**1. Define Objectives**

* **Primary Goal**: Evaluate how different tokenization strategies impact the performance and efficiency of large language models (LLMs) in multilingual tasks.
* **Key Aspects**:
  + Quantify performance metrics (e.g., BLEU, perplexity, F1-score).
  + Measure efficiency trade-offs (model size, computational cost, inference speed).
  + Explore specific challenges like rare words and code-switching.

**2. Background Study**

**Related Work Review**

* Analyze **Sennrich et al. (2016)** for insights into handling rare words with subword units.
* Study **Kudo (2018)** to understand subword regularization and its effect on neural machine translation models.
* Review other literature to understand how tokenization impacts multilingual tasks and LLM efficiency.

**Theory and Concepts**

* Understand how **Byte Pair Encoding (BPE)**, **SentencePiece**, and **WordPiece** work, focusing on:
  + Vocabulary generation.
  + Handling of out-of-vocabulary (OOV) words.
  + Implications for model training and inference.

**3. Research Methodology**

**Tokenization Strategies**

* **Strategies to Compare**:
  + **Byte Pair Encoding (BPE)**.
  + **SentencePiece (Unigram/Regularization)**.
  + **WordPiece**.
* Optionally include newer or less-explored tokenization methods for comparison.

**Multilingual Datasets**

* **Dataset Selection Criteria**:
  + **Language Diversity**: Include high-resource (e.g., English, Chinese) and low-resource (e.g., Swahili, Urdu) languages.
  + Tasks:
    - **Translation**: Use datasets like WMT, OPUS, or Flores.
    - **Sentiment Analysis**: Use datasets like Multilingual Amazon Reviews or XNLI.
* Preprocess datasets to standardize input for each tokenization strategy.

**LLM Training/Fine-tuning**

* **Model Selection**:
  + Start with a pre-trained LLM (e.g., mBERT, XLM-R, or BLOOM) to reduce computational cost.
  + Fine-tune models for each tokenization strategy on the selected datasets.
* **Training Protocol**:
  + Keep hyperparameters constant across experiments.
  + Train/fine-tune models to convergence or for a fixed number of epochs.
  + Log metrics during training (e.g., loss, validation accuracy).

**4. Evaluation Metrics**

**Performance Metrics**

* Translation tasks:
  + **BLEU Score**: Measures translation quality.
  + **Perplexity**: Evaluates the fluency of language generation.
* Sentiment analysis:
  + **F1-Score**: Measures classification performance.
  + **Accuracy**: Simpler metric for classification tasks.

**Efficiency Metrics**

* **Model Size**: Measure vocabulary and parameter size for each tokenization method.
* **Computational Cost**: Track GPU hours, memory usage, and FLOPs during training and inference.
* **Inference Speed**: Measure tokenization and decoding speed.

**Qualitative Analysis**

* Examine how each tokenization strategy handles:
  + Rare or unknown words.
  + Code-switching.
  + Morphologically rich languages.

**5. Experimental Design**

**Baseline Experiment**

* Train/fine-tune a model using a default tokenization strategy (e.g., BPE).
* Use this as the baseline for comparison.

**Comparative Analysis**

* Train/fine-tune models using other tokenization strategies (SentencePiece, WordPiece).
* Compare performance and efficiency metrics against the baseline.

**Controlled Variables**

* Ensure identical training datasets, hyperparameters, and training settings across experiments.
* Standardize tokenization preprocessing (e.g., consistent sequence lengths).

**Exploratory Analysis**

* Evaluate the impact of tokenization on specific challenges:
  + Rare word handling: Assess how strategies tokenize and predict rare or OOV words.
  + Code-switching: Analyze performance on datasets with mixed-language inputs.

**6. Implementation Plan**

**Phase 1: Preparation**

1. Install required tools: Tokenizers library, Hugging Face Transformers, PyTorch/TF.
2. Gather multilingual datasets for translation and sentiment analysis.
3. Preprocess datasets for each tokenization method.

**Phase 2: Experiments**

1. Train/fine-tune baseline model with default tokenization.
2. Train/fine-tune models with alternate tokenization strategies.
3. Log training metrics and model checkpoints.

**Phase 3: Evaluation**

1. Evaluate models on translation tasks (BLEU, perplexity).
2. Evaluate models on sentiment tasks (F1-score, accuracy).
3. Measure efficiency metrics (model size, computational cost, inference speed).

**Phase 4: Analysis**

1. Compare performance across tokenization strategies.
2. Analyze qualitative results for rare words and code-switching.
3. Visualize trade-offs using charts (e.g., bar plots for metrics, line graphs for speed vs. accuracy).

**7. Tools and Resources**

* **Programming**: Python, PyTorch, TensorFlow.
* **Libraries**: Hugging Face Transformers, SentencePiece, Tokenizers.
* **Compute Resources**: Use local GPU (4080s) or rent cloud GPUs (AWS, Colab Pro).
* **Dataset Sources**: WMT, OPUS, XNLI, or any open multilingual datasets.

**8. Expected Deliverables**

1. **Models**: Fine-tuned LLMs for each tokenization strategy.
2. **Performance Reports**: Metrics for translation and sentiment tasks.
3. **Efficiency Analysis**: Reports on model size, computational cost, and inference speed.
4. **Qualitative Insights**: Case studies on rare words and code-switching.

**9. Timeline**

* **Week 1-2**: Literature review and dataset preparation.
* **Week 3**: Tokenization preprocessing and baseline model training.
* **Week 4-5**: Train/fine-tune models with alternate tokenization strategies.
* **Week 6**: Evaluate models and analyze results.
* **Week 7**: Draft the report and visualizations.
* **Week 8**: Finalize report and presentation.