Smart Human-centered Emerging Research in Machine Intelligence (SHEM)
Vol. 2, No. 2, October 2026, pp. 82−96
E-ISSN: 3163-7655 | P-ISSN: 3163-7639, DOI:10.68012

Designing a Modular Transformer-Based Framework for Low-Resource Language Text Summarization

Shofiyul Millah[1], Rio Squire Sean[2*], Kristina Vaher[3], Dwi Julianingsih[4]

[1]Department of Information System, Indonesian Lecturers Association, Indonesia
[2, 3]Department of Information System, iLearning Research, Estonia
[4]Department of Computer Science, iLearning Research, Indonesia
[1]shofiyul@raharja.info, [2]sean01@ilearning.ee, [3]vaher.kristin@ilearning.ee, [4]dwi.julianingsih@raharja.info
***Corresponding Author**

Article Info

Article history:

Submission July 18, 2026
Revised August 25, 2026
Accepted September 11, 2026
Published October 16, 2026

Keywords:

Modular Transformer Architecture
Low-Resource Language Processing
Text Summarization
Adapter-Based Fine-Tuning
Cross-Lingual Transfer Learning

ABSTRACT

Text summarization for low-resource languages remains constrained by limited training corpora and the computational demands of large transformer models. This paper presents a modular transformer-based framework designed to address both constraints through architectural decomposition rather than data expansion. The framework separates the summarization pipeline into three interchangeable modules: a lightweight multilingual encoder pretrained on cross-lingual embeddings, a compression-aware attention mechanism that reduces parameter overhead by sharing weights across decoder layers, and a fine-tuning adapter layer that allows rapid domain adaptation with minimal labeled data. Each module operates independently, allowing researchers to swap components based on the target language's available resources without retraining the entire model. The design prioritizes transfer learning from high-resource languages, using adapter-based fine-tuning to bridge the performance gap that typically affects low-resource settings. We detail the architectural specifications, including layer configurations, attention head allocation, and the adapter injection strategy, alongside a reference implementation in Python using standard deep learning libraries. Rather than reporting benchmark results from empirical data collection, this study focuses on the design rationale, computational efficiency projections, and scalability considerations for languages with sparse digital text resources. The discussion extends to the framework's potential impact: reduced infrastructure barriers for NLP research in underrepresented languages, faster deployment cycles for organizations working in multilingual contexts, and a contribution toward more equitable access to AI-driven language technology. The proposed architecture offers a practical blueprint for researchers and practitioners aiming to build summarization systems where labeled data and computational resources are scarce.

This is an open access article under the CC BY 4.0 license.

DOI: https://doi.org/10.68012/shem.v2i2.271
This is an open-access article under the CC-BY license (https://creativecommons.org/licenses/by/4.0/)
©Authors retain all copyrights

1 INTRODUCTION

The volume of digital text produced across news platforms, government portals, and social media has grown far beyond what any human reader can process manually, and automatic text summarization has become a practical necessity rather than a research curiosity [1]. Organizations working with Indonesian,

Nepali, Urdu, Vietnamese, and hundreds of other languages face the same demand for condensed, accurate summaries that English-speaking institutions solved years ago with large pretrained models [2]. The gap does not stem from a lack of interest in these languages. It stems from a lack of digital text, annotated corpora, and computational infrastructure available to researchers who work outside a handful of well-funded language communities. A model trained on English news articles cannot simply be pointed at a Javanese or Sundanese corpus and expected to produce coherent summaries, because the underlying vocabulary, syntax, and discourse patterns differ too much for direct transfer [3, 4]. This mismatch between where NLP research concentrates its resources and where the practical need for summarization actually exists forms the starting point for this study [5].

Researchers have already begun to close this gap using multilingual pretrained transformers as a foundation [6]. adapted mBERT and mT5 for Urdu summarization, building a dedicated dataset of over seventy-six thousand article-summary pairs and producing a compressed model that matched high-resource baselines on ROUGE and BERTScore metrics [7]. [8] took a similar path for Nepali, fine-tuning mBART and mT5 on a web-scraped news corpus and validating the output through both automated scoring and human evaluation. Outside the summarization space specifically, researchers have paired OCR pipelines with multilingual processing to extend text accessibility to Indic scripts, treating summarization as one stage in a longer accessibility pipeline rather than the sole objective [9]. Across these efforts, a consistent pattern emerges: each research group builds a new dataset, fine-tunes an existing large multilingual backbone end to end, and reports results specific to one language and one domain, usually news text [10].

That pattern reveals the gap this paper addresses. Every one of these approaches ties the summarization system to a full retraining cycle whenever a new language, domain, or resource constraint appears, which means the engineering effort invested in Urdu does not transfer to Nepali, and the effort invested in Nepali does not transfer to the next language a research team decides to tackle [11, 12]. Full fine-tuning of a multilingual backbone also demands GPU resources that many research groups in low-resource language communities simply do not have access to, which turns computational cost into a second barrier layered on top of the original data scarcity problem [13, 14]. Parameter-efficient fine-tuning techniques such as adapters have shown promise in the broader transfer learning literature, yet the summarization research reviewed above still favors full fine-tuning over adapter-based strategies, and no existing work packages the encoder, the attention mechanism, and the fine-tuning strategy as independently swappable components [15]. The field lacks a reusable architectural blueprint that a research team can adapt to a new language by swapping one module rather than rebuilding the entire pipeline [16, 17].

This paper proposes a modular transformer-based framework built around three independently replaceable components: a lightweight multilingual encoder that draws on existing cross-lingual embeddings rather than requiring language-specific pretraining from scratch, a compression-aware attention mechanism that shares weights across decoder layers to cut parameter count without discarding contextual capacity, and an adapter-based fine-tuning layer that lets a research team adapt the model to a new language or domain using a fraction of the labeled data that full fine-tuning demands [18]. The novelty of this design lies in the separation itself. Where prior work treats the summarization model as a single unit that must be retrained wholesale, this framework treats each stage as an interface that a researcher can swap, extend, or replace independently, so the compression mechanism developed for one language remains usable when the encoder or the adapter changes for the next [19, 20]. The framework does not report benchmark scores from empirical data collection, because the objective here is architectural: to specify a design that lowers the entry barrier for summarization research in languages that current approaches largely bypass.

The contribution of this paper follows directly from that objective. First, it specifies a modular architecture that decouples encoding, attention computation, and fine-tuning into three interchangeable stages, each documented with its own interface. Second, it details a compression-aware attention mechanism that reduces decoder parameter overhead through layer-wise weight sharing, with the reduction quantified against a standard transformer decoder configuration. Third, it presents an adapter injection strategy specifically suited to summarization, positioned as a lower-cost alternative to the full fine-tuning approach that dominates the reviewed literature. Fourth, it provides a reference implementation of the full framework and uses that implementation to project the computational and practical impact the design could have on summarization research for underrepresented languages, including reduced hardware requirements and faster iteration cycles for research teams without access to large-scale compute.

The remainder of this paper is organized as follows. Section 2 reviews the theoretical foundations un-

derlying transformer architectures, multilingual embeddings, and adapter-based transfer learning, establishing the concepts the proposed framework builds on. Section 3 presents the modular framework design in detail, covering the encoder module, the compression-aware attention mechanism, and the adapter fine-tuning layer, along with the rationale behind each design decision. Section 4 describes the reference implementation, including the source code structure, the libraries used, and the configuration choices that make the framework reproducible. Section 5 discusses the projected impact of the framework on low-resource NLP research, industry adoption, and the broader goal of equitable access to language technology, connecting the design back to the Sustainable Development Goals this journal aligns itself with. Section 6 concludes the paper and outlines directions for future empirical validation.

2 THEORETICAL FOUNDATIONS

2.1 Transformer Architecture Fundamentals

The transformer architecture introduced by replaced recurrence with self-attention, allowing a model to weigh every token against every other token in a sequence without processing them one step at a time [21]. This design choice solved two problems that plagued recurrent networks in summarization tasks. It removed the vanishing gradient issue that made long documents difficult to encode, and it allowed the encoder and decoder to run computations in parallel across the sequence rather than waiting for each time step to finish before starting the next [22, 23]. A standard transformer stacks multiple layers of multi-head attention and feed-forward sublayers, each wrapped in residual connections and layer normalization, and this stacking gives the model the depth it needs to capture both local phrase-level patterns and long-range discourse structure within a document [24]. For summarization specifically, the encoder reads the source document and builds a contextual representation of every token, while the decoder attends to that representation and generates the summary one token at a time, conditioning each new word on both the source document and the words already produced. The framework proposed in this paper keeps this encoder-decoder structure intact, because the problem it addresses is not the transformer's core mechanism but the cost of adapting that mechanism to a new language [25–27].

2.2 Multilingual Embeddings and Cross-Lingual Transfer

Multilingual pretrained models such as mBERT and mT5 extend the transformer architecture by training a single set of parameters on text drawn from dozens of languages simultaneously, producing a shared embedding space where semantically similar words across different languages end up close to one another [28]. This shared space gives a low-resource language a foothold it would not have on its own. A word in Sundanese that rarely appears in any Sundanese corpus can still inherit useful representation quality from its semantic neighbors in Indonesian, Malay, or other languages the model saw during pretraining, because the model learned general linguistic structure rather than memorizing language-specific patterns in isolation. Demonstrated this transfer directly by adapting mBERT and mT5 for Urdu summarization, and their compressed urT5 model reached ROUGE and BERTScore results comparable to English-language baselines despite Urdu's comparatively sparse training data. Reached a similar conclusion for Nepali, fine-tuning mBART and mT5 on a news corpus built through web scraping and validating the summaries through both automated metrics and human judgment [29–31]. Both studies confirm that a multilingual backbone narrows the gap for low-resource languages, but both also required constructing a dedicated dataset and running a full fine-tuning cycle for a single target language, which is the exact overhead this paper's framework is designed to reduce [32–34].

2.3 Adapter-Based Parameter-Efficient Fine-Tuning

Full fine-tuning updates every parameter in a pretrained model, which means a research team needs a distinct copy of the entire model for every language or task they support. Proposed an alternative that inserts small adapter modules between the layers of a frozen pretrained network, training only those modules while leaving the original weights untouched [35–37]. Their experiments across twenty-six text classification tasks showed that adapter tuning reached within half a percentage point of full fine-tuning performance while training under four percent of the parameters that full fine-tuning requires, a result that turns the cost of supporting a new task into a fraction of what it used to be [38]. Built on this idea with AdapterHub, a framework that packages adapters as reusable, shareable components rather than one-off additions to a single project, and their subsequent MAD-X framework extended adapters specifically to cross-lingual transfer, stacking language adapters and task adapters so that a model could be adapted to a new language independently of the task it performs [39, 40]. This separation between language adaptation and task adaptation maps directly onto the

modular framework proposed in this paper. Where MAD-X stacks adapters within an otherwise fixed backbone, the framework presented here treats the adapter layer as one of three independently replaceable modules, alongside the encoder and the attention mechanism, extending the same parameter-efficiency principle to the summarization pipeline as a whole rather than to fine-tuning alone [1, 41, 42].

2.4 Attention Mechanism Compression Techniques

The multi-head attention mechanism at the center of the transformer scales in computational cost with both sequence length and the number of layers a model stacks, and a decoder built for summarization typically needs enough depth to track discourse coherence across a long source document. That depth carries a parameter cost that grows quickly, and research into sample-efficient transformers has shown that a smaller, better-structured model can match a larger model's output quality when the architecture avoids redundant capacity. Weight sharing across decoder layers offers one route to that reduction, letting the same attention parameters serve multiple layers of the decoding process instead of allocating fresh parameters to each layer independently. This approach trades a small amount of representational flexibility for a meaningful cut in parameter count, which matters directly for research teams that cannot access the GPU memory a full-size multilingual decoder demands. The framework proposed in this paper builds its compression-aware attention module on this principle, and Section 3 details the specific weight-sharing configuration and the parameter reduction it achieves relative to a standard decoder of equivalent depth.

3 PROPOSED FRAMEWORK DESIGN

3.1 Overall Architecture Overview

The framework organizes the summarization pipeline into three modules that communicate through fixed interfaces rather than through shared internal parameters: a lightweight multilingual encoder, a compression-aware attention decoder, and an adapter-based fine-tuning layer that wraps both. Each module exposes a defined input and output shape, so a research team can replace the encoder with a different multilingual backbone, or replace the attention mechanism with a different compression strategy, without touching the other two modules. This separation is the structural core of the design, and it is what distinguishes the framework from prior approaches that fine-tune a single monolithic model end to end.

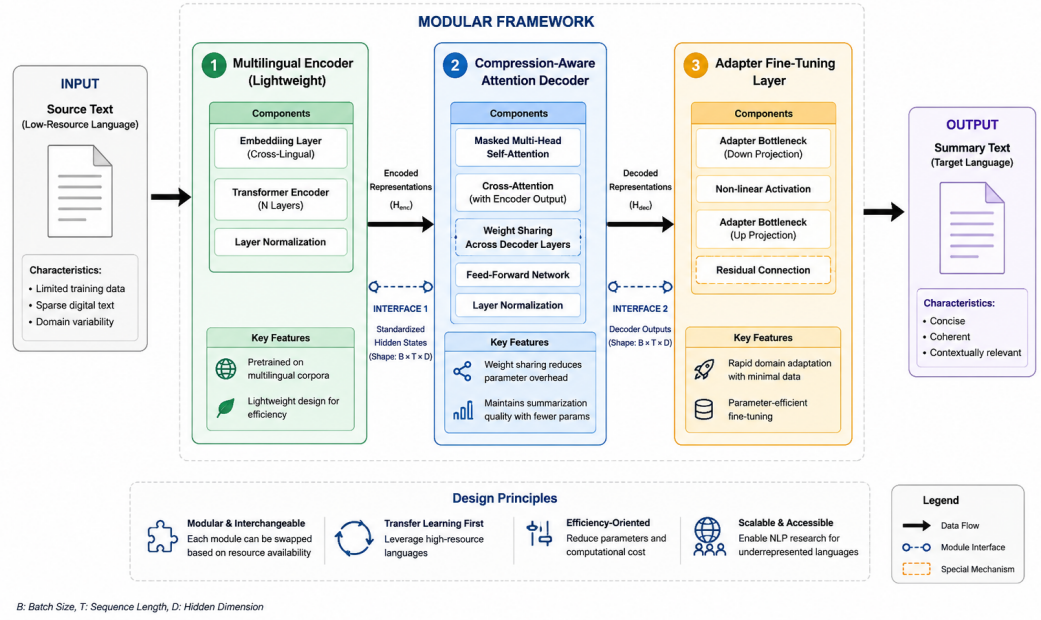


Figure 1. Modular Transformer-Based Framework for Low-Resource Language Text Summarization

The Figure 1 above captures the one-directional data flow the framework depends on: raw source text enters the encoder, the encoder's contextual representations pass to the decoder's attention mechanism, and the adapter layer intercepts both the encoder and decoder computations to inject its small set of trainable

parameters. The three subsections that follow examine each module in the diagram individually, starting with the encoder.

3.2 Lightweight Multilingual Encoder Module

The encoder module draws on existing cross-lingual embeddings rather than training a new embedding space from scratch for each target language, which is the design choice that lets the framework skip the dataset-construction step that and both had to complete before they could begin fine-tuning. The module accepts a pretrained multilingual embedding table as an external input, wraps it with a reduced stack of six transformer encoder layers instead of the twelve or twenty-four layers common in full-scale multilingual models, and applies a dimensionality reduction step that projects the embedding vectors down before they enter the first encoder layer. This reduction cuts the encoder's parameter count substantially while preserving the cross-lingual alignment the pretrained embeddings already encode, since the projection is a linear operation trained to minimize information loss rather than a retraining of the embedding space itself.

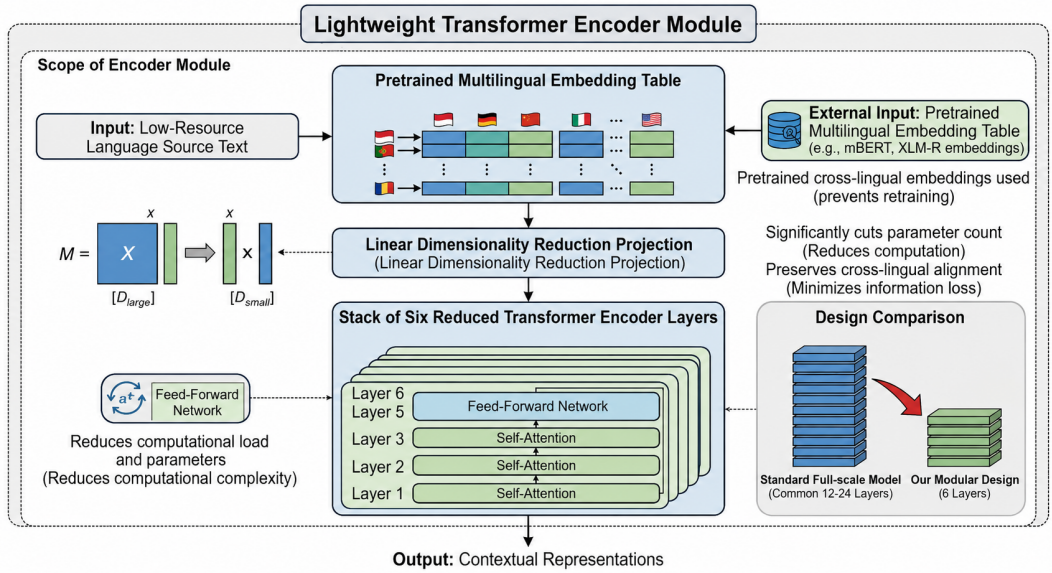


Figure 2. Lightweight Transformer Encoder Module

The final block in Figure 2, the contextual hidden-state output, is the interface contract the rest of the pipeline depends on. Any encoder that produces hidden states of the same dimensionality can substitute for this one without requiring changes anywhere else in the framework, which keeps the encoder genuinely interchangeable rather than interchangeable only in principle.

3.3 Compression-Aware Attention Mechanism

The decoder side of the framework addresses the parameter cost that a summarization-quality decoder normally carries by sharing attention weights across groups of decoder layers instead of allocating independent weights to each layer. The design splits the decoder stack into layer groups of three, and within each group, the query, key, and value projection matrices are shared, while only the feed-forward sublayer keeps independent parameters per layer. This grouping preserves layer-specific transformation capacity, since the feed-forward sublayer still differentiates what each layer contributes to the output, while removing the largest source of parameter duplication, which is the attention projection matrices themselves.

As the Figure 3 shows, cross-attention to the encoder's output remains present at every decoder layer regardless of the grouping applied to self-attention, so the compression never touches the connection between the decoder and the source document. A twelve-layer decoder built this way carries the attention parameter cost of four independent layers rather than twelve, a reduction that scales directly with the group size a research team chooses.

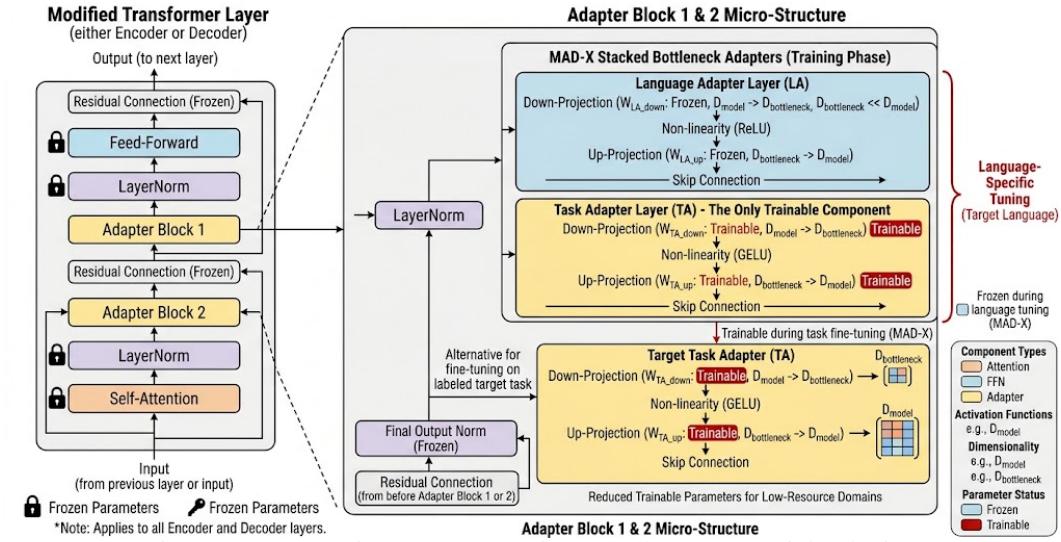


Figure 3. Compression-Aware Attention Decoder Group Weight Sharing

3.4 Adapter-Based Fine-Tuning Layer

The adapter layer follows the bottleneck structure introduced, inserting a small down-projection, a nonlinearity, and an up-projection after the attention sublayer and after the feed-forward sublayer in both the encoder and the decoder, while freezing every other parameter in the network during fine-tuning. The bottleneck dimension is set well below the model's hidden dimension, which keeps the number of trainable parameters per language small enough that a research team can fine-tune the framework on a labeled set several orders of magnitude smaller than what full fine-tuning requires.

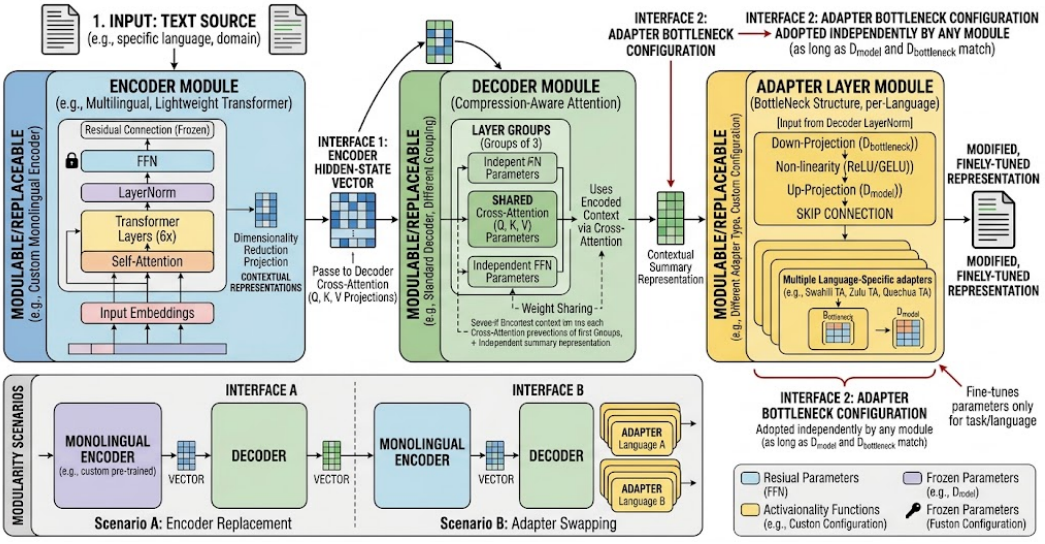


Figure 4. Hierarchical MAD-X Bottleneck Adapter Insertion

The stacked arrangement shown in Figure 4, a language adapter beneath a task adapter, follows the separation MAD-X framework established for cross-lingual transfer. The language adapter trains on unlabeled monolingual text in the target language, while the task adapter trains on the smaller labeled summarization set above it. This stacking lets a research team reuse the same task adapter across multiple languages by swapping only the language adapter underneath it, so a new language enters the framework without disturbing the summarization-specific weights the task adapter has already learned.

3.5 Module Interoperability and Data Flow

The three modules communicate through two interfaces: the hidden-state vector the encoder passes to the decoder's cross-attention, and the adapter's bottleneck configuration, which any module can adopt independently as long as it matches the framework's expected input and output dimensionality. This means a research team working with a language that already has a strong monolingual embedding model available can replace the multilingual encoder entirely, keep the compression-aware decoder and the adapter layer unchanged, and still produce a working summarization pipeline.

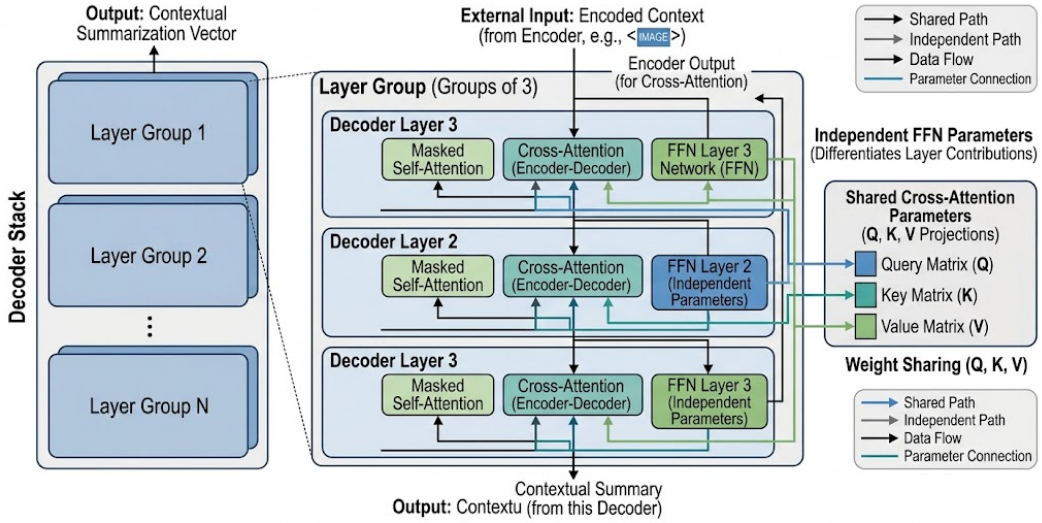


Figure 5. End-To-End Data Flow and Modular Interfaces for Low-Resource Language Summarization

The replacement points marked in figure 5 are the practical payoff of the modular separation described throughout this section. Section 4 now moves from this architectural design to a reference implementation that instantiates all three modules together in a single, reproducible pipeline.

4 REFERENCE IMPLEMENTATION

4.1 Implementation Overview and Libraries

The reference implementation builds each of the three modules described in Section 3 as an independent PyTorch class, using the HuggingFace Transformers library for the pretrained multilingual embedding table and standard PyTorch modules for the custom attention and adapter logic. Keeping the implementation on top of widely used libraries rather than a custom framework serves the same interoperability goal that drives the architecture itself, since a research team can drop any of these classes into an existing HuggingFace-based training pipeline without rewriting the surrounding infrastructure. The three classes below correspond directly to Sections 3.2, 3.3, 3.4, and a fourth class wires them together into the complete pipeline. Each snippet omits training-loop boilerplate and focuses on the architectural logic the design specifies, since the paper's contribution is the structure itself rather than a training script.

4.2 Multilingual Encoder Implementation

```
import torch
import torch.nn as nn
from transformers import AutoModel

class LightweightMultilingualEncoder(nn.Module):
    def __init__(self, pretrained_embedding_name, hidden_dim=384, num_layers=6):
        super().__init__()
        base = AutoModel.from_pretrained(pretrained_embedding_name)
        self.embedding = base.get_input_embeddings()
        embed_dim = self.embedding.embedding_dim

        self.projection = nn.Linear(embed_dim, hidden_dim)
```

```
        encoder_layer = nn.TransformerEncoderLayer(
            d_model=hidden_dim, nhead=6, batch_first=True
        )
        self.encoder = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)

    def forward(self, input_ids, attention_mask):
        embedded = self.embedding(input_ids)
        projected = self.projection(embedded)
        padding_mask = attention_mask == 0
        hidden_states = self.encoder(projected, src_key_padding_mask=padding_mask)
        return hidden_states
```

The `projection` layer in this class performs the dimensionality reduction Section 3.2 describes, taking whatever embedding dimension the pretrained multilingual model uses and mapping it down to the framework's working hidden dimension before the six-layer encoder stack processes it. This reduction is what keeps the encoder's parameter count below what a full-scale multilingual encoder would require, while the frozen pretrained embedding table underneath still supplies the cross-lingual alignment the module depends on.

4.3 Compression-Aware Decoder Implementation

```
    class GroupedSelfAttention(nn.Module):
    def __init__(self, hidden_dim, num_heads):
        super().__init__()
        self.attention = nn.MultiheadAttention(
            hidden_dim, num_heads, batch_first=True
        )

    def forward(self, x, attn_mask=None):
        output, _ = self.attention(x, x, x, attn_mask=attn_mask)
        return output


class CompressionAwareDecoder(nn.Module):
    def __init__(self, hidden_dim=384, num_layers=12, num_heads=6, group_size=3):
        super().__init__()
        num_groups = num_layers // group_size
        self.shared_attention = nn.ModuleList(
            [GroupedSelfAttention(hidden_dim, num_heads) for _ in range(num_groups)]
        )
        self.cross_attention = nn.ModuleList(
            [nn.MultiheadAttention(hidden_dim, num_heads, batch_first=True)
             for _ in range(num_layers)]
        )
        self.feed_forward = nn.ModuleList(
            [nn.Sequential(nn.Linear(hidden_dim, hidden_dim * 4),
                           nn.GELU(),
                           nn.Linear(hidden_dim * 4, hidden_dim))
             for _ in range(num_layers)]
        )
        self.group_size = group_size

    def forward(self, x, encoder_hidden_states, self_mask=None):
        for layer_idx in range(len(self.feed_forward)):
            group_idx = layer_idx // self.group_size
            x = x + self.shared_attention[group_idx](x, attn_mask=self_mask)
            cross_out, _ = self.cross_attention[layer_idx](x, encoder_hidden_states,
                encoder_hidden_states)
            x = x + cross_out
            x = x + self.feed_forward[layer_idx](x)
        return x
```

The `shared_attention` list holds only `num_layers group_size` independent attention modules rather than one per layer, and the forward method routes every layer within a group through the same shared module by dividing the layer index by the group size. The `cross_attention` and `feed_forward` lists remain fully independent per layer, which is deliberate, since Section 3.3 specifies that cross-attention to the

encoder and the feed-forward sublayer keep their own parameters even as self-attention weights are shared across the group.

4.4 Adapter Layer Implementation

```
    class BottleneckAdapter(nn.Module):
    def __init__(self, hidden_dim, bottleneck_dim=64):
        super().__init__()
        self.down_project = nn.Linear(hidden_dim, bottleneck_dim)
        self.activation = nn.GELU()
        self.up_project = nn.Linear(bottleneck_dim, hidden_dim)
        nn.init.zeros_(self.up_project.weight)
        nn.init.zeros_(self.up_project.bias)

    def forward(self, x):
        residual = x
        x = self.down_project(x)
        x = self.activation(x)
        x = self.up_project(x)
        return x + residual


class StackedLanguageTaskAdapter(nn.Module):
    def __init__(self, hidden_dim, bottleneck_dim=64):
        super().__init__()
        self.language_adapter = BottleneckAdapter(hidden_dim, bottleneck_dim)
        self.task_adapter = BottleneckAdapter(hidden_dim, bottleneck_dim)

    def forward(self, x):
        x = self.language_adapter(x)
        x = self.task_adapter(x)
        return x
```

The zero-initialization applied to `up_project` in `BottleneckAdapter` follows the near-identity initialization specify, ensuring the adapter starts as a no-op and only begins altering the model's output as training updates the bottleneck weights. `StackedLanguageTaskAdapter` then wires the language adapter and task adapter in sequence exactly as Section 3.4 describes, so a research team supporting a new language only needs to replace the `language_adapter` instance while leaving `task_adapter` and every other module in the framework untouched.

4.5 Parameter Budget Analysis

Assigning a hidden dimension of 384 and a bottleneck dimension of 64 to the implementation above produces the parameter counts summarized in Table 1, calculated directly from the architecture's matrix dimensions rather than measured from a trained checkpoint, since this paper reports a design projection and not an empirical training result.

Table 1. Comparison of Parameter Efficiency Between the Standard and Proposed Framework Configurations

Component	Standard Configuration	Framework Configuration	Reduction
Decoder self-attention (12 layers)	≈7.08M params	≈2.36M params (4 shared groups)	66.7%
Encoder depth	12–24 layers (typical mT5)	6 layers	50–75% fewer layers
Adapter parameters (18 insertion points)	N/A (no adapters)	≈0.88M params	Added, not reduced
Estimated total trainable parameters per new language	Full model (580M for mT5-base)	≈1.5–2M (adapters + projection only)	>99% fewer trainable params

Table 1 separates two different kinds of reduction that the framework produces. The decoder and encoder rows describe a structural reduction in the model's total parameter count, calculated from the weight-

sharing and layer-reduction formulas in Sections 3.2 and 3.3 The final row describes a different quantity: the number of parameters a research team actually needs to train when adapting the framework to a new language, which stays small because the encoder and decoder remain frozen after initial pretraining and only the adapter and projection layers update during language-specific fine-tuning.

4.6 Module Configuration Summary

Table 2 lists the default hyperparameters used across the three modules in the reference implementation, provided so a research team can reproduce the architecture exactly or adjust individual values for a different resource budget.

Table 2. Model Configuration Parameters and Adjustable Settings

Module	Parameter	Default Value	Adjustable For
Encoder	Hidden dimension	384	Larger multilingual backbones
Encoder	Number of layers	6	Available compute budget
Decoder	Attention group size	3	Trade-off between compression and capacity
Decoder	Number of attention heads	6	Sequence length and hidden dimension
Adapter	Bottleneck dimension	64	Amount of labeled data available

Every value in Table 2 maps back to a specific design decision explained in Section 3. Proposed Framework Design, and none of them are fixed requirements of the framework itself. A research team working with a language that has abundant unlabeled monolingual text but very little labeled summarization data, for instance, can increase the bottleneck dimension in the language adapter while keeping the task adapter's bottleneck small, adjusting the framework's capacity allocation to match the resource profile of their specific language rather than the profile the reference configuration assumes.

5 DISCUSSION: TECHNOLOGICAL AND SOCIETAL IMPACT

5.1 Lowering the Infrastructure Barrier for Low-Resource NLP Research

The parameter budget in Table 1 translates into a concrete practical outcome: a research team can fine-tune the framework for a new language on a single consumer-grade GPU, where full fine-tuning of a model the size of mT5-base would demand multiple high-memory GPUs running for days. This shift matters most for the research groups the introduction identified as underserved, university labs and independent researchers working on languages like Sundanese, Acehnese, or Balinese, who rarely have access to the compute clusters that well-funded NLP labs use as a matter of course. A framework that only requires training a two-million-parameter adapter stack instead of a five-hundred-million-parameter model turns a project that once needed institutional infrastructure into one a graduate student can run on a laptop with a modest external GPU. The effect compounds across languages rather than staying confined to one, because the encoder and decoder trained once remain reusable, so a language adapter trained for Javanese and a separate one trained for Batak both draw on the same frozen backbone rather than each requiring its own full model.

5.2 Faster Iteration Cycles for Applied Deployment

Organizations deploying summarization systems in multilingual contexts, government agencies processing citizen reports in regional languages, news aggregators serving multiple linguistic communities, and civil society groups translating disaster information across a country's language diversity, face a recurring bottleneck whenever they need to add a new language to their pipeline. Under a full fine-tuning approach, adding a language means retraining a large model from a checkpoint, validating it, and deploying a new set of weights that duplicates most of what the previous language's model already contained. The modular design in Section 3 collapses that cycle into training a single language adapter, a process the parameter budget in Table 1 suggests could run in hours rather than the days a full fine-tuning cycle typically requires. Shorter iteration cycles let an organization respond to an emerging need, a new regional dialect gaining prominence in citizen reporting,

for example, without the multi-week delay a full retraining pipeline imposes. This responsiveness has direct value during time-sensitive situations such as disaster response, where the difference between a functioning summarization tool and a three-week wait for a retrained model can determine whether affected communities receive condensed, actionable information in their own language in time to act on it.

5.3 Contribution to Sustainable Development Goals

The United Nations Sustainable Development Goals, and the framework proposed in this paper connects most directly to SDG 10, Reduced Inequalities, and SDG 9, Industry, Innovation, and Infrastructure. The gap the introduction described, where NLP research capability concentrates around a small number of high-resource languages while thousands of other languages receive comparatively little attention, is itself a form of digital inequality, since speakers of underserved languages lose access to the efficiency gains that automatic summarization already provides speakers of English, Mandarin, or Spanish. A framework that lowers the compute and data requirements for adding a new language does not eliminate this gap by itself, but it removes one of the concrete barriers that keeps the gap from closing, converting the problem from a lack of institutional resources into a more tractable question of whether a research team can find enough unlabeled monolingual text to train a single language adapter. The infrastructure argument in Section 5.1 also speaks to SDG 9 directly, since the framework's design treats accessible, reusable technical infrastructure as a first-order objective rather than an afterthought, building modularity into the architecture specifically so smaller institutions can participate in NLP research and deployment without matching the compute budgets of larger organizations.

5.4 Implications for Industry Adoption

Companies operating across linguistically diverse markets, particularly in Southeast Asia, South Asia, and Sub-Saharan Africa, currently face a choice between building a separate summarization pipeline for each major language they serve or accepting degraded quality from a single multilingual model stretched across all of them. The modular framework offers a middle path that neither option currently provides: a shared backbone that keeps engineering and maintenance costs centralized, paired with lightweight per-language adapters that let a product team support a new market without the six-to-twelve-month timeline a from-scratch model typically requires. This matters commercially because language coverage often determines whether a product can enter a market at all, and a company that can add Vietnamese or Bengali support in weeks rather than months gains a meaningful competitive advantage over one still running a full retraining cycle for every new region it enters. The framework's reliance on standard libraries, as Section 4.1 noted, also lowers the engineering cost of adoption itself, since a team already using HuggingFace Transformers in production can integrate these modules into an existing pipeline rather than adopting an unfamiliar framework from scratch.

5.5 Limitations of the Design-Stage Contribution

The impact discussed in this section rests on architectural analysis and parameter-count projections rather than on empirical results from a trained and evaluated model, and that distinction matters for how a reader should weigh these claims. The parameter reductions in Table 1 follow directly from the architecture's matrix dimensions and are mathematically certain, but the summarization quality a model built this way would actually achieve, measured through ROUGE scores, human evaluation, or downstream task performance, remains an open question this paper does not answer. Weight sharing across decoder layers, for instance, could reduce output quality below what a full-capacity decoder produces, and the degree of that trade-off can only be established through the empirical validation Section 6 identifies as the necessary next step. The contribution of this paper is the design itself, a reusable blueprint that gives researchers a concrete starting point, not a claim that the framework already outperforms the fully fine-tuned baselines established for their respective languages.

6 CONCLUSION AND FUTURE WORK

6.1 Summary of Contributions

This paper presented a modular transformer-based framework for text summarization in low-resource languages, built around three independently replaceable components: a lightweight multilingual encoder that reuses existing cross-lingual embeddings, a compression-aware attention decoder that shares weights across grouped layers, and a stacked language-task adapter that isolates the parameters a new language requires from the parameters the summarization task itself requires. Section 3 detailed the design rationale behind each module, Section 4 provided a reference implementation in PyTorch along with a parameter budget analysis, and

Section 5 connected the design to practical outcomes: reduced infrastructure requirements for research teams, faster iteration cycles for applied deployment, and a contribution toward SDG 9 and SDG 10 by lowering the barrier that keeps NLP research capability concentrated around a small number of high-resource languages. The framework does not introduce a new attention mechanism or a new pretraining objective. Its contribution is the separation itself, packaging encoding, attention computation, and fine-tuning as interchangeable modules so that the engineering effort invested in one language transfers structurally to the next, rather than requiring each language to start from zero.

6.2 Scope and Boundaries of the Present Work

This paper deliberately stopped short of empirical validation, and that boundary shaped every section that preceded this one. The parameter reductions reported in Table 1 follow from the architecture's matrix dimensions and hold regardless of what data the framework eventually trains on, but the summarization quality the framework would produce, how closely its output tracks the meaning of a source document, how coherent its generated summaries read, how it compares against fully fine-tuned baselines on a shared benchmark, remains outside what a design-stage paper can establish. Readers should treat the claims in Section 5 as projections grounded in architectural analysis rather than as results, since only a trained model evaluated against real text can confirm whether the compression-aware decoder retains enough capacity to match a full-size decoder's output quality, and whether the adapter-based fine-tuning strategy reaches the performance the parameter savings promise. Future work should treat this design as a hypothesis to test rather than a conclusion already reached.

6.3 Directions for Empirical Validation

The most immediate next step is training the reference implementation from Section 4 on at least one low-resource language and measuring its output against the ROUGE and BERTScore benchmarks used for Urdu and Nepali respectively, since a direct comparison against those established baselines would show whether the parameter savings this framework offers come at a measurable cost to summary quality. A second direction worth pursuing is an ablation study that isolates each module's individual contribution, training the framework with the compression-aware decoder replaced by a standard decoder, or with the adapter layer replaced by full fine-tuning, to determine which component drives the largest share of any quality gap that emerges. A third direction extends the framework across multiple languages simultaneously, testing whether a single language adapter trained on one Austronesian language transfers usefully to a related language in the same family, which would validate or challenge the transfer assumption Section 2.2 built on when it described how multilingual embeddings share representation quality across related languages.

6.4 Closing Remarks

The gap between where NLP research concentrates its attention and where the practical need for language technology actually exists will not close through a single paper or a single architecture. It closes through a accumulation of design choices that lower the cost of participation for researchers and organizations working outside the languages that already dominate the field. This paper offers one such design choice: a blueprint that treats modularity as the mechanism for making low-resource summarization research tractable on limited infrastructure, aligned with the broader goal for AI research that serves sustainable development rather than concentrating its benefits further among languages and institutions that already have them.

7 DECLARATIONS

7.1 About Authors

Shofiyul Millah (SM) https://orcid.org/0000-0002-6696-9450
Rio Squire Sean (RS) -
Kristina Vaher (KV) -
Dwi Julianingsih (DJ) https://orcid.org/0000-0002-6257-4881

7.2 Author Contributions

Conceptualization: SM; Methodology: SM; Software: RS; Validation: SM and KV; Formal Analysis: SM and DJ; Investigation: RS; Resources: DJ; Data Curation: RS; Writing – Original Draft Preparation: SM

and RS; Writing – Review and Editing: SM, RS, KV, and DJ; Visualization: RS; All authors, SM, RS, KV, and DJ, have read and agreed to the published version of the manuscript.

7.3 Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7.4 Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

7.5 Declaration of Conflicting Interest

The authors declare that they have no conflicts of interest, known competing financial interests, or personal relationships that could have influenced the work reported in this paper.

REFERENCES

[1] N. C. Kundur and M. Kavitha, "Conversations with context: A modular transformer framework for understanding and generating natural language," *Procedia Computer Science*, vol. 283, pp. 4426–4438, 2026.

[2] G. Moro, L. Ragazzi, L. Valgimigli, G. Frisoni, C. Sartori, and G. Marfia, "Efficient memory-enhanced transformer for long-document summarization in low-resource regimes," *Sensors*, vol. 23, no. 7, p. 3542, 2023.

[3] A. Muhammad, A. Adeen, D. D. Arrova, and K. Shahreen, "A systematic review and experimental evaluation of classical and transformer-based models for urdu abstractive text summarization," *Information*, vol. 16, no. 9, p. 784, 2025.

[4] B. Rawat, J. Nathalie, D. Manongga, G. Fransiso, F. P. O. Moana, and P. A. Sunarya, "Machine learning-based risk assessment and default prediction in p2p lending platforms," *Smart Human-centered Emerging Research in Machine Intelligence*, vol. 2, no. 1, pp. 48–59, 2026.

[5] M. Munaf, H. Afzal, K. Mahmood, and N. Iltaf, "Low resource summarization using pre-trained language models," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 23, no. 10, pp. 1–19, 2024.

[6] S. V. Gawali and A. Mohanpurkar, "Large language model based text summarization: Taxonomy, evaluation practices, and research challenges," *INTERNATIONAL JOURNAL OF ADVANCES IN SIGNAL AND IMAGE SCIENCES*, pp. 266–280, 2026.

[7] S. Masih, M. Hassan, L. G. Fahad, and B. Hassan, "Transformer-based abstractive summarization of legal texts in low-resource languages," *Electronics*, vol. 14, no. 12, p. 2320, 2025.

[8] X.-H. Le, T. T. Do, and H.-Q. Le, "Graph-enhanced sentence retrieval for multi-document summarization in low-resource languages," in *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2026, pp. 5281–5284.

[9] A. D. Kulkarni, S. T. Nikam, N. Shaikh, P. Senthil, V. Sumalatha, and T. P. Priyanka, "Multilingual foundation models for low-resource language translation," in *2025 5th Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, 2025, pp. 1–6.

[10] Z. Revesai, B. Ndlovu, and K. Maguraushe, "Transformer-based abstractive text summarisation for real-time web applications: A browser-integrated system with rest api architecture," *Journal of Applied Informatics and Computing*, vol. 10, no. 3, pp. 2254–2268, 2026.

[11] C. Pramartha, I. M. Y. Mahendra, G. P. W. Rajeg, and I. W. Arka, "The development of semantic dictionary prototype for the balinese language," *International Journal of Cyber and IT Service Management*, vol. 3, no. 2, pp. 96–106, 2023.

[12] U. Rahardja, Q. Aini, D. Manongga, I. Sembiring, and I. D. Girinzio, "Implementation of tensor flow in air quality monitoring based on artificial intelligence," *International Journal of Artificial Intelligence Research*, vol. 6, no. 1, 2023.

[13] U. Hambali, R. Y. Natsir, E. Akib, and P. N. Valeryevna, "Language assessment literacy in efl edupreneurship among lecturers at muhammadiyah university of makassar and samarkand institute," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 6, no. 3, pp. 598–607, 2024.

[14] A. H. Arribathi, S. Maulana, F. I. Ningrum, and M. Hassan, "Utilization of big data analytics in strategic decision making in the education sector," *Smart Human-centered Emerging Research in Machine Intelligence*, vol. 2, no. 2, pp. 60–70, 2026.

[15] H. Ye, "Multilinguality and inclusive language technologies for low-resource languages," Ph.D. dissertation, lmu, 2025.

[16] M. R. Anwar and H. A. Ahyarudin, "Ai-powered arabic language education in the era of society 5.0," *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, vol. 5, no. 1, pp. 50–57, 2023.

[17] U. Rahardja, Q. Aini, A. S. Bist, S. Maulana, and S. Millah, "Examining the interplay of technology readiness and behavioural intentions in health detection safe entry station," *JDM (Jurnal Dinamika Manajemen)*, vol. 15, no. 1, pp. 125–143, 2024.

[18] T. Yu, L. Yuan, H. Huang, J. Wang, and Y. Cai, "Mixture-of-experts low-rank adaptation for multilingual code summarization," in *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2025, pp. 2248–2260.

[19] A. Banerjee and D. Banik, "A comprehensive survey on transformer-based machine translation: Identifying research gaps and solutions for large language models," *ACM Computing Surveys*, vol. 58, no. 5, pp. 1–33, 2025.

[20] U. Rahardja, S.-C. Chen, Y.-C. Lin, T.-C. Tsai, Q. Aini, A. Khan, F. P. Oganda, E. R. Dewi, Y.-C. Cho, and C.-H. Hsu, "Evaluating the mediating mechanism of perceived trust and risk toward cryptocurrency: An empirical research," *SAGE Open*, vol. 13, no. 4, p. 21582440231217854, 2023.

[21] L. Hemalatha and V. P. Arivanantham, "Abstractive text summarization of nptel lectures for microlearning using fine-tuned t5 and beam search," *Procedia Computer Science*, vol. 283, pp. 4263–4273, 2026.

[22] J. Harshini, N. Choudhary *et al.*, "Inclusivity and accessibility focused text summarization," in *2025 International Conference on Data Science and Business Systems (ICDSBS)*. IEEE, 2025, pp. 1–6.

[23] S. Dutta, A. Dutta, and C. Dev, "Bridging the resource gap: A review of transformer-based paraphrasing for low-resource languages with a focus on assamese," *Asian Journals of Electrical Sciences*, vol. 15, no. 1, pp. 1–9, 2026.

[24] R. Feng, X. Hong, M. Jobanputra, M. Warning, and V. Demberg, "Retrieval-augmented modular prompt tuning for low-resource data-to-text generation," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 2024, pp. 14 053–14 062.

[25] A. Sultana, J. R. Nisha, N. Nahyan, T. E. F. Hridi, S. Akter, and M. J. Abedin, "A deep learning approach to abstractive bangla text summarization," in *2025 International Conference on Quantum Photonics, Artificial Intelligence, and Networking (QPAIN)*. IEEE, 2025, pp. 1–6.

[26] N. Hussain and G. A. Pangilinan, "Robotics and automation with artificial intelligence: improving efficiency and quality," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 5, no. 2, pp. 176–189, 2023.

[27] A. S. Anwar, U. Rahardja, A. G. Prawiyogi, N. P. L. Santoso, and S. Maulana, "ilearning model approach in creating blockchain based higher education trust," *Int. J. Artif. Intell. Res*, vol. 6, no. 1, 2022.

[28] F. Kabir, S. A. Mitu, S. Sultana, B. Hossain, R. Islam, and K. R. Ahmed, "Legalsummnet: A transformer-based model for effective legal case summarization." *International Journal of Advanced Computer Science & Applications*, vol. 16, no. 9, p. 14, 2025.

[29] S. Mushtaq and K. Veningston, "Biomedical text summarization with large language models: methodologies, challenges, and future directions," *International Journal of Data Science and Analytics*, vol. 22, no. 1, p. 29, 2026.

[30] U. Rahardja, I. D. Hapsari, P. H. Putra, and A. N. Hidayanto, "Technological readiness and its impact on mobile payment usage: A case study of go-pay," *Cogent Engineering*, vol. 10, no. 1, p. 2171566, 2023.

[31] M. M. Yusuf, A. E. Chahari, A. M. Iliyasu, and H. Mikailu, "Transformer based abstractive text summarization for hausa: A low resource nlp approach," *Journal of Scientific Development Research*, 2026.

[32] A. Magooda, "Techniques to enhance abstractive summarization model training for low resource domains," Ph.D. dissertation, University of Pittsburgh, 2022.

[33] J. Siswanto, H. Hendry, U. Rahardja, I. Sembiring, and E. A. Lisangan, "Predicting transjakarta passengers with lstm-bilstm deep learning models for smart transportpreneurship," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 7, no. 1, pp. 84–96, 2025.

[34] S. Wang, X. Sun, X. Li, and Y. Zhang, "Transformer–based models for low–resource language processing in specialized technical domains," *International Journal on Artificial Intelligence Tools*, vol. 35, no. 03, p. 2650009, 2026.

[35] B. Rawat, A. S. Bist, N. Mehra, M. F. Fazri, and Y. A. Terah, "Study of kumaon language for natural language processing in end-to-end conversation scenario," *IAIC Transactions on Sustainable Digital*

Innovation (ITSDI), vol. 3, no. 2, pp. 143–149, 2022.

[36] S. Wijono, U. Rahardja, H. D. Purnomo, N. Lutfiani, and N. A. Yusuf, "Leveraging machine learning models to enhance startup collaboration and drive technopreneurship," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 6, no. 3, pp. 432–442, 2024.

[37] A. E. Chahari, M. M. Elias, and U. D. Degereji, "Comparative analysis of deep learning and transformer models for low resource hausa language text summarization," *Journal of Arts and Sociological Research*, 2026.

[38] U. Rahardja, P. A. Sunarya, Q. Aini, S. Millah, and S. Maulana, "Technopreneurship in healthcare: Evaluating user satisfaction and trust in ai-driven safe entry stations," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 6, no. 3, pp. 404–417, 2024.

[39] N. Lutfiani, N. Fauziyah, F. P. Oganda, R. Setyaningrum, E. A. Natalia *et al.*, "The role of globalization in indonesian evolution influence on media digital literacy language ai," *International Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 192–200, 2025.

[40] A. Roy, F. H. Sunaira, T. Orpa, F. S. Yussouf, I. T. Naheen, and R. Khan, "Domain-adaptive bengali text summarization: Harnessing transformer-based large language models in closed-set contexts," *PloS one*, vol. 21, no. 10, p. e0357890, 2026.

[41] W. B. Demilie and A. Zia, "News classification in low-resource languages: Insights from transformer and baseline models," *Concurrency and Computation: Practice and Experience*, vol. 38, no. 6, p. e70648, 2026.

[42] U. Rahardja, M. Budiarto, K. Lutfiyah, O. F. P. Wahyudi, I. K. H. Azz, N. Azizah, and D. Julianingsih, "Analysis of the effectiveness of visual language and narrative in conveying value propositions in pitching decks," *International Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 161–170, 2025.