openthaigpt1.5-7b-instruct	53.03
7	Tsunami-th/Tsunami-0.5-7B-Instruct	52.62
8	SeaLLMs/SeaLLMs-v3-7B-Chat	51.38
9	Qwen/Qwen2.5-7B-Instruct	49.70
10	Konthee/Qwen2.5-7B-ThaiInstruct	48.44

Excluding language models that use an application programming interface (API), the unsloth/gemma-2-9b-it exhibited the best average exam score among all models. Therefore, we utilized the variant of this model called unsloth/gemma-2-9b-bnb-4bit (Gemma-2U) as it possesses the best demonstration of Thai knowledge in the pre-training phase with a 4-bit BitsAndBytes format.

2.3. Data Preparation

Data were obtained by extracting herbal recipes and symptoms from two TTM resources: Wat Ratcha-orasaram Ratchaworawihan (WRO) and Tamra Osot Phra Narai (NR). The extraction process consisted of manual transcription from books to the computer's textual data in spreadsheet format.

Then, the data were cleaned by removing errors and inconsistencies in data to ensure model training efficiency [14]. Given that this step would consume the majority of the duration, effective data handling would substantially reduce both time and effort [15]. Since TTM diseases compress several underlying symptoms, we extracted these symptoms into list variables based on the WRO or NR itself. We used a mix of automated (see Algorithm 1) and manual replacement methods to minimize errors and ensure data validity. Human-validated JSON file formatting was used for ontological mapping. We acknowledged this process as the most critical step; therefore, we manually verified the spell corrections and TTM references to ensure accuracy (see Supplementary Materials).

We prepared two types of data sets to pass into the language model:

1. Treatment Prediction (TrP) for 405 rows—The model received cleaned data of herbal recipes as x and made next-word predictions as y about treatment based on the provided x.
2. Herbal Recipe Generation (HRG) for 256 rows—The model received cleaned data of symptoms presented as x and made next-word predictions as y, crafting appropriate

herbal recipes to cure the provided x. Some formulas can treat the same symptoms, so they were merged.

Algorithm 1 Python Algorithm Used in Data Preprocessing.

```
Input: DataFrame(X) = [Code, raw_data, herb_list, symp_list]
Output: DataFrame(Y) = [Code, raw_context, herbsymp] : shape = (N, 3)
1. Select Columns: It keeps only the columns "Code", "raw_data", "herb_list", and
   "symp_list" from a manually validated csv data set.
2. Clean Text: For every string column, it removes newline characters.
3. Remove Invalid Data: It drops any rows that have missing (null) values or are
   duplicate rows.
4. Validate and Convert "symp_list":
    a. It checks whether each value in the "symp_list" column can be converted
       (using eval) from a string representation to a Python list.
    b. For debugging, it prints the "Code" value of any row that cannot be
       converted.
    c. It then actually converts the "symp_list" column from strings to Python lists.
5. Format the List: It reformats each "symp_list" so that if it contains multiple items,
   the Thai word "และ" (which means "and") is inserted before the last item, making
   the phrasing natural in Thai.
6. Add a Combined Text Column: It creates a new column called "herbsymp" that
   summarizes information about the herbs and symptoms. This column uses a
   template that states the components (herb_list) and what conditions they treat
   (symp_list), with Thai-language labels.
7. Normalize Digits: It converts any Thai numerals in the "raw_context" column into
   standard Arabic numerals.
8. Output: It returns a new DataFrame with only the columns "Code",
   "raw_context", and "herbsymp", ready for use in training a model.
```

For the train–test data split, we divided the train and test sets in a ratio of 9:1 to ensure that the model would learn the most from our limited and relatively small data set. With this setting, the training and test sets for the TrP data set were 364 and 41 rows, respectively. For the HRG data set, the training and test sets contained 229 and 26 rows, respectively.

2.4. Model Setup

2.4.1. Import Pre-Trained Model and Tokenizer

The tokenizer and pre-trained model were both loaded from the same directory, Gemma-2U, to ensure consistent tokenization and embedding representations. The model was imported using 4-bit precision to optimize memory efficiency without compromising performance. The 4-bit size minimizes GPU memory usage, enabling more training efficiency and time processing. The loading process was facilitated by the Unsloth library [16], which is built on PyTorch [17].

2.4.2. Prompt Engineering

We adopted the Alpaca prompt format as a template and translated it into Thai. While the model can handle various formats, including Alpaca, our instructions are constructed based on best practices in prompt engineering, which emphasize clarity, precision, and role-prompting [18]. For each task, two distinct instructions will be employed, as demonstrated. There is no variant in the prompt for each task. The full prompt is listed in Supplementary Materials.

2.4.3. Low-Rank Adaptation

The key variables used are the rank of the low-rank matrices (r) and lora_alpha. The rank of the low-rank matrices added during fine-tuning was 8, and lora_alpha is a scaling factor that controls the level of influence of LoRA layers on the final model's behavior [19]. Table 2 provides the complete set of LoRA parameters to facilitate reproducibility of our experimental results. All values were based on Unsloth's recommendation for Gemma-2U.

Table 2. LoRA Parameters Configuration.

Variable Name	Python Variable	Value
Rank	r	8
Alpha	lora_alpha	16
Drop out	lora_dropout	0
Bias	bias	"none"
Random state	random_state	3407
Rank stabilized LoRA	use_rslora	False
LoftQ	loftq_config	False

2.4.4. Fine-Tuning

We fine-tuned each model three times to calculate the average training loss using the Unsloth framework, which is built upon Gemma-2, an open-source model developed by Google that underpins the Gemini family [20], as illustrated in Figure 1.

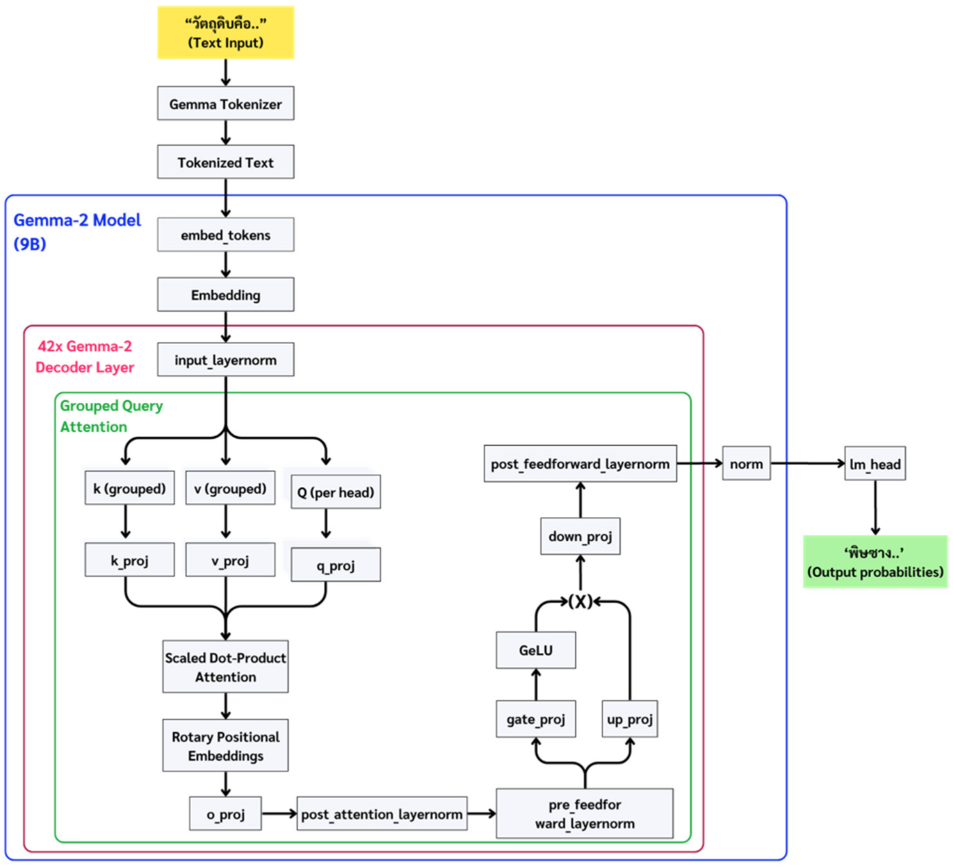

Figure 1. Gemma-2 9B Model Architecture.

We imported Gemma-2U, a 9-billion-parameter (9B) model. Subsequently, the Unsloth framework was applied to modify the original architecture by integrating 42 Query, Key, and Value (QKV) layers. This modification significantly enhanced memory and time efficiency, allowing fine-tuning to run twice as fast while reducing video random access memory (VRAM) usage by 63% [21]. Since VRAM stores GPU-related data [22], lowering its consumption allows more memory for other computations during training. Notably, Gemma-2 employs a specialized attention mechanism known as group query attention, which delivers quality comparable to multi-headed attention and speed on par with multi-query attention [23,24]. The detailed trainer configurations are provided in Table 3.

Table 3. Trainer Parameter Configurations.

Variable Name	Python Variable	TrP and HRG
Steps	max_steps	300
Batch Size	per_device_train_batch_size	2
Gradient Accumulation	gradient_accumulation_steps	4
Warmup Steps	warmup_steps	10
Learning Rate	learning_rate	5.00×10^{-5}
Use Half Precision	fp16	False
Use Brain Float 16	bf16	True
Optimizer	optim	“adamw_8 bit”
Weight Decay	weight_decay	0.1
Learning Rate Scheduler	lr_scheduler_type	“cosine”
Seed	seed	3407

2.5. Model Evaluation

Prior to evaluation, the saved text was tokenized using the PyThaiNLP *deepcut* engine [25], which offers the most granular Thai word segmentation among available tokenization tools [26].

The main modes of evaluation were the precision, recall, and F1 score. $\text{LCS}(R, G)$ was the length of the longest common subsequence. $|G|$ was the length of generated tokens, where G represents the set or sequence of generated tokens, and the length of reference tokens was denoted as $|R|$, where R represents the set or sequence of reference tokens.

$$\text{Precision} = \frac{\text{LCS}(R, G)}{|G|} \tag{1}$$

$$\text{Recall} = \frac{\text{LCS}(R, G)}{|R|} \tag{2}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

To further capture semantic analysis, we used BERTScore to compute semantic similarity between generated tokens and reference tokens through contextual embeddings from a transformer model [27]. Specifically, we employed the “xlm-roberta-base” model [28], which is a multilingual model that supports the Thai language. We also included the Bilingual Evaluation Understudy (BLEU) score, a precision-based metric that measures the number of contiguous word sequences in the generated text that match those in the reference text.

Each task required specific parameter settings to achieve optimal evaluation metrics for the traditional precision, recall, and F1 score, as Gemma-2U occasionally generated repetitive and nonsensical text. To mitigate this, we experimented with various configurations involving three key parameters as shown in Table 4: Repetition penalty, Min-P, and Temperature:

Table 4. Configuration Used for TTM Tasks: TrP and HRG.

Variable Name	Python Variable	Value	
		TrP	HRG
Repetition penalty	repetition_penalty	1.1	1.2
Min-P	min_p	0.6	0.6
Temperature	temperature	0.75	0.85

Repetition penalty (repetition_penalty) penalizes repetitive tokens, encouraging more meaningful and varied responses [29].

Min-P (min_p) limits the model's sampling to a smaller set of high-probability tokens. High value ensures that the model samples only from the most probable next token [30].

Temperature encourages more deterministic outputs while maintaining some creativity [31].

3. Results

For the TrP training data set, Gemma-2U exhibited a maximum average training loss of 270.00%, which progressively decreased over the course of 300 training steps, reaching a minimum loss of 18.20%, as illustrated in Figure 2. Similarly, for the HRG training data set, the model recorded a maximum average training loss of 312.25%, which was reduced to 19.78% after 300 training steps, as shown in Figure 3. The AppHerb fine-tuning process took approximately 59 min to 64 min for TrP and 66 min to 74 min for HRG.

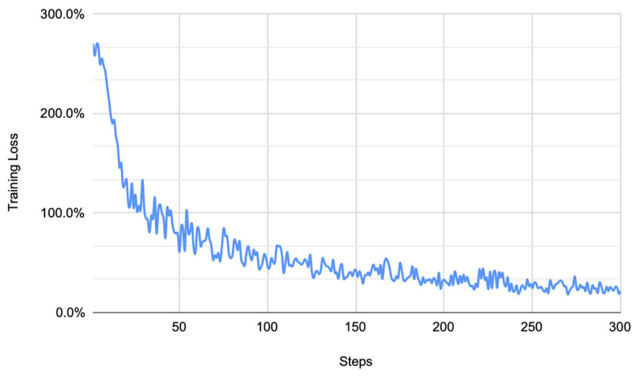

Figure 2. Average Training Loss Result for Executing TrP Three Times on Gemma-2U.

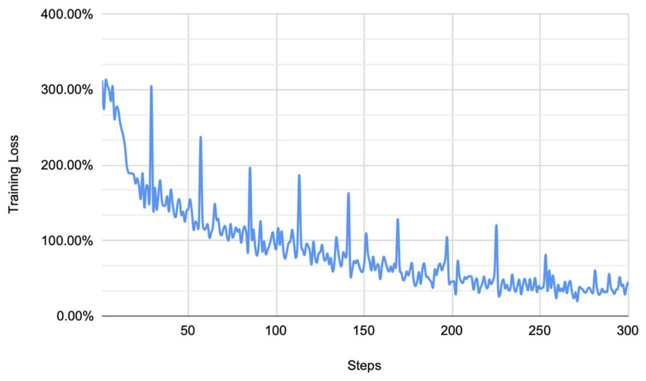

Figure 3. Average Training Loss Result for Executing HRG Three Times on Gemma-2U.

Gemma-2U was trained separately for different data sets, leading to the first version of AppHerb, which includes two variations: TrP task and HRG task. We evaluated system performance using mean (triplicate) precision, recall, and F1 scores with 95% bootstrapped confidence intervals to account for test set variability. For the test set, the TrP exhibited mean precision, recall, and F1 scores of 26.54, 28.14, and 24.00 percent, respectively. Similarly, the HRG exhibited mean precision, recall, and F1 scores of 32.51, 24.42, and 24.84 percent, respectively. All AppHerb test performance is shown in Table 5, where "Gemma-2U-" represents the original model before fine-tuning with the bootstrapped confidence interval included.

Table 5. AppHerb Test Performance Results Before (Gemma-2U) and After Fine-Tuning.

Model-Task	**Precision**		**Recall**		**F1**	
	Mean	**95% CI**	**Mean**	**95% CI**	**Mean**	**95% CI**
AppHerb-TrP	26.54	[21.95–31.76]	28.14	[23.38–33.46]	24.00	[18.65–28.89]
Gemma-2U-TrP	2.70	[2.22–3.19]	30.37	[28.59–32.24]	4.44	[3.80–5.09]
AppHerb-HRG	32.51	[29.12–35.58]	24.42	[22.47–26.56]	24.84	[23.19–26.70]
Gemma-2U-HRG	17.46	[14.24–20.91]	3.51	[2.37–5.24]	4.44	[3.33–5.94]

AppHerb's BERTScore: precision, recall, and F1 exhibited 84.10, 84.77 and 84.39, respectively, for TrP, and 85.19, 84.24 and 84.68, respectively, for HRG (see Table 6). Additionally, AppHerb-TrP and -HRG's BLEU scores were 74.28 and 76.16, respectively (see Table 7). These results highlight the importance of reporting confidence intervals alongside point estimates, particularly with limited test sizes.

Table 6. AppHerb Test BERTScore Results Before and After Fine-Tuning.

Model-Task	**BERT Precision**		**BERT Recall**		**BERT F1**	
	Mean	**95% CI**	**Mean**	**95% CI**	**Mean**	**95% CI**
AppHerb-TrP	84.10	[74.18–100.00]	84.77	[83.73–86.01]	84.39	[83.33–85.45]
Gemma-2U-TrP	74.62	[67.90–78.99]	80.28	[79.98–80.59]	77.33	[76.99–77.67]
AppHerb-HRG	85.19	[80.92–89.60]	84.24	[83.69–84.75]	84.68	[84.27–85.11]
Gemma-2U-HRG	77.58	[74.90–81.99]	74.81	[74.04–75.63]	76.16	[75.62–76.78]

Table 7. AppHerb Test BLEU Results Before and After Fine-Tuning.

Model-Task	**BLEU**	**[95% CI]**
AppHerb-TrP	74.28	[73.01–75.63]
Gemma-2U-TrP	22.99	[22.64–23.31]
AppHerb-HRG	76.16	[75.38–76.90]
Gemma-2U-HRG	10.33	[9.39–11.23]

4. Discussion

In this study, we found that the HRG model shows inconsistency in training loss reduction compared to the TrP model. This may be attributed to the structure of the HRG data set, in which the input (x) is relatively brief, while the expected output (y) is more complex and lengthier. In contrast, the TrP data set exhibits the opposite structure, where the input (x) is relatively long and descriptive, while the output (y) consists of a short disease name or clinical outcome. To enhance the training performance of the HRG model, future studies should consider incorporating more diverse data. Additionally, adjusting the LoRA parameters or adopting a new base model could further improve the model's generative capabilities.

AppHerb-TrP achieved a BERT F1 score of 84.39 (95% CI: [83.33–85.45]), compared to Gemma-2U-'s 77.33 (95% CI: [76.99–77.67]), indicating a notable improvement in the model's ability to correctly identify and generate relevant treatment outputs after fine-tuning. Precision and recall were also improved, with a marked improvement in precision (84.10 vs. 74.62), suggesting that AppHerb-TrP produced more accurate and relevant treatment predictions with fewer false positives. AppHerb-HRG outperformed its base model, with a BERT F1 score of 84.68 (95% CI: [84.27–85.11]) compared to 76.16 (95% CI: [75.62–76.78]) for the base model. Gains in both precision and recall indicate that the fine-tuned model was not only generating more precise herbal recipes but was also better at capturing relevant components.

We observed a linguistic complexity in the data cleaning step. Gemma-2U was pre-trained on Thai examinations, which is based on modern Thai language, while our TTM data was a traditional form of the Thai language used at least 200 years ago. We preserved the original wording for two reasons: to maintain familiarity for TTM practitioners, and because some words cannot be accurately translated into modern Thai. The significant change made was expanding the disease into symptoms (TrP data set), and a mix of manually and automatically extracting plant common names from the data (HRG data set).

A total of 9 out of 41 cases exhibited precision higher than 50 percent, and they were related to the fire element (5 cases), wind element (2 cases), and urinary tract-related disease (1 case). When examining the results generated by the HRG model, higher performance was observed in cases related to skin diseases. Notably, 4 out of 27 cases achieved a precision of 50% or higher, with 2 of those cases involving skin-related conditions. This result suggests that the AppHerb model may be effective in addressing TTM and skin-related problems. See Supplementary Materials for more details.

After analyzing the best and worst performance from the test set, we found that reference tokens in the TrP task were shorter and typically contained only a single entity, which was sets of symptoms. Therefore, the scoring outcome tended to be all-or-none. F1 score ranged from 0 to 100%. In contrast, for HRG, the reference tokens were lengthier and richer in entities such as the part of the plant used, method of preparation, and symptoms of treatment. For this reason, the model's scoring outcome differed from that of the other task, as reflected in the HRG F1 scores, which ranged from approximately 13 to 37%. However, these scores are intended primarily for comparison with the original TTM textbooks. The generated output should be further evaluated by experts in TTM to ensure clinical relevance and appropriateness.

To better understand the generation hyperparameter, we conducted a sensitivity analysis to ensure the model's best performance and study how different combinations of hyperparameters affect the model score, as shown in Figure 4 for TrP and Figure 5 for HRG, and extended details are provided in Supplementary Materials. However, we acknowledge that our exploration of hyperparameter configurations was not expansive, as the peak model F1 scores remained modest (F1 = 24.00 and 24.84). Based on these results, we do not expect that more extensive hyperparameter tuning alone would yield a substantial improvement in performance without addressing broader challenges such as data quality, model architecture, or domain knowledge integration.

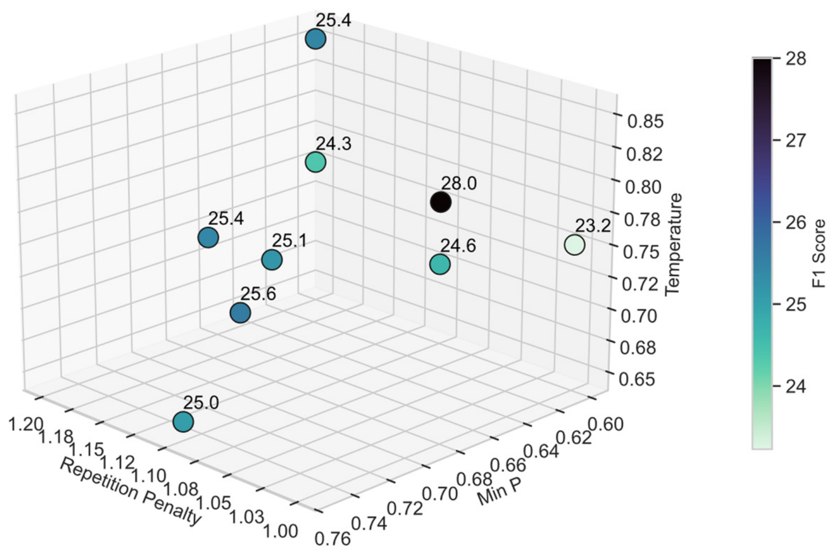

Figure 4. TrP F1 Score Hyperparameter Sensitivity Analysis.

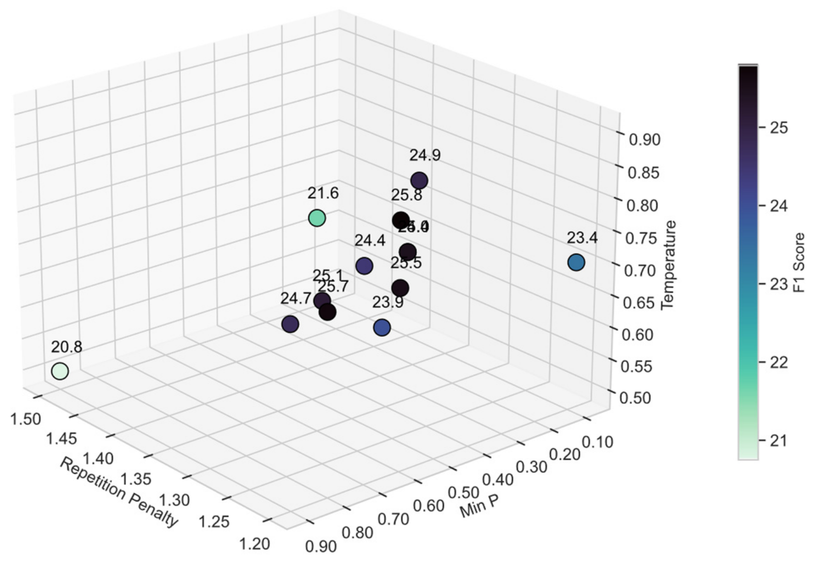

Figure 5. HRG F1 Score Hyperparameter Sensitivity Analysis.

LoRA has become a critical component in language model development, as it has been shown to deliver fine-tuning performance on par with, or even superior to, traditional methods in LLMs [8]. In this study, this process is not only used to address hardware limitations but also to preserve the original 9B parameters of the base model. During fine-tuning, only the newly added LoRA layers are updated, while the base model remains static. This approach significantly improves time efficiency and reduces hardware demands.

AppHerb (post-fine-tune stage) models significantly outperformed Gemma-2U (base model) for the different tasks, except the superior recall score for Gemma-2U-TrP. The model's performance was comparable to existing models, as stated in the literature review chapter. Table 8 and Figure 6 provide a preliminary comparison of AppHerb performance results in HRG with two much larger TCM language models that perform similar tasks: GSCCAM [32] and RoKEPG [33]. While the initial results (F1 = 24.84 vs. 29.99 and 25.92) appear encouraging, several important limitations must be considered. We used a 575-times smaller training data set (229 rows compared to over 100,000 rows in GSCCAM), which inherently caused overfitting and consequently constrained its generalizability. While the reported F1 scores (26–31%) demonstrate some predictive capability, these modest scores are far from indicating reliable performance. Our work represents an application of established methods to a specific domain rather than a novel methodological invention. Furthermore, this work constitutes a proof-of-concept that has not undergone clinical validation, which would be a necessary step before considering any practical applications in healthcare settings.

Table 8. AppHerb and TCM Models: GSCCAM and RoKEPG's Base Model and Training Data Comparison.

Model Name	Base Model	Training Data Size
AppHerb	Gemma-2	229 rows of Symptoms-prescription data set
GSCCAM	Custom	106,168 data for Symptoms-prescription 25,563 data for Prescription-efficacy
RoKEPG	RoBERTa	13.5 M words

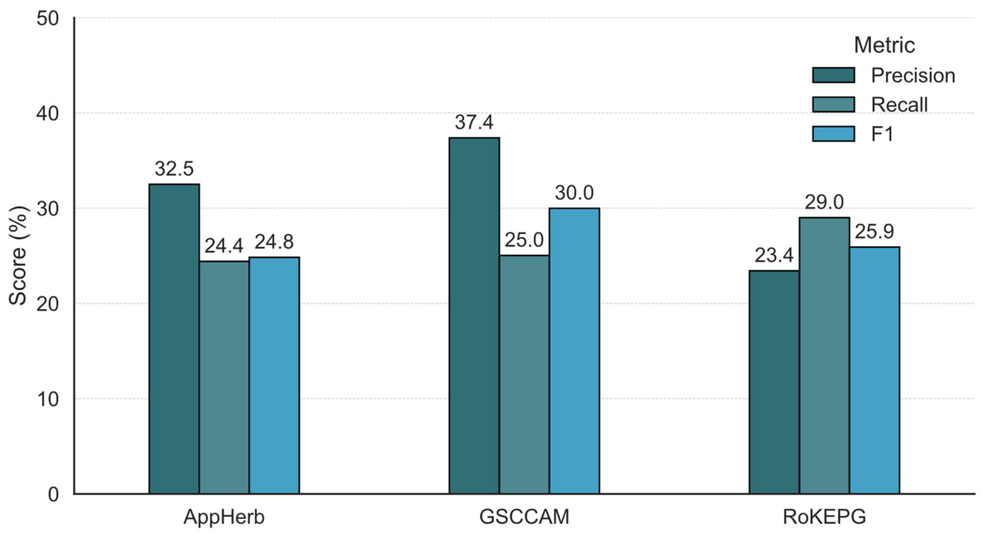

Figure 6. Comparison of AppHerb Performance to TCM Agents for Prescription Generation Task.

To enhance model transparency, we attempted to visualize a self-attention matrix of the base model. However, the Unsloth model does not allow output attentions. Consequently, we explored alternative Gemma-2 variants with characteristics similar to the Unsloth model (9B parameters and 4-bit quantization).

We selected the "StoyanGanchev/gemma-2-9b-nf4" model, which features 9B parameters and utilizes the NormalFloat4 datatype. Figure 7 illustrates the self-attention matrix generated for the sentence, แบบจำลองภาษาสำหรับแนะนำตำรับแผนไทย (Language model for recommending traditional Thai medicine). The input sentence was tokenized into [แบบ, จำ, ลอง, ภาษา, สำหรับ, แนะนำ, ตำ, รับ, แผน, ไทย] and self-attention was performed to explore inter-sentential relationships. The Gemma-2 model employed in this study comprises 42 layers, each with 16 attention heads. The visualized self-attention matrix represents the averaged attention weights across these 16 heads for a specific layer.

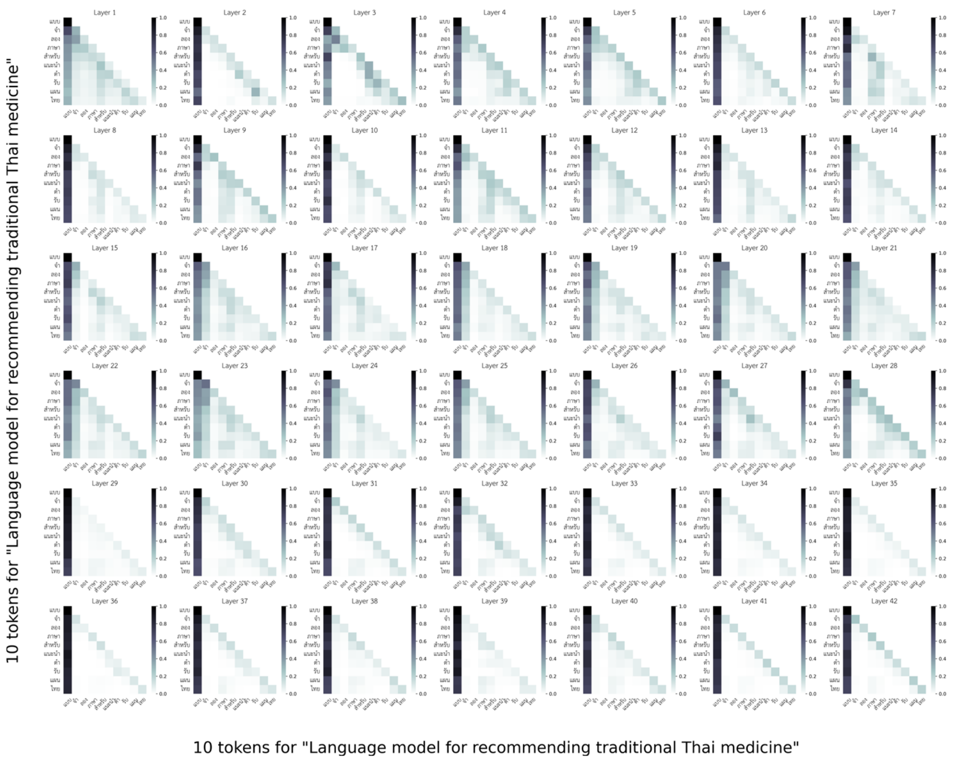

Figure 7. Thai Sentence's Self-Attention Matrix of a Gemma-2 Variant. The x- and y-axis represents series of tokens from the Thai sentence. Darker color shows high correlation between tokens, while lighter color shows low correlation between tokens.

For a given layer's attention tensor A (where $A_{b,h,i,j}$ is the weight from the source token j to the target token i in batch b and head h), between the i-th target and j-th source token for a single input (batch 1) across all H heads:

$$\overline{A_{i,j}} = \frac{1}{H}\sum_{h=1}^{H} A_{1,h,i,j} \tag{4}$$

The distribution of attention weights varied across the model's layers. A prominent pattern was the strong correlation of the first token's attention with all other tokens. However, it is important to emphasize that the group-query attention patterns we observed in the Gemma-2 variant workaround may not necessarily generalize to the LoRA-tuned model or other variants applied to different tasks or data domains. Moreover, we did not compare the model's performance and attention behavior between LoRA and non-LoRA variants, which may be worth exploring in a future study. A direct comparison would be valuable for validating and extending these attention mechanisms in the context of LoRA-based fine-tuning.

The challenge of this study is that it is based on the Thai language, which is a low-resource language, which may affect the accuracy of text generation in a multilingual model like the Gemma family. Unlike English or Chinese, the Thai language lacks explicit sentence boundary markers such as full stops, allowing for unlimited nested sub-sentences that can make it a difficult learning system for a machine. In terms of writing system, Thai language is an abugida, where consonants and vowels are combined in complex ways, and words are written without spaces—necessitating advanced segmentation techniques. In contrast, the Chinese language is a logographic writing system, where each character represents a word or concept. This comparison highlights the inherent complexity of the Thai language and the difficulties it poses for NLP tasks. A bilingual or monolingual LLM pre-trained on a Thai corpus may better assess the impact for low-resource language understanding and perhaps better utilize our TTM data set. Despite these challenges, our preliminary findings suggest potential for further development, while acknowledging the substantial differences in scale and scope between our application and the established Chinese medicine models we referenced.

Potentially, AppHerb could support a range of practical uses. In educational settings, the system may serve as an interactive tool for students, health practitioners, or the general public to explore TTM knowledge and receive clear explanations of traditional concepts, formulas, and practices. Educators might use the platform to design exercises that improve students' understanding of herbal ingredients, treatment methods, or the rationale behind specific TTM prescriptions. In clinical environments, a validated form of AppHerb could assist practitioners by offering rapid, possible suggestions for herbal treatments, facilitating the preparation of herbal formulas.

Future work will need to address the current limitations by expanding data sets, enhancing model transparency, collaborating with TTM practitioners, and refining the fine-tuning approach to improve performance and reliability for real-world testing. While our current model was limited to 14 billion parameters due to hardware constraints, future iterations may explore the use of larger-scale models, which could offer enhanced representational power and improved predictive performance. Additionally, the next version of model development may incorporate multilingual models or character-level tokenizers, which can help alleviate segmentation challenges in low-resource language data sets and improve model robustness across diverse languages and scripts.

5. Conclusions

This study presents a domain-specific application of language model fine-tuning for traditional Thai medicine, focusing on TrP and HRG tasks. By utilizing LoRA, the fine-tuning process became significantly faster, more memory-efficient, and hence more accessible, making it suitable for low-resource languages like Thai. Our proof-of-concept model achieved promising initial results when compared to the gigantic TCM models such as GSCCAM and RoKEPG. We acknowledge the substantial differences in scale, scope, and maturity between these projects. It is important to emphasize that clinical validation remains an essential next step that would need to be completed before consideration of practical implementation.

To the best of our knowledge, AppHerb represents one of the first attempts at developing a language model specifically for representing TTM knowledge in Thailand. While our work is modest in scale compared to established models in TCM, it provides a foundation for future research in this understudied domain.

For practical application, the AppHerb models are available for immediate testing and further development. The TrP and HRG models can be accessed via Hugging Face at https://huggingface.co/thanawatpi/appherb-treatment-pred-beta (accessed on 25 July 2025) and https://huggingface.co/thanawatpi/appherb-herb-rec-gen-beta (accessed on 25 July 2025), respectively. Additionally, a Thai-language chat interface is provided at https://github.com/aptheory/appherb-alpha (accessed on 25 July 2025) to demonstrate real-world usage. While these tools are readily accessible, clinical validation is essential before deployment in healthcare settings.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/ai6080170/s1, Data preparation phase: Table S1. Examples of raw data from WRO and NR combined, Table S2. Example of disease-to-symptom mapping, Table S3. Example of the preprocessed data set, Table S4. An example of the Treatment Prediction task (TrP), Table S5. Example data of the Herbal Recipe Generation task (HRG), Table S6. Alpaca prompt template in English and Thai, and Table S7. Instructions for TTM tasks. Tokenizer selection for evaluation: Table S8. Thai sentence tokenization results. Raw contents of text generation tasks: Table S9. TrP outputs that achieved more than 50 percent precision and Table S10. HRG outputs that achieved more than 50 percent precision.

Author Contributions: Conceptualization, T.P., S.T. and T.S.; methodology, T.P., S.T. and T.S.; data curation, T.P., S.T. and T.S.; formal analysis, T.P.; software, T.P. and T.S.; validation, T.P. and T.S.; investigation, T.P. and T.S.; resources, S.T. and T.S.; writing—original draft, T.P. and T.S.; writing—review and editing, T.S.; visualization, T.P. and T.S.; supervision, S.T. and T.S.; project administration, T.S.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Fund of Khon Kaen University from the National Science, Research, and Innovation Fund or NSRF, Thailand (Project no. 68A103000041).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: AppHerb models are available at huggingface.com: (1) https://huggingface.co/thanawatpi/appherb-treatment-pred-beta (TrP model; accessed on 25 July 2025). (2) https://huggingface.co/thanawatpi/appherb-herb-rec-gen-beta (HRG model; accessed on 25 July 2025). We also provide a chat interface in Thai, available at https://github.com/aptheory/appherb-alpha. However, the authors do not have the authority to disclose the data set or source code. Our models are LoRA adapters licensed for non-commercial use only.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

A-level	Applied Knowledge Level
AI	Artificial Intelligence
API	Application Programming Interface
BLEU	Bilingual Evaluation Understudy
CI	Confidence Interval
CPU	Central Processing Unit
DL	Deep Learning
Gen AI	Generative Artificial Intelligence
Gemma-2U	Unsloth's Gemma-2 with 9 billion parameters
GPU	Graphics Processing Unit
HRG	Herbal Recipe Generation
IC	Investment Consultant (exam)
LLM	Large Language Model
LoRA	Low-Rank Adaptation
NLP	Natural Language Processing
NR	Tamra Osot Phra Narai
NSRF	National Science, Research, and Innovation Fund
ONET	Ordinary National Education Test
QKV	Query, Key, and Value
QLoRA	Quantized Low-Rank Adaptation
TCM	Traditional Chinese Medicine

TGAT	Thai General Aptitude Test
TKMM	Task-Qualified Knowledge Base Matching Module
TPAT-1	Thai Professional Aptitude Test 1
TrP	Treatment Prediction
TTM	Traditional Thai Medicine
VRAM	Video Random Access Memory
WHO	World Health Organization
WRO	Wat Ratcha-orasaram Ratchaworawihan

References

1. World Health Organization. *Declaration of Astana*; World Health Organization: Geneva, Switzerland, 2018.
2. Cheng, N.; Chen, Y.; Gao, W.; Liu, J.; Huang, Q.; Yan, C.; Huang, X.; Ding, C. An Improved Deep Learning Model: S-TextBLCNN for Traditional Chinese Medicine Formula Classification. *Front. Genet.* **2021**, *12*, 807825. [CrossRef] [PubMed]
3. Roonkaseam, N. SHARE: A Framework and Tool for Decoding Fake News in Thai Social Context. *J. Innov. Media Amp Commun.* **2025**, *4*, 49–68. [CrossRef]
4. Wanset, S.; Onchomchant, D. A Study of Medicinal Plants Utilization of Folk Healers: A Case Study of Folk Healers in Chiang Rai Province Thailand. *J. Thai Tradit. Altern. Med.* **2018**, *16*, 420–435.
5. Reddy, S. Generative AI in Healthcare: An Implementation Science Informed Translational Path on Application, Integration and Governance. *Implement. Sci.* **2024**, *19*, 27. [CrossRef] [PubMed]
6. Bastani, H.; Bastani, O.; Sungu, A.; Ge, H.; Kabakcı, Ö.; Mariman, R. *Generative AI Can Harm Learning*; University of Pennsylvania: Pennsylvania, PA, USA, 2024.
7. Nov, O.; Singh, N.; Mann, D. Putting ChatGPT's Medical Advice to the (Turing) Test: Survey Study. *JMIR Med. Educ.* **2023**, *9*, e46939. [CrossRef] [PubMed]
8. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**, arXiv:2106.09685.
9. Kaggle Staff New GPU (T4s) Options & More CPU RAM. Available online: https://www.kaggle.com/discussions/product-feedback/361104 (accessed on 3 January 2025).
10. Franchitti, J.-C. Parallel Computing Overview. In *Introduction to Computer Science*; OpenStax: Houston, TX, USA, 2024.
11. Li, M.; Bi, Z.; Wang, T.; Wen, Y.; Niu, Q.; Liu, J.; Peng, B.; Zhang, S.; Pan, X.; Xu, J.; et al. Deep Learning and Machine Learning with GPGPU and CUDA: Unlocking the Power of Parallel Computing. *arXiv* **2024**, arXiv:2410.05686. [CrossRef]
12. Zhang, W.; Aljunied, S.M.; Gao, C.; Chia, Y.K.; Bing, L. M3Exam: A Multilingual, Multimodal, Multilevel Benchmark for Examining Large Language Models. *arXiv* **2023**, arXiv:2306.05179. [CrossRef]
13. Pipatanakul, K.; Jirabovonvisut, P.; Manakul, P.; Sripaisarnmongkol, S.; Patomwong, R.; Chokchainant, P.; Tharnpipitchai, K. Typhoon: Thai Large Language Models. *arXiv* **2023**, arXiv:2312.13951. [CrossRef]
14. Ridzuan, F.; Wan Zainon, W.M.N. A Review on Data Cleansing Methods for Big Data. *Procedia Comput. Sci.* **2019**, *161*, 731–738. [CrossRef]
15. Côté, P.-O.; Nikanjam, A.; Ahmed, N.; Humeniuk, D.; Khomh, F. Data Cleaning and Machine Learning: A Systematic Literature Review. *arXiv* **2024**, arXiv:2310.01765. [CrossRef]
16. Daniel Han, M.H.; Team, U. Unsloth 2023. Available online: http://github.com/unslothai/unsloth (accessed on 25 July 2025).
17. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in PyTorch. In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
18. Chen, B.; Zhang, Z.; Langrené, N.; Zhu, S. Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review. *arXiv* **2024**, arXiv:2310.14735.
19. Unsloth Documentation LoRA Parameters Encyclopedia. Available online: https://docs.unsloth.ai/basics/lora-parameters-encyclopedia (accessed on 22 December 2024).
20. Google for Developers Gemma Open Models. Available online: https://ai.google.dev/gemma (accessed on 28 December 2024).
21. Han, D.; Han, M. Finetune Gemma 2 with Unsloth. Available online: https://unsloth.ai/blog/gemma2 (accessed on 4 January 2025).
22. AceCloud Why GPU Memory Matters More Than You Think? Available online: https://acecloud.ai/resources/blog/why-gpu-memory-matters-more-than-you-think/ (accessed on 4 January 2025).
23. Ainslie, J.; Lee-Thorp, J.; Jong, M.d.; Zemlyanskiy, Y.; Lebrón, F.; Sanghai, S. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv* **2023**, arXiv:2305.13245.

24. Gemma Team; Riviere, M.; Pathak, S.; Sessa, P.G.; Hardin, C.; Bhupatiraju, S.; Hussenot, L.; Mesnard, T.; Shahriari, B.; Ramé, A.; et al. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv* **2024**, arXiv:2408.00118.
25. Phatthiyaphaibun, W.; Chaovavanich, K.; Polpanumas, C.; Suriyawongkul, A.; Lowphansirikul, L.; Chormai, P. PyThaiNLP: Thai Natural Language Processing in Python. In Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023), Singapore, 6 December 2023.
26. Kanoktipsatharporn, S. Python ตัดคำภาษาไทย ด้วย PyThaiNLP API ตัดคำ Word Tokenize ภาษาไทย ตัวอย่างการตัดคำภาษาไทย อัลกอริทึม Deepcut, Newmm, Longest, Pyicu, Attacut—PyThaiNLP Ep.2. Available online: https://www.bualabs.com/archives/3740/python-word-tokenize-pythainlp-example-algorithm-deepcut-newmm-longest-python-pythainlp-ep-2/ (accessed on 3 January 2025).
27. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. *arXiv* **2020**, arXiv:1904.09675. [CrossRef]
28. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised Cross-Lingual Representation Learning at Scale. *arXiv* **2020**, arXiv:1911.02116. [CrossRef]
29. TitanML Repetition Penalty. Available online: https://www.titanml.co/glossary/repetition-penalty (accessed on 7 January 2025).
30. Nguyen, M.; Baker, A.; Neo, C.; Roush, A.; Kirsch, A.; Shwartz-Ziv, R. Turning Up the Heat: Min-p Sampling for Creative and Coherent LLM Outputs. *arXiv* **2024**, arXiv:2407.01082.
31. Peeperkorn, M.; Kouwenhoven, T.; Brown, D.; Jordanous, A. Is Temperature the Creativity Parameter of Large Language Models? *arXiv* **2024**, arXiv:2405.00492. [CrossRef]
32. Hou, J.; Song, P.; Zhao, Z.; Qiang, Y.; Zhao, J.; Yang, Q. TCM Prescription Generation via Knowledge Source Guidance Network Combined with Herbal Candidate Mechanism. *Comput. Math. Methods Med.* **2023**, *2023*, 3301605. [CrossRef] [PubMed]
33. Pu, H.; Mi, J.; Lu, S.; He, J. RoKEPG: RoBERTa and Knowledge Enhancement for Prescription Generation of Traditional Chinese Medicine. *arXiv* **2023**, arXiv:2311.17307.