NTU IR Final Project Report

Team_17

b10705005 陳思如
b10204022 金京
b10501024 游善喆

2024 Fall

1 Introduction

This project aims to develop a reliable model capable of accurately retrieving and providing contextually relevant legal references based on user queries. The main challenges include addressing the linguistic disparity between colloquial user queries and formal legal language, as well as handling data preprocessing and model training to achieve optimal retrieval performance.

To address these challenges, we begin by standardizing the legal texts and queries to reduce language differences. We then use both the Traditional Architecture Method and the Large Language Model (LLM) Method. The Traditional Architecture Method relies on keyword matching and context analysis in retrieval models, while the LLM Method uses deep learning techniques to understand query semantics and provide more accurate answers. After retrieving results from the BiEncoder and LLM, we refine them through postprocessing, which involves filtering irrelevant results, correcting typos, narrowing down answers by intersecting multiple results, and using GPT for further refinement. If necessary, we use human inference to finalize the answer.

2 Method

2.1 Task Definition

The goal of this task is to develop a model capable of retrieving relevant legal references for a given query. Each query Q is uniquely associated with the following components:

- **ID** (ID): A unique identifier for the query.
- **Title** (T): A concise summary of the query.
- **Question** (q): A detailed description or elaboration of the query.

The distinction between T and q is as follows:

- T provides a high-level overview or categorization of the query.
- q elaborates on T with specific details, scenarios, or clarifications.

Given T and q as inputs, the task is to generate a set of relevant legal references

$\{P_1, P_2, \ldots, P_n\}$, where each P_i corresponds to a specific legal clause or statute. The generated predictions $\{P_i\}$ should align with the content and intent of the query.

The output should be:

- Precise and relevant legal references.
- Filtered to include only valid legal clause identifiers from the provided law list.

2.2 Data Preprocess

2.2.1 Law

The provided legal documents are initially in `.rtf` format, with contents distributed across multiple files, making it challenging to map them to independent pairs of law index numbers and their clauses. To address this issue, we first convert the `.rtf` files to `.txt` format and parse each law article using the following steps:

1. **Law Name**: Extracted from the `.rtf` filename.
2. **Law Articles**: Use regex to identify the pattern " 第 XX 條" and extract the content between each occurrence.
3. **Filtering**: Exclude the law whose content marked as "(刪除)".
4. **Double-Checking**: Verify the context for any irrelevant suffixes related to other chapters, sections, or subsections, such as " 第 X 章", " 第 X 節", etc.

After preprocessing, the legal content is stored in a JSON file. Each entry includes an "id" representing the law name and index, and a "context" containing the corresponding clause and statute.

2.2.2 Data

Both the training and test datasets consist of a query containing a title and a question. The title generally serves as a summary, while the question provides a detailed description. For simplicity, these two components are concatenated to form a single input.

In the case of the training data, we randomly shuffle the dataset and create a training-validation split. The traditional model uses a 90:10 ratio for the split, while the Llama model treats the first 1,000 queries as the training data and the remaining 175 queries as validation data, which is used to evaluate performance.

2.2.3 ChatGPT Rewrite

We have observed that user queries and the legal context often differ in phrasing style. User queries are typically phrased in a more casual tone, while the legal context is presented in a formal, classical style, which may include terminology that is difficult for some individuals to understand. To address this, we utilize ChatGPT's context understanding and rewriting capabilities to align both the user queries and the legal context to the same speaking tone.

We set the following prompts in ChatGPT:

1. law context:

   你是一位法律專家，能夠解釋各種法律條文。請簡單的解釋這項法律：< 法條內容 >。(限一百字內)

2. user query:

   你是一位法律專家，能夠針對各種法律問題提出建議。請簡單的重新說明以下問題：<title+question>。(限一百字內)

This process generates two sets of data for training our model: one consisting of the original dataset from Kaggle and the other consisting of the rewritten data from ChatGPT. We will discuss the performance of both datasets in Section 3.2.

2.3 Traditional Architecture Method

Inspired by the work of BSARD [1], we applied the traditional model using the BiEncoder architecture. This approach maps the query and document into dense vector representations and calculates the similarity scores between the two vectors to retrieve relevant documents.

Since this approach involves one-to-one pair training, we perform an additional preprocessing step to transform the training dataset into a one-query-to-one-law-article mapping. As a result, multiple law contexts are mapped to the same query in the training data. The encoder also splits the context into overlapping chunks to address the issue of long contexts. After preprocessing, we train the BiEncoder model with the following parameters:

- Architecture: Siamese
- Pretrained model: `hfl/chinese-roberta-wwm-ext` [2]
- Similarity function: Cosine similarity
- Epochs: 100
- Batch size: 16
- Chunk size: 200
- Window size: 20

The trained BiEncoder model is then used to infer the test data. We load the trained model and encode both the test queries and law contexts into dense vectors for subsequent calculation of similarity scores and retrieval. Using the encoded embeddings, we perform retrieval with the `sentence_transformers.util.semantic_search` function to obtain the top relevant documents.

2.4 LLM Method

Given the strong natural language understanding capabilities, we experimented with the retrieval task using Llama-3-Taiwan [3] [4], which is capable of interpreting Mandarin and has comprehensive knowledge of Taiwan's legal articles. This approach encodes both the instruction and user query, allowing the model to generate a response based on its own knowledge using text-generation techniques.

The training workflow is visualized in Figure1.

We first converted the training data into the Alpaca format, including three fields:

- `id`: A unique identifier.
- `instruction`: A combination of title and question.
- `output`: The corresponding label.

Next, we submitted the instruction to the LLM as follows:

> 作爲法律達人，你的任務是告訴我可以從臺灣的哪個法條得到相關問題的答案。
> ### 問題：
> <instruction>
> ### 回應：你可以參考

Subsequently, we fine-tuned the model using the QLoRA framework [5] with the following parameters:

- **Model**: `yentinglin/Llama-3-Taiwan-8B-Instruct`.
- **Dataset**: 1000 queries from `train_data.json` in Alpaca format.
- **Learning Rate**: 2e−4.
- **Precision**: 4-bit quantization with BF16 support.
- **Batching**: Gradient accumulation with a per-device batch size of 4.
- **Training Steps**: 200 maximum steps, with checkpoints saved every 20 steps.
- **Scheduler**: Constant learning rate scheduler.

After the training process, we performed inference on the validation data across different checkpoints of the fine-tuned model to obtain the generated answers. We selected the checkpoint with the highest F1 score as the final model for testing data. To ensure the correctness of the generated answers from the LLM, we also implemented a **voting strategy** based on multiple inference runs to increase confidence in the results.

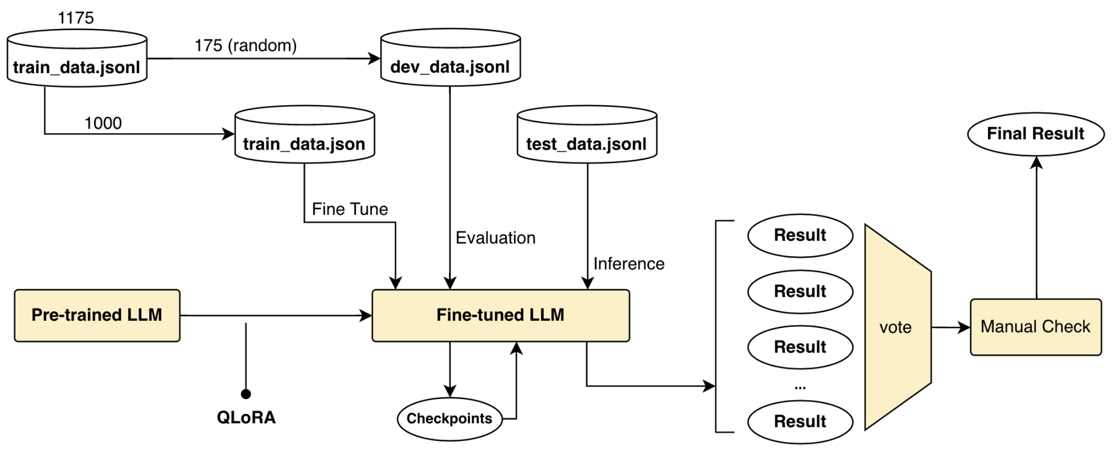

Figure 1: LLM Method Flow Chart

2.5 Postprocess

Once the results are retrieved from either the BiEncoder or the LLM, they require further processing to refine the answer to the query. We propose several postprocessing techniques and apply them to enhance the quality of the answers:

1. **Direct filter** (Match provided law index):
   Perform strict string comparison with all the provided law names and exclude any that do not match.
2. **Special Case Filter** (Match provided law index):
   Some outputs contained typos or slight variations, so we added an additional step to check for common substrings and replace common mistakes.
3. **Intersection of Generated Results** (Narrow down):
   We use the intersection of answers from multiple generated results to narrow down the potential answers and increase the confidence in the predictions.
4. **GPT check** (Narrow down):
   For answers containing more than **5 law references**, we feed the generated answers into ChatGPT to retrieve more relevant law contexts, narrowing down the results further.
5. **Human Inference**:
   If the filtered or narrowed down answers result in `NULL`, we manually review the raw-generated answer and provide a manual correction.

3 Experiment & Discussion

Each ablation and feature will be evaluated using precision, recall, and F1 score. For the traditional model, the entire training dataset will be used for evaluation, as the validation split was performed during the training stage, and the specific validation IDs were not recorded. For Llama, we will directly use the 175 queries from the validation split as the evaluation data.

3.1 Traditional Method Top-K selection

To determine the optimal number of retrieved documents, we conducted two analyses. The first analyzed the number of retrieved law contexts in the training data 1, and the second assessed the overall precision, recall, and F1 score for different top-K retrieval results 2. From the results, we observed that the median number of retrieved laws is 2, and the top-K = 2 retrieval resulted in the highest F1 score. Therefore, we selected top-K = 2 as the final retrieval configuration for the traditional model.

	Number of Retrieved law
min	1
max	18
mean	3.24
Q1	1
median	2
Q3	4

Table 1: Number of retrieved law

Top-K	Precision	Recall	F1
1	0.7645	0.3897	0.47070
2	0.6194	0.5467	**0.5224**
3	0.5045	0.6235	0.5009
4	0.4332	0.6792	0.4761
5	0.3764	0.7170	0.4460

Table 2: Precision, Recall, F1 score on Different Top-K

3.2 Different Set of Data for training

As mentioned in Section 2.2.3, we generated an additional set of data using ChatGPT for rewriting. We also used this set as training material to compare it with the original dataset. The qualitative results are presented in Table 3 and Table 4.

We observed that the ChatGPT-rewritten data performs better with the BiEncoder approach, while the original data yields better performance with the LLM approach. Our hypothesis is that the BiEncoder model computes the similarity between two embeddings, and sentences generated from the same source may help the model learn better alignment, minimizing ambiguous terms. On the other hand, the LLM performs better with the original data because it is pretrained to determine answers based on its own knowledge, and rewriting the context or adding additional explanations may confuse or mislead the model. Based on this observation, we decided to use the rewritten data for training the BiEncoder model and the original data for training the LLM model.

Dataset	Precision	Recall	F1
original	0.6194	0.5467	0.5224
rewrite query and original law context	0.6211	0.5503	0.5245
original query and rewrite law context	0.5025	0.4380	0.4193
rewrite query and rewrite law context	0.6284	0.5548	**0.5302**

Table 3: Precision, Recall, F1 score on Different Dataset in Traditional Model

Dataset and Model	Precision	Recall	F1
original	0.2758	0.2622	**0.2688**
rewrite query	0.2652	0.24347	0.2539

Table 4: Precision, Recall, F1 score on Different Dataset in Llama

3.3 Voting Strategy

We analyzed the predictions from 20 validation sets, examining the impact of merging different numbers of files (vertical axis) and varying voting thresholds (horizontal axis) on performance. The results in Table5 and Table6 indicate that merging more than 7 files and applying around 3 to 5 votes yields better prediction performance.

	Vote 1	Vote 2	Vote 3	Vote 4	Vote 5	Vote 6	Vote 7	Vote 8	Vote 9	Vote 10
1 file(s)	0.276	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2 file(s)	0.273	0.217	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3 file(s)	0.269	0.255	0.193	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4 file(s)	0.255	0.289	0.24	0.165	0.0	0.0	0.0	0.0	0.0	0.0
5 file(s)	0.245	0.297	0.261	0.219	0.152	0.0	0.0	0.0	0.0	0.0
6 file(s)	0.242	0.293	0.292	0.237	0.203	0.139	0.0	0.0	0.0	0.0
7 file(s)	0.235	**0.302**	**0.303**	0.265	0.232	0.191	0.138	0.0	0.0	0.0
8 file(s)	0.231	0.296	**0.309**	0.288	0.244	0.225	0.182	0.136	0.0	0.0
9 file(s)	0.226	0.293	**0.314**	**0.31**	0.264	0.246	0.217	0.167	0.129	0.0
10 file(s)	0.221	0.289	**0.306**	**0.309**	0.282	0.258	0.237	0.214	0.164	0.12
11 file(s)	0.218	0.279	**0.306**	**0.308**	0.293	0.272	0.238	0.236	0.211	0.16
12 file(s)	0.214	0.272	0.298	**0.308**	0.297	0.288	0.262	0.24	0.231	0.198
13 file(s)	0.21	0.269	0.293	**0.304**	**0.307**	0.285	0.268	0.26	0.238	0.217
14 file(s)	0.206	0.267	0.285	0.299	**0.3**	0.29	0.276	0.271	0.252	0.227
15 file(s)	0.204	0.263	0.284	0.295	**0.302**	0.298	0.282	0.278	0.258	0.25
16 file(s)	0.203	0.26	0.281	0.297	**0.305**	0.297	0.288	0.285	0.261	0.255
17 file(s)	0.197	0.26	0.282	0.289	**0.303**	**0.3**	0.287	0.285	0.261	0.259
18 file(s)	0.195	0.255	0.281	0.284	**0.304**	0.298	0.287	0.284	0.283	0.255
19 file(s)	0.195	0.251	0.282	0.284	0.298	**0.3**	0.295	0.287	0.285	0.285
20 file(s)	0.193	0.245	0.281	0.285	0.296	0.299	0.298	0.294	0.29	0.289

Table 5: Performance Comparison of File Count and Votes (vote 1-10)

	Vote 11	Vote 12	Vote 13	Vote 14	Vote 15	Vote 16	Vote 17	Vote 18	Vote 19	Vote 20
11 file(s)	0.115	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12 file(s)	0.15	0.111	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13 file(s)	0.188	0.148	0.111	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14 file(s)	0.217	0.175	0.139	0.113	0.0	0.0	0.0	0.0	0.0	0.0
15 file(s)	0.222	0.211	0.172	0.138	0.113	0.0	0.0	0.0	0.0	0.0
16 file(s)	0.242	0.222	0.207	0.17	0.135	0.113	0.0	0.0	0.0	0.0
17 file(s)	0.245	0.232	0.223	0.192	0.168	0.136	0.113	0.0	0.0	0.0
18 file(s)	0.244	0.241	0.224	0.2	0.19	0.155	0.133	0.113	0.0	0.0
19 file(s)	0.254	0.239	0.235	0.221	0.199	0.187	0.154	0.133	0.113	0.0
20 file(s)	0.268	0.251	0.239	0.233	0.214	0.2	0.17	0.154	0.132	0.113

Table 6: Performance Comparison of File Count and Votes (vote 11-20)

3.4 Postprocess strategy

To refine the retrieved answers generated by the BiEncoder and LLM, we experimented with the proposed approaches detailed in Section 2.5. The performance of these methods is summarized in Table 7.

The results indicate that the F1 score improves consistently across all postprocessing strategies. Notably, the improvement is more pronounced when applying the narrowing-down approach. Furthermore, incorporating an additional LLM for double-checking the answers has proven to be an effective method for enhancing performance.

Dataset and Model	public score	private score
original	0.18159	0.16409
direct filter	0.19248	0.23407
special case filter	0.19248	0.23407
intersection of generated results	0.21088	0.25742
GPT check	**0.38853**	**0.25925**

Table 7: Kaggle score on Different Postprocess approach

3.5 Training Results

3.5.1 Traditional

Figures 2 and 3 show the training loss and F1 score of the BiEncoder, using the ChatGPT-rewritten data. From the figures, we can observe that the training loss converges around 0.2 after 20 epochs, and the validation F1 score reaches its highest value (0.1638) at epoch 88. We used this checkpoint as the model for the traditional approach to evaluate other features.

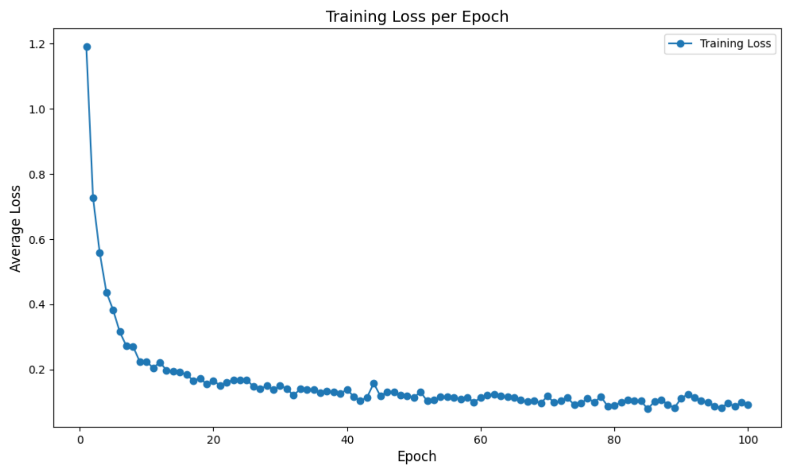

Figure 2: BiEncoder Training loss

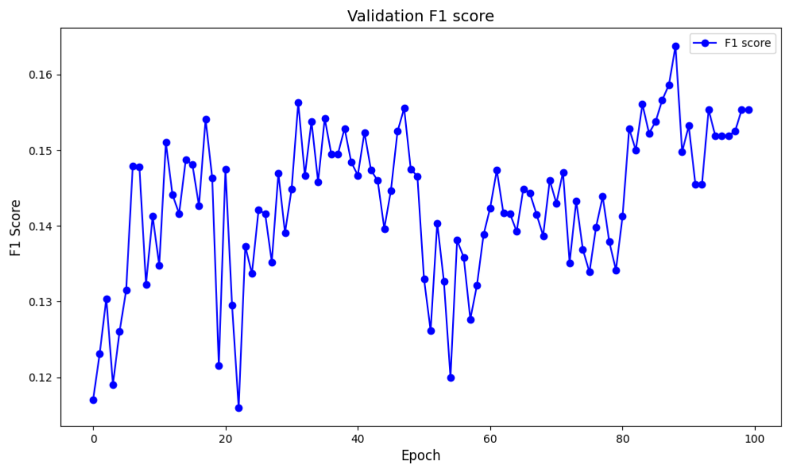

Figure 3: BiEncoder Validation F1 score

3.5.2 LLM

Figure 4 illustrates the training loss curve obtained during the fine-tuning of `Taiwan-LLaMa`, where checkpoints were saved every 20 steps, resulting in a total of 300 checkpoints and approximately 10.67 epochs on the training data. The results show that as the number of epochs increases, the training loss steadily decreases and eventually stabilizes around 0.02. This suggests that the model's performance on the training set improves over time.

To evaluate the training effectiveness and assess whether overfitting occurs, we used the validation set `dev_data.jsonl` to measure the model's Precision, Recall, and F1 scores at each checkpoint. As shown in Figure 5, the F1 score aligns with the task requirements. Therefore, we selected checkpoint-105, which achieved the highest F1 score, as the final model for further evaluation.

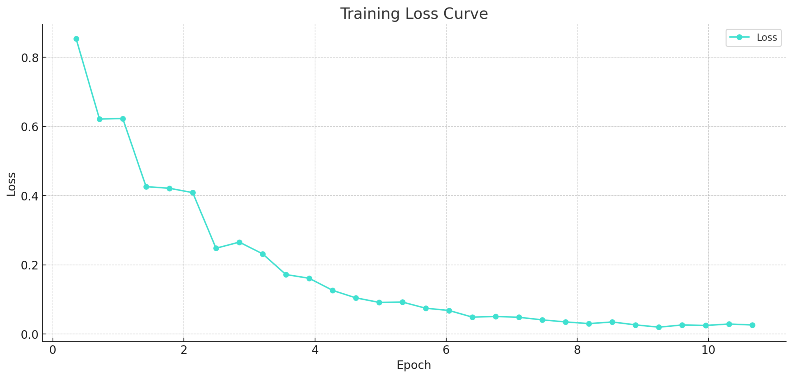

Figure 4: LLM Training Loss Curve

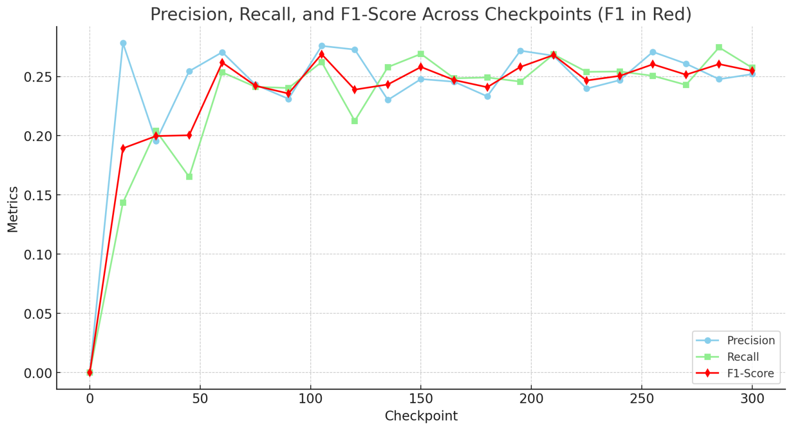

Figure 5: LLM Results across checkpoints

3.5.3 Overall

Based on the experiments and findings presented above, we submitted our retrieved document answers to the Kaggle competition. The results are shown in Table 8. The fine-tuned LLM, combined with the voting strategy and ChatGPT postprocessing, achieved the best public score among all the approaches. In contrast, the performance of the traditional method lags significantly behind the LLM approach. We believe this is due to the rigidity of the BiEncoder architecture, which cannot effectively connect terms into synonyms or related contexts to infer the most relevant law articles from the query. On the other hand, for the LLM approach, addressing the issue of too many candidate answers and using a voting strategy to increase confidence in the predicted answer proved to be an effective approach that boosted the public score.

Method	Public Score	Private Score
BiEncoder (zero shot)	0.01628	0.01027
BiEncoder (original data)	0.13680	0.15753
BiEncoder (rewrite data)	0.13355	0.19520
LLM (zero shot)	0.05732	0.07491
LLM (train 200 steps)	0.19248	0.23407
LLM (vote)	0.28125	0.23728
LLM (vote+intersect)	0.20779	**0.26119**
LLM (vote+ChatGPT)	**0.38853**	0.25925

Table 8: Precision, Recall, F1 score on Different Postprocess approach

4 Conclusion

The performance of LLMs in the experimental data was notably better, primarily because they effectively handle semantic-related issues. This capability is especially valuable when bridging the gap between informal language queries and formal language documents.

However, LLMs may generate an excessive number of candidate answers. To address this, manual intervention or the use of additional tools is required to filter out irrelevant answers. Therefore, we attempted to integrate a recent high-performing retrieval model—ChatGPT.

Despite a significant gap between public and private scores caused by overfitting on the public dataset, the experimental results indicate that combining different LLMs holds considerable promise for future development.

5 Contribution

陳思如	data preprocesss, traditional model method
金京	LLM method, voting and postprocess strategy
游善喆	report organizing and writing

References

[1] Antoine Louis and Gerasimos Spanakis. A statutory article retrieval dataset in french. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 6789–6803, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[2] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*, 2019.

[3] Yen-Ting Lin and Yun-Nung Chen. Taiwan LLM: bridging the linguistic divide with a culturally aligned language model. *CoRR*, abs/2311.17487, 2023.

[4] Po-Heng Chen, Sijia Cheng, Wei-Lin Chen, Yen-Ting Lin, and Yun-Nung Chen. Measuring taiwanese mandarin language understanding. *CoRR*, abs/2403.20180, 2024.

[5] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.