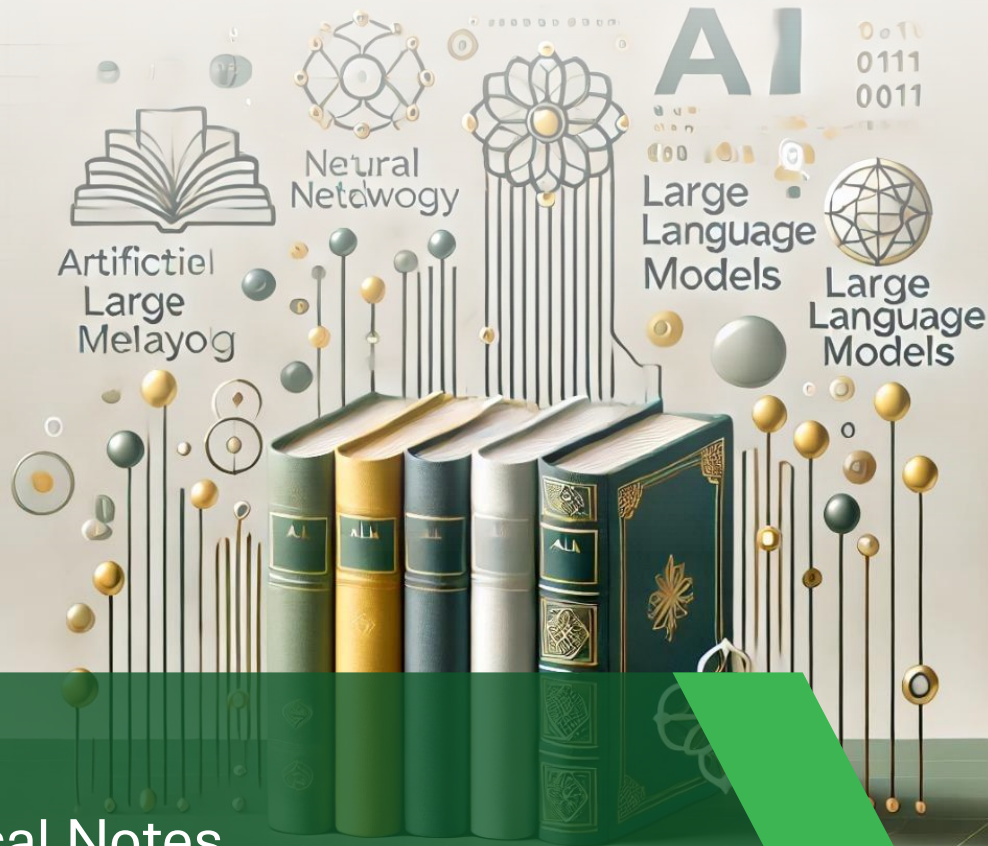


Bahasa Melayu



Technical Notes

LLM-BMR

2024-10-29

PREPARED BY:

Vince Ng, Balraj Pannu and Wan M Hasni Wan Sulaiman

DWH Mesra AI | DWH Technology

DWH Mesra AI Sdn. Bhd., DWH Technology Suite 19-12 G Tower Kuala Lumpur, 50450 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur.

 www.dwh-international.com

Table of Content

1. Introduction.....	2
2. Defining the Tasks of the LLM-BMR	2
2.1 Core Tasks	2
3. Perform Base LLM Selection	3
4. Preparations	4
4.1 Curated and Verified Corpus for Bahasa Melayu	4
4.2 Data Augmentation and Harmonization	4
4.3 Tokenization Approach and Methodology.....	5
5. Model Building via Fine-Tuning Strategy	5
6. Evaluation and Benchmarking	7
6.1 Evaluation Using Quantitative Metrics	7
6.2 Qualitative Evaluation.....	8
6.3 Benchmarking the Models.....	9
7. Computing Requirements	10
7.1 Computing Resources	10
7.2 Estimated Compute Costs	11
8. Talent Resources	12
9. Fast-Track Timeline	14
10. Summary.....	15

1. Introduction

This paper presents a technical discussion and notes on the development of a Large Language Model (LLM) specifically designed for Bahasa Melayu Rasmi (BMR), which is called as “LLM Bahasa Melayu Rasmi” or LLM-BMR project. The primary focus of the LLM-BMR is on textual Bahasa Melayu, meaning the modality of the LLM is centered exclusively on text-based processing. At this stage, we will limit our scope to textual data, avoiding other modalities such as audio or image processing.





The paper begins with outlining the targeted tasks for the LLM, and followed by the detailed steps involved in building the LLM-BMR, including the dataset preparation, model training, evaluation metrics, and fine-tuning techniques specific to each task. Then we will include the estimated resources required for the exercise and proposed timeline for the project completion.

This paper needs to be read in conjunction with another paper titled *Kertas Kerja LLM Bahasa Melayu Rasmi*.

2. Defining the Tasks of the LLM-BMR

The development of the LLM-BMR requires careful task definition to ensure the model is optimized for specific use cases in Bahasa Melayu.

2.1 Core Tasks

 Text Generation	 Text Summarization	 Text Translation	 Question Answering
<p>Generate coherent and contextually appropriate text in Bahasa Melayu.</p> <p>The length of the generated text must be specified based on the use case (e.g., <i>short-form responses vs. long-form content</i>).</p>	<p>Condense larger bodies of text into concise summaries, preserving the essential information.</p> <p>It is important to define the length of the context and the level of compression required for various applications.</p>	<p>Support translation tasks between Bahasa Melayu and major global languages. This includes two-way translation: <i>from major languages to Bahasa Melayu and vice-versa</i>.</p> <p>High-quality translation is crucial for cross-language communication and resource accessibility.</p>	<p>Handle question answering based on specific sets of Q&A domains.</p> <p>This task involves providing accurate and contextually relevant answers to questions posed in Bahasa Melayu, leveraging a predefined corpus of domain-specific knowledge.</p>

2.2 Additional Potential Tasks

In addition to the core tasks, the LLM can be extended to handle the following advanced capabilities:

Named-Entity Recognition (NER)	Grammar Correction	Creative Writing	Report Generation
<p>Involves identifying and classifying proper nouns (such as names of people, organizations, locations, etc.) within a given text.</p> <p>NER is essential for tasks like information retrieval and extraction in Bahasa Melayu.</p>	<p>Can perform grammar corrections on input text, ensuring that the output is grammatically accurate.</p> <p>The length of the text and the complexity of the grammar context must be defined to maintain efficiency and accuracy.</p>	<p>Can assist in generating creative content, such as stories or articles, in Bahasa Melayu.</p> <p>Here, the length and context limits need to be clearly set to guide the creative process while maintaining coherence.</p>	<p>Involves generating structured reports based on specific contexts or domains.</p> <p>The LLM would require input constraints and guidelines to ensure the generated reports meet the relevant criteria for content, format, and length.</p>

3. Perform Base LLM Selection

First we need to select candidates of base LLM; for this purpose it's crucial to consider multiple factors. Below are the key criteria that will guide the selection process:

- Multilingual Performance:** The model must demonstrate strong performance across multiple languages, especially in Bahasa Melayu. This ensures that the LLM is well-suited for tasks in a multilingual environment and can generalize effectively in Bahasa Melayu. The model should have been evaluated on tasks like translation, text generation, and understanding in various languages, including low-resource languages like Bahasa Melayu.
- Task Benchmarking:** The model must perform exceptionally well on standard NLP tasks such as text generation, summarization, translation, and question answering, particularly in its foundational language (which is typically English). These benchmarks will serve as a comparative measure to understand how well the model can be fine-tuned for specific tasks in Bahasa Melayu.
- Fine-Tuneability:** The LLM should have a proven architecture that supports transfer learning and fine-tuning techniques. This allows the model to be adapted efficiently to new tasks or languages, particularly for custom use cases involving Bahasa Melayu. The architecture should allow fine-tuning with task-specific and language-specific data while maintaining high performance.
- Model Size:** The model's size should be considered based on the balance between the number of parameters and context length performance. A larger model often performs better but requires more computational resources. The goal is to select a model size that offers optimal performance for a variety of tasks while being feasible to deploy and fine-tune.
- Model Architecture:** The architecture of the model plays a critical role in determining its suitability for different NLP tasks. There are three main types of architectures:

Decoder-based	Encoder-based	Encoder-decoder-based
Primarily used for text generation tasks	Suited for text understanding and classification tasks	A hybrid approach ideal for tasks that require both text understanding and generation, such as translation or summarization

6. **Trained Data:** The model's training data is a key factor, particularly its exposure to Bahasa Melayu. The selection process should consider models that have been trained on diverse datasets with good representation of Bahasa Melayu. Understanding the source and quality of the data on which the model was trained will help assess how well it is likely to perform on Bahasa Melayu specific tasks.

4. Preparations

4.1 Curated and Verified Corpus for Bahasa Melayu

The first part of preparation is the corpus to be used in model training. This corpus must be properly curated and verified. Preparation of the corpus entails:

- **Cleaning the data:** The corpus must undergo thorough cleaning to ensure the data is relevant and high quality. This includes the removal of non-Bahasa Melayu text, elimination of duplicates, filtering out special characters, and addressing any other noise that could hinder model training.
- **Achieving sufficient size:** The dataset must be sufficiently large to support the capacity of the models chosen in the first step. Larger models require more data for optimal performance, so the size of the corpus should scale with the model size to ensure effective learning and generalization.
- **Ensuring sufficient entropy:** The dataset should exhibit high diversity in both the sources and the structure of the texts. This ensures the model can generalize well across different contexts and use cases. In particular, the corpus should be domain-driven, incorporating a broad range of text sources to capture various nuances of Bahasa Melayu. Examples of domains to include are:
 - Official usages (e.g., *government documents, formal writings*)
 - Colloquial usages (e.g., *informal conversations, social media language*)
 - News articles (e.g., *journalism, reports*)
 - Social media (e.g., *Twitter, Facebook posts*)
 - Blogs (e.g., *personal and opinion blogs*)
 - Malay Wikipedia (e.g., *encyclopedic knowledge*)
 - Professional usages (e.g., *legal, medical, technical texts*)

4.2 Data Augmentation and Harmonization

Bahasa Melayu is considered a low-resource language, which poses challenges in assembling a large enough corpus for effective model training. To solve this, we need to deploy strategies to increase the dataset size and variability. Techniques like back-translation, paraphrasing, and text expansion can be used to artificially augment the corpus without introducing noise. Here, we need to deploy the following techniques:

- **Synthetic data creation:** Where possible, create synthetic data that mimics real-world usage of Bahasa Melayu. This can involve using smaller datasets to generate new samples via well known larger pre-trained models (*such as GPT-4, Llama3.1 70B, etc*).
- **Data harmonization and balancing:** The corpus should be carefully balanced across its various domains to ensure that no single type of usage dominates the model's training. A target set of balance between formal and informal data, professional and colloquial contexts, and diverse text structures must be established to ensure broad generalization.

4.3 Tokenization Approach and Methodology

The next major task after the creation of the corpus with data augmentation, is to create the tokenizer. The tokenizer will be used in the model fine tuning and training. The process of tokenizer entails the following:

- **Training and creation of token library:** Based on the curated corpus, a token library should be created that captures the vocabulary and linguistic structure of Bahasa Melayu. The tokenization process should reflect the language's unique syntax and grammar while minimizing the vocabulary size to maintain computational efficiency.
- **Amalgamation with pre-existing tokenizers:** If the chosen model already deploys a tokenizer (from Step 1), it may need to be adjusted or amalgamated with the newly generated token library from the corpus. This process ensures the tokenizer can handle Bahasa Melayu-specific tokens without degrading the model's performance.
- **Token optimization strategy:** An appropriate token optimization strategy should be deployed to fine-tune the tokenizer's performance. This may include:
 - Subword tokenization to capture root words and affixes common in Bahasa Melayu.
 - Adjusting token size and granularity to ensure the model can process both short, common words and longer, more complex terms.
 - Ensuring efficient tokenization to maintain a balance between context length and vocabulary richness without overwhelming the model's capacity.

5. Model Building via Fine-Tuning Strategy

The process of building a high-performing LLM for Bahasa Melayu involves applying a series of fine-tuning strategies to optimize the chosen base model candidates. Below is an outline of the steps required:

1. Train the chosen base LLM candidates:

- **Fine-tune the base model:** Using the curated Bahasa Melayu corpus, fine-tune the selected base model(s) to specialize the model's outputs for tasks in Bahasa Melayu.
- **Transfer learning approach:** Leverage transfer learning to adapt the pre-trained model to the specific nuances of the Bahasa Melayu language. This ensures that the model retains its knowledge of general language constructs while becoming proficient in handling Bahasa Melayu specific language structures.

2. Parameter-Efficient Fine-Tuning (PEFT)

PEFT techniques are essential for handling low-resource languages like Bahasa Melayu and for small-sized datasets, as they reduce the computational burden during both training and inference.

a. Model architecture adjustments:

- **Adapter layers:** Insert adapter layers into the model architecture to allow for task-specific fine-tuning without modifying the entire model's parameters. Adapter layers allow selective tuning, making the process more resource-efficient.
- **Rank adaptations of LoRA (Low Rank Adaptation) and QLoRA (Quantized Low Rank Adaptation):** Use LoRA and QLoRA strategies to inject low-rank adaptations into the model's layers, enabling efficient fine-tuning by focusing on fewer parameters. QLoRA quantizes these low-rank adaptations to further optimize memory usage, making it suitable for resource-constrained environments.
- **Hyperparameter optimization:** Experiment with and optimize key hyperparameters such as learning rate, batch size, and the number of epochs to ensure the most efficient and effective training cycle. Finding the optimal balance is critical for resource-efficient fine-tuning and for achieving the best possible model performance.
- **Layer freezing and firing cycles:** Implement layer freezing to limit updates to only certain parts of the model while keeping the rest unchanged. Layer firing (or gradual unfreezing) can then be used to progressively unlock deeper layers for fine-tuning as training progresses. This helps balance computational efficiency with model performance.

b. Prefix-Tuning for Task-Specific Fine-Tuning:

- Apply prefix-tuning to adapt the model for specific tasks without retraining the entire model. In prefix-tuning, a set of learnable prefixes is added to the input sequence, allowing task-specific adaptations while keeping the core model frozen.
- Target tasks (as defined earlier) include:
 - Text Generation: Generate coherent text in Bahasa Melayu.
 - Summarization: Summarize longer texts effectively while maintaining key information.
 - Translation: Translate between Bahasa Melayu and other languages.
 - Question Answering: Provide accurate answers based on context in Bahasa Melayu.
 - Others as defined.

c. Model tuning via Reinforcement Learning with Human Feedback (RLHF):

In the final stages of fine-tuning, Reinforcement Learning with Human Feedback (RLHF) is employed to align the model outputs more closely with human experts.

- Establish a human feedback framework:
 - Feedback collection: Set up a process for collecting feedback from human annotators on the model's outputs. Annotators evaluate model outputs based on predefined criteria (e.g., fluency, accuracy, relevance).

- Ranking mechanism: Implement a ranking methodology where annotators rank multiple outputs from best to worst or assign quality scores. These rankings are used to train the reward model.
- Reward model creation: Build a reward model that uses the ranking data to predict the quality of model outputs. This model provides a feedback loop for further tuning.
- Establish Reinforcement Learning (RL) framework:
 - Train the reward model: Typically, a distilled version of the main model is trained as the reward model to evaluate the outputs of the fine-tuned LLM.
 - Apply Proximal Policy Optimization (PPO): Use the PPO algorithm to fine-tune the model based on feedback from the reward model. PPO ensures that the model's outputs align more closely with human preferences by updating the policy in a stable, incremental manner.
 - Iterative feedback loop: Implement an iterative process of generating outputs, collecting human feedback, and further refining the model based on the reward signals. This cycle continues until the model achieves the desired performance level.

6. Evaluation and Benchmarking

Evaluating LLM involves both quantitative and qualitative assessments. Benchmarking involves comparing the model's performance against other models or a set of predefined standards.

6.1 Evaluation Using Quantitative Metrics

These are standard metrics used to evaluate LLM performance and are often used for comparison across models or iterations of the same model. Here's a breakdown of key evaluation metrics:

1. Perplexity:

- What it measures: Perplexity is a metric for evaluating language models generation tasks. It measures how well the model predicts the next word in a sequence.
- Lower is better: A lower perplexity score means the model is more confident in its predictions. This metric is especially important when evaluating the generative capabilities of models.
- Applicability: For tasks such as text generation or summarization, lower perplexity indicates better language modeling capabilities.

2. Accuracy:

- What it measures: Accuracy is important for classification tasks; it measures the percentage of correct predictions.
- Applicability: In tasks like sentiment analysis or text classification, accuracy can be a straightforward measure of model performance.

3. F1 Score (Harmonic Mean of Precision and Recall):

- What it measures: F1 score is the balance between precision (the fraction of relevant instances among the retrieved instances) and recall (the fraction of relevant instances that were retrieved).

- **Applicability:** For token-level tasks like Named Entity Recognition (NER) or question answering, particularly in classification models, F1 score gives a clearer picture of model performance beyond simple accuracy.
- **Use case:** This can be used for evaluating tasks like NER, where the language's specific named entities (places, organizations, etc.) need to be correctly recognized.

4. BLEU (Bilingual Evaluation Understudy) Score:

- **What it measures:** BLEU is commonly used to evaluate the quality of text generation models by comparing machine-generated text to a reference.
- **Applicability:** It is especially useful for translation tasks or summarization, like Bahasa Melayu-to-English translation or summarization in Bahasa Melayu of English texts.

5. ROUGE (Recall-Oriented Understudy for Gisting Evaluation):

- **What it measures:** ROUGE measures the overlap of n-grams between the generated text and a reference text. It is commonly used to evaluate summarization models.
- **Applicability:** Use ROUGE when testing the performance of summarization models in Bahasa Melayu. A high ROUGE score means the model is good at producing summaries that match human-generated ones in terms of content.

6. Human Ratings:

- **What it measures:** In RLHF setups, human annotators provide feedback through ratings or ranking. These ratings measure fluency, relevance, cultural appropriateness, and correctness in tasks like Bahasa Melayu dialogue generation or text completion.
- **Applicability:** This can be particularly useful for text generation models when producing outputs that need cultural alignment or nuanced understanding, such as creating conversational agents in the Bahasa Melayu.

6.2 Qualitative Evaluation

Quantitative metrics alone may not provide a complete picture of a model's performance, especially for nuanced tasks. Qualitative evaluations help ensure that the model meets human standards in terms of fluency, coherence, and appropriateness. This is particularly important where cultural context and linguistic subtleties need to be evaluated.

1. Human Evaluation:

- **What it measures:** Native BM speakers or subject-matter experts evaluate model outputs based on qualitative criteria such as grammatical correctness, fluency, and cultural appropriateness.
- **Applicability:** For text generation tasks (e.g., BM dialogue systems or creative writing), human evaluators can assess whether the generated text sounds natural and whether the model adheres to Malay cultural norms and idiomatic expressions.

2. Error Analysis:

- **What it measures:** In error analysis, human evaluators or model developers look at common patterns of errors made by the model. This could include mistranslations, incorrect sentiment predictions, or misclassification of entities.
- **Applicability:** Error analysis is especially useful for classification tasks or question answering in Malay, where you need to understand what kinds of mistakes the model is making and why.

3. Guardrailings and Bias Removals

- What it measures: Measures various guardrails developed by the developer, with particular emphasis on bias removal.
- Applicability: To ensure that all safety and reliability issues are adhered to.

6.3 Benchmarking the Models

Benchmarking involves comparing the model's performance against other models or a set of predefined standards. It helps assess how the fine-tuned model performs relative to other existing models or iterations.

Choosing a Benchmark Dataset

Selecting the right benchmark dataset is essential. We need datasets that are representative of the tasks we want to evaluate (e.g., classification, generation, summarization) and the language in question.

1. Official Bahasa Melayu Datasets:

- Using standard Bahasa Melayu datasets like official dictionaries or parallel corpora for translation tasks can help ensure that the model is being evaluated against an accepted benchmark.
- Localized datasets: For tasks like sentiment analysis or classification, we may need Bahasa Melayu specific datasets such as social media corpora (e.g., from Malaysian Twitter or forums) or government/public datasets.

2. Multilingual Benchmark Datasets:

- For tasks involving multiple languages, we can use multilingual benchmarks like XTREME or XGLUE, which contain tasks like translation, NER, and sentiment analysis in various languages, which includes Bahasa Melayu.
- FLoRes: For translation tasks, FLoRes from Meta.ai is an excellent multilingual dataset for benchmarking translation quality.

Comparison Among the Models

We will benchmark the various chosen models against each other as well as against other established foundation models. The following are the benchmarking approaches:

1. Against Each Other Among the Candidate Models:

Compare the fine-tuned candidate models with each other on the various tasks which was set in the designed tasks.

Use case:

Comparative metrics on task based such as text generation, summarization, etc.

2. Compare with Established Multilingual Foundation Models

Compare fine-tuned model's performance against large, pre-trained multilingual models like mT5, XLM-R, or Llama 3.1 70B.

Use case:

For example, in tasks like Bahasa Melayu-to-English translation, comparing performance with larger foundation models will highlight whether fine-

tuning improved specific Bahasa Melayu tasks.

3. Task-Specific Models:

Compare against models that have been specifically fine-tuned for similar tasks (e.g., sentiment analysis, question answering, text summarization).

Use case:

Benchmark it against other sentiment analysis models or even rule-based systems that handle Bahasa Melayu language.

Issues and Areas of Attention

1. Handling Domain-Specific Tasks:

Model is fine-tuned for general tasks instead of specific domains (e.g., legal, medical, or governmental Malay language), standard benchmarks may not fully reflect the specialized needs of the task.

Solution:

Create domain-specific benchmarks by gathering text corpora from relevant fields (e.g., legal documents, health reports) and evaluating the model's performance on these specialized datasets.

2. Cultural and Linguistic Biases

Evaluating a model fine-tuned on a specific language can reveal cultural or linguistic biases that may affect model performance (e.g., favoring certain dialects or forms of speech).

Solution:

Involve human evaluators familiar with different forms of the Malay language (formal, informal, dialects) to assess how well the model handles variations.

3. Generalization to Real-World Scenarios:

Fine-tuning a model on curated datasets can sometimes lead to overfitting or performance degradation when applied to real-world tasks (e.g., generating text or answering questions in less-structured environments).

Solution:

After benchmark evaluations, test the model in live environments (e.g., with real user queries or through user testing) to ensure it generalizes well to real-world use cases.

7. Computing Requirements

In this section, we will outline the computing resources requirement to perform the intended program for developing the model

7.1 Computing Resources

We plan to use Amazon Web Services (AWS) as the platform of choice. AWS is one of the most popular choices for ML workloads due to its wide range of services and scalable infrastructure.

More importantly, the familiarity of our engineering team in setting up and deployment on AWS, as well as latency to nearest AWS region, which is in Singapore.

We plan to deploy six base models, with each model undergoing five to six cycles of exercises involving the freezing and firing of various parts of the model architecture, we provide estimates of the cost and resource requirements accordingly. Here's a breakdown of what each cycle involves:

- Model Size: Assume we're working with mid-sized models (~7B to 13B parameters).
- Cycle Details:
 - Each cycle involves varying degrees of model training (freezing layers, fine-tuning certain parts), but with adjustments in model architecture for efficiency.
 - Since the model is not fully fine-tuned in every cycle (some parts are frozen), the compute required for each cycle should be lower than a full fine-tuning pass.

For optimal performance and scalability, we plan to use Amazon SageMaker and EC2 P4d Instances (with A100 GPUs). The resource configuration for each cycle are:

- EC2 P4d Instances: Using 4x A100 GPUs for each model fine-tuning cycle. This setup allows for high parallelism and efficient processing of the 7B-13B parameter models.
- Training time per cycle: Given that each cycle involves partial fine-tuning, it will involve lesser compute time per cycle.
- Storage: For each model, we estimate an amount of ~50GB of data storage per checkpoint and multiple checkpoints across the 6 cycles.

7.2 Estimated Compute Costs

Task	Time per Cycle (Hours)	Compute Cost per Hour (4x A100 GPUs)	Cost per Cycle	Total Cost per Model (6 Cycles)	Total Cost for 6 Models (6 Cycles)
Fine-Tuning	288 hours	~USD32/hour	~USD9,216	~USD55,296	~USD331,776
Reward Model	144 hours	~USD16/hour (2x A100 GPUs)	~USD2,304	~USD13,824	~USD82,944
PPO	216 hours	~USD32/hour (4x A100 GPUs)	~USD6,912	~USD41,472	~USD248,832
TOTAL COSTS				~USD663,552	

Note:

1. The cost does not include data storage costs. The storage costs remain fixed and negligible in the total costs computations.
2. The cost does not include deployment costs, which is a subject to be address post completion of the model development phase.

8. Talent Resources

In this section, we outline the talent resources and its requirements. There are three sets of talents involved, namely data scientists, AI/ML engineers and human annotators. We will cover only the data scientists and AI/ML engineers since the resources for human annotators is not defined yet, and will be defined after engagement with the subject matter experts.

Key Task for Data Scientist:

- Dataset Preparation: Cleaning, preprocessing, and augmenting the Bahasa Melayu corpus.
- Data annotation: Collaborating with annotators to handle the RLHF phase and ensuring that the dataset is properly labeled and verified.
- Data pipeline development: Building and maintaining the data pipeline for efficient processing and storage of the corpus.
- Evaluating model outputs: Analyzing and verifying model outputs for correctness, language fluency, and overall performance, especially during the RLHF phase.

Key Task for AI/ML Engineers:

- System architecture: Setting up and managing the cloud infrastructure (AWS) and ensuring that it is optimized for training cycles.
- Model fine-tuning: Executing and managing the fine-tuning of the selected LLMs, including PEFT, LoRA, QLoRA, Prefix-Tuning, etc.
- Reinforcement learning: Implementing the RLHF framework, including developing feedback loops, training reward models, and deploying PPO.
- Hyperparameter tuning: Optimizing learning rates, batch sizes, and epochs for all cycles.
- Model evaluation and benchmarking: Evaluating models on task-specific metrics and adjusting the model accordingly for task improvement.

8.1 Estimated Man-Hours for Various Tasks:

To estimate man-hours, we'll break the project down into stages: data preparation, model fine-tuning, RLHF implementation, and evaluation.

Data Preparation (Data Scientists)

Task	Estimated Man-Hours
Cleaning and preparing the Corpus	~80-120 man-hours
Data augmentation and synthetic data creation	~60-80 man-hours
Tokenization strategy development	~40-60 man-hours
Pipeline development for preprocessing	~40-60 man-hours
Collaborating with annotators for RLHF data	~50-70 man-hours
Total Estimated Man-Hours	270-390 man-hours

Fine-Tuning and RLHF (AI/ML Engineers)

Task	Estimated Man-Hours
Cloud infrastructure setup and management (AWS)	~40-60 man-hours
Fine-tuning the base models (6 models, 6 cycles)	~200-240 man-hours
Managing model freezing/firing	~50-80 man-hours
Implementing PEFT/LoRA for efficient fine-tuning	~80-100 man-hours
RLHF (Reward Model training, PPO)	~150-200 man-hours
Hyperparameter tuning for each cycle	~100-120 man-hours
Model evaluation and benchmarking	~120-150 man-hours
Total estimated Man-Hours (Fine-Tuning + RLHF)	740-950 n-hours

8.2 Evaluation and Final Adjustments:

This stage will involve both Data Scientists and AI/ML Engineers collaborating to evaluate the model outputs, perform error analysis, and adjust the model based on results.

Task	Estimated Man-Hours
Model output evaluation (Data Scientists)	~80-100 man-hours
Final model adjustments (AI/ML Engineers)	~60-80 man-hours
Total estimated man-hours (evaluation and adjustments)	~140-180 man-hours

8.3 Estimated Human Resources Costs

Task	Rate	Man-Hours	Total Cost
Data scientists	RM120/hour	350 - 490 hours	~RM42,000 - RM58,800
AI/ML engineers	RM150/hour	800 - 1,100 hours	~RM120,000 - RM165,000
Total talent cost			~RM162,000 - RM223,800

Note: The human resources does not include human annotators and human-feedback costs.

9. Fast-Track Timeline

In this section we provide an estimate of timeline for the project execution and deliveries.

Phase 1: Project Setup and Dataset Preparation (Weeks 1-2)

Week 1:

Project Kickoff & Cloud Infrastructure Setup

- **Task:** Set up AWS environment (SageMaker, EC2 instances, S3 buckets) for compute and data processing
- **Responsibility:** AI/ML Engineers

Week 2:

Dataset Preparation and Preprocessing

- **Task:** Collect, clean, and augment the Bahasa Melayu corpus
- **Responsibility:** Data Scientists

Phase 2: Model Fine-Tuning (Weeks 3-6)

Week 3:

Initial Fine-Tuning (Cycle 1)

- **Task:** Fine-tune the first cycle of all 6 models using freezing and firing strategies
- **Responsibility:** AI/ML Engineers

Week 4-5:

Fine-Tuning (Cycles 2 & 3)

- **Task:** Continue with cycles 2 and 3
- **Responsibility:** AI/ML Engineers

Week 6:

Final Fine-Tuning (Cycles 4)

- **Task:** Perform the final cycle of fine-tuning
- **Responsibility:** AI/ML Engineers

Phase 3: Reinforcement Learning with Human Feedback (Weeks 7-10)

Week 7:

Setup RLHF Framework

- **Task:** Establish the RLHF framework, including human feedback tools
- **Responsibility:** Data Scientists & AI/ML Engineers

Week 8-10:

Human Annotation and RLHF Model Training

- **Task:** Train the reward model and apply the PPO algorithm
- **Responsibility:** AI/ML Engineers

Note: We do not have a clear estimates of how much time requirement for the human annotators to perform their tasks. This depends on the actual number of annotators involved and tasking to be performed by different set of annotators.

Phase 4: Model Evaluation and Adjustments (Weeks 11-14)

Week 11-12:

Model Evaluation

→ **Task:** Perform model evaluation and benchmarking

→ **Responsibility:** Data Scientists & AI/ML Engineers

Week 13-14:

Final Adjustments

→ **Task:** Apply final adjustments and optimizations

→ **Responsibility:** AI/ML Engineers

Phase 5: Documentation and Completion (Week 15)

Week 15:

Project Completion

→ **Task:** Final documentation of the process and project wrap-up (codes cleaning, refactoring, and freezing the model)

→ **Responsibility:** Data Scientists & AI/ML Engineers

10. Summary

In this paper, we provide the approaches that we plan to undertake for the development of LLM-BMR project as envisaged. The undertakings involved the overall strategy, preparations of datasets, the general methodology for fine-tuning and finalizing the LLM model, and the resources required for the project. And finally the proposed timeline for the project to be completed on fast-track basis.