TALL: A Trainable Architecture for Enhancing LLM Performance in Low-Resource Languages

Moshe Ofer, Orel Zamler, Amos Azaria

Ariel University

Correspondence: mosheo@ariel.ac.il, amos.azaria@ariel.ac.il

Abstract

Large Language Models (LLMs) excel in high-resource languages but struggle with low-resource languages due to limited training data. This paper presents *TALL* (**T**rainable **A**rchitecture for Enhancing **L**LM Performance in **L**ow-Resource Languages), which integrates an LLM with two bilingual translation models. TALL transforms low-resource inputs into high-resource representations, leveraging the LLM's capabilities while preserving linguistic features through dimension alignment layers and custom transformers. Our experiments on Hebrew demonstrate significant improvements over several baselines, including direct use, naive translation, and fine-tuning approaches. The architecture employs a parameter-efficient strategy, freezing pre-trained components while training only lightweight adapter modules, balancing computational efficiency with performance gains.[1]

1 Introduction

Large Language Models (LLMs) have significantly advanced natural language processing (NLP) for high-resource languages like English, but their performance in low-resource languages remains limited due to sparse data, linguistic complexity, and the scarcity of annotated datasets and pre-trained models. Efforts to address these challenges have primarily focused on transfer learning (Pan and Yang, 2010), which leverages knowledge from high-resource languages to enhance performance in low-resource settings, though notable limitations persist due to linguistic diversity and persistent data scarcity.

This paper introduces *TALL*, a novel architecture that harnesses the strengths of LLMs trained on high-resource languages to improve performance on low-resource languages. The key innovation lies in integrating three pre-trained models: an LLM trained primarily on a high-resource language, and two translation models that bridge the gap between high and low-resource languages. Through carefully designed dimension alignment adapters (Rebuffi et al., 2017; Houlsby et al., 2019) and custom transformer (Vaswani et al., 2017) components, *TALL* ensures that the linguistic knowledge embedded in high-resource LLMs benefits low-resource language processing.

We validate *TALL* through experiments on Hebrew, chosen for its rich morphology, complex syntax, and limited available datasets. Our results demonstrate significant improvements in accuracy compared to direct use of LLMs, naive translation approaches, soft prompting, and fine-tuning, highlighting the effectiveness of our architecture.

2 Related Work

Our work builds upon several approaches to improving LLM performance in low-resource settings:

Cross-lingual Transfer Prior research has explored transferring knowledge from high-resource to low-resource languages through various methods. Cross-lingual in-context learning (Cahyawijaya et al., 2024) leverages examples from high-resource languages to improve performance in low-resource contexts. However, these approaches often lack the structural integration that *TALL* provides.

Parameter-Efficient Adaptation Adapter-based approaches (Pfeiffer et al., 2020a,b) insert small, trainable modules into frozen pre-trained models to adapt them to new tasks or languages. Our dimension alignment adapters follow this principle, but specifically focus on aligning representations between different language spaces.

[1]Our code is available at `https://github.com/MosheOfer1/TALL`.

Translation-based Strategies Recent work has introduced creative translation-based strategies. Notable examples include TaCo (Upadhayay and Behzadan, 2024), which integrates translation into reasoning; CoTR (Deshpande et al., 2024), which translates inputs into a high-resource language for processing; and TALENT (Guo et al., 2024), which employs a curriculum-inspired approach to grammar learning before translation.

Concept-level Processing Large Concept Models (LCM-team et al., 2024) operate on sentence-level "concepts" in a unified multilingual embedding space, offering an alternative to token-level prediction. Similar to *TALL*, they aim to create cross-lingual representations that preserve semantic meaning across languages.

3 Model Architecture

3.1 Architecture Overview

TALL strategically integrates pre-trained components with lightweight, trainable adapters to efficiently process low-resource language inputs. As illustrated in Figure 1, the architecture consists of seven main stages (numbered 1-7 in the figure):

1. **Source Language Encoding:** The input sentence in the low-resource language is processed by a pre-trained low-to-high-resource (LR-HR) encoder, generating hidden states that represent the intermediate semantic form.

2. **Dimension Alignment (Source to LLM):** A trainable dimension alignment adapter transforms the encoder's hidden states to match the dimensional requirements of the LLM, ensuring seamless integration.

3. **Custom Transformer (LR-HR):** This trainable component receives the LLM tokenized translated sentence embeddings (during training, the last word is removed before the sentence is translated) and applies cross attention (Gheini et al., 2021) to the aligned hidden states, ensuring the LLM receives representations enriched with information from the original input.

4. **LLM Integration:** The processed sequence is passed to the frozen LLM for next-token prediction, leveraging its strong language modeling capabilities in the high-resource language.

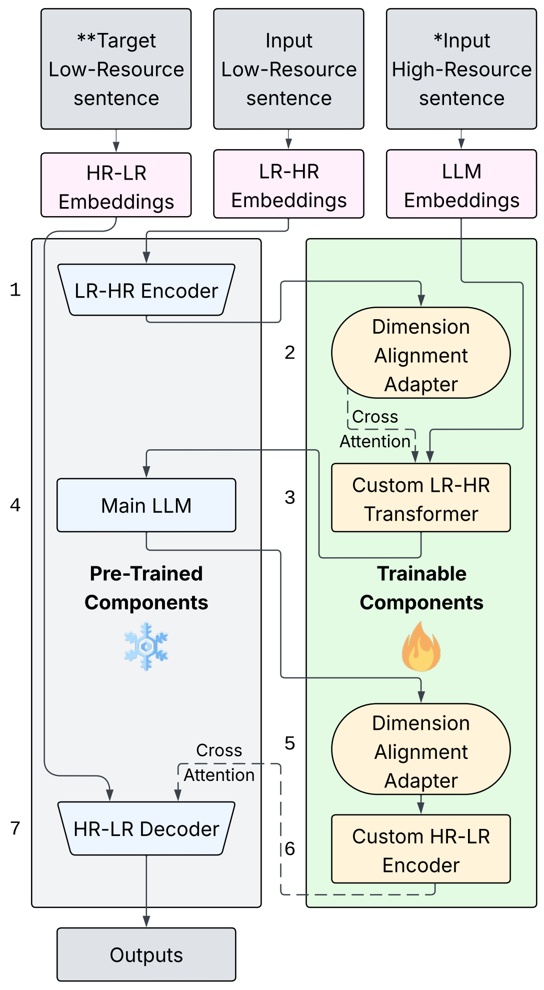

Figure 1: Overview of the TALL architecture with numbered components (1-7) corresponding to the processing stages described in Section 3.1. * For input high-resource sentences, the last word is removed before translation during training; ** For target sentences, teacher forcing is used during training while outputs are generated auto-regressively during inference.

5. **Dimension Alignment (LLM to Target):** A second alignment adapter transforms the LLM's hidden states to match the dimensional requirements of the second translator.

6. **Custom Transformer (HR-LR):** This component processes the aligned hidden states from the LLM, preparing them for the final decoding stage.

7. **Target Language Decoding:** The HR-LR decoder generates the final text in the target low-resource language by applying cross-attention between the encoded representation and the target language embeddings.

The implementation details for each component are as follows:

Translation Models We use pre-trained Marian MT models (Junczys-Dowmunt et al., 2018) for translation between Hebrew and English, leveraging their robust bilingual capabilities.

Dimension Alignment Adapters These adapters consist of trainable fully-connected layers with layer normalization and GELU activations. They project hidden states between the different dimensional spaces required by each pre-trained component.

Custom Transformers We implement two custom transformer modules derived from Marian configurations: (1) a transformer that conditions on high-resource tokens and cross-attends to dimension-aligned low-resource encoder states, and (2) a transformer that refines representations before the final decoding stage.

Parameter Efficiency A key advantage of *TALL* is its parameter efficiency. By freezing the large pre-trained components (translation models and LLM) and training only the lightweight alignment adapters and custom transformers, we minimize computational overhead while maximizing performance gains. In our implementation with bloomz-560m (Muennighoff et al., 2023), only 14.35% of the total parameters (126.8M out of 883.5M) are trainable, while for QWEN2.5-0.5b (Qwen-Team, 2024), 13.47% of the total parameters (107.7M out of 799.2M) are trainable.

A detailed breakdown of the TALL parameters, including overall statistics and module-wise counts for both bloomz TALL and Qwen TALL, is provided in Tables 2, 3, and 4 in the Appendix.

3.2 Training Procedure

We trained *TALL* on 256,000 Hebrew sentences from news articles. The dataset underwent standard preprocessing, including removal of duplicates and filtering for sentences between 5-30 words in length.

The model is trained using teacher forcing, where ground-truth target tokens are provided as decoder inputs. Importantly, the loss is computed only for the final missing token in the sequence. This limitation is inherent to the TALL architecture design: we translate the truncated sentence (without the final word) into the high-resource language and feed it to the Custom Transformer (LR-HR). Since the rest of the tokens are already presented as the high-resource language translation, we can only compute the prediction signal for the missing final token. While it would be preferable to obtain training signals from all tokens (as in standard language modeling), the current cross-lingual architecture constrains us to optimizing for the final token prediction only (see Section 6).

Training hyperparameters and optimization details are provided in the Appendix 7.

4 Experiments and Results

4.1 Experimental Setup

We evaluated *TALL* on the task of predicting a missing Hebrew word at the end of a given sentence, comparing it against several baseline approaches:

- **Direct Hebrew:** The LLM attempts to predict the missing Hebrew word directly.
- **Fine-tuned:** The LLM is fine-tuned on the news dataset.
- **From-Scratch:** A language model with identical architecture to the original LLM but trained entirely from scratch on the news dataset.
- **Naive:** The Hebrew sentence is translated to English, the LLM predicts the next English token, and the completed English sentence is translated back to Hebrew.
- **Soft Prompt:** A set of trainable soft prompt parameters is added to guide the LLM toward better predictions (Lester et al., 2021).
- ***TALL*:** Our proposed architecture that integrates translation models with the LLM through alignment adapters.

We used two LLMs with different levels of Hebrew exposure: bloomz-560m (Muennighoff et al., 2023) with minimal Hebrew exposure, and QWEN2.5-0.5b (Qwen-Team, 2024) with moderate Hebrew exposure. The evaluation was conducted on two datasets of 50,000 Hebrew sentences each: WIKI-SLVM from the Hebrew Wikipedia Corpus (NLPH, 2019) and HE-LYRICS sampled from the Hebrew Stage and Lyrics dataset (Norod78, 2023).

Dataset	Approach	Bloomz-560m	QWEN2.5-0.5b
WIKI-SLVM	Direct Hebrew	0.63	3.77
	Fine-tuned	0.16	2.75
	Naive	2.11	2.85
	Soft prompt	2.50	1.96
	From Scratch	2.93	3.99
	TALL	**5.59**	**5.15**
HE-LYRICS	Direct Hebrew	0.14	2.11
	Fine-tuned	0.36	2.22
	Naive	1.83	2.20
	Soft prompt	2.05	1.90
	From Scratch	2.08	2.92
	TALL	**5.44**	**4.77**

Table 1: Accuracy comparison (%) for predicting the missing Hebrew word. TALL, soft prompt, fine-tuning, and from-scratch were trained on the news dataset.

4.2 Results and Analysis

Table 1 presents the accuracy results for predicting the missing Hebrew word. *TALL* consistently outperforms all baseline approaches, achieving up to 5.59% accuracy with bloomz-560m and 5.15% with QWEN2.5-0.5b on the WIKI-SLVM dataset, compared to the next best methods at 2.93% and 3.99% respectively.

Direct Hebrew prediction performs poorly (0.63% with bloomz-560m), and surprisingly, fine-tuning often degrades performance further (0.16%). Translation-based approaches show modest improvements over direct Hebrew prediction, suggesting that leveraging high-resource representations provides some benefit. However, these approaches fail to fully bridge the gap between languages. QWEN2.5-0.5b demonstrates higher baseline accuracy than bloomz-560m, reflecting its greater Hebrew exposure during pretraining, yet the relative improvement from *TALL* is more pronounced with bloomz-560m.

Despite freezing approximately 86% of parameters, *TALL* achieves substantial improvements by effectively aligning representations between languages while preserving the strengths of pretrained models. This demonstrates that our architecture successfully leverages high-resource language capabilities to enhance performance in low-resource contexts without extensive retraining.

5 Conclusion and Future Work

This paper introduced *TALL*, a novel architecture for enhancing LLM performance in low-resource languages. By integrating high-resource language LLMs with bilingual translation models through trainable alignment adapters, *TALL* effectively leverages the strengths of pre-trained models while minimizing computational overhead.

Our experimental results on Hebrew demonstrate that *TALL* significantly outperforms direct use of LLMs, naive translation approaches, models trained from scratch, and fine-tuning techniques. The architecture maintains strong performance across different domains, highlighting its robustness and generalization capabilities.

Key advantages of *TALL* include:

- **Parameter Efficiency:** By freezing large pre-trained components and training only lightweight adapters, *TALL* achieves significant performance gains with minimal computational overhead.
- **Cross-domain Generalization:** *TALL* demonstrates robust performance consistency across different text domains, maintaining its effectiveness when applied to diverse content types.
- **Modularity and Extensibility:** The architecture can be adapted to various low-resource languages by substituting the appropriate translation models, offering a flexible solution for multilingual NLP.

Future work will extend *TALL* to additional low-resource languages and further refine the alignment mechanisms. By continuing to develop this approach, we aim to contribute to more inclusive and accessible NLP technologies that support all languages, regardless of their resource availability.

6 Limitations

While *TALL* presents a promising framework for enhancing LLM performance in low-resource languages, several limitations warrant consideration:

Inference Efficiency Challenges The integration of multiple translation and alignment modules complicates efficient inference, particularly for key-value caching typically used in LLM pipelines. This results in slower inference compared to standard LLMs. Future work could explore partial caching techniques or optimized implementations to address this limitation.

Reliance on Translation Models *TALL* depends on pre-trained translation models for the target language pair, which may introduce biases or errors inherent in these models. For extremely low-resource languages where quality translation models are unavailable, this dependency could limit applicability. Alternative approaches using multilingual translation models or zero-shot cross-lingual transfer could potentially address this challenge.

Single-Token Learning Signal Our current implementation focuses on predicting only the final token of a sequence, which limits the learning signal compared to conventional LLM training that predicts every token. This design choice effectively addresses our evaluation task but may constrain the model's ability to learn broader contextual dependencies. Future iterations could explore curriculum learning strategies or multi-stage training protocols with additional per-token predictions.

Error Propagation As a multi-stage pipeline, errors can accumulate and propagate across translation, alignment, and decoding stages.

Adapter Layer Information Bottleneck The dimension alignment adapters, while necessary for integration, could introduce information bottlenecks that limit the maximum achievable performance. Future refinements may explore more expressive adapter architectures or non-linear alignment techniques to mitigate this limitation.

These limitations highlight clear research directions for improving the efficiency, generalizability, and performance of cross-lingual transfer approaches like *TALL*.

References

Samuel Cahyawijaya, Holy Lovenia, and Pascale Fung. 2024. Llms are few-shot in-context low-resource language learners. *arXiv preprint arXiv:2403.16512*, pages 405–433.

Tejas Deshpande, Nidhi Kowtal, and Raviraj Joshi. 2024. Chain-of-translation prompting (cotr): A novel prompting technique for low resource languages. *Preprint*, arXiv:2409.04512.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. On the strengths of cross-attention in pretrained transformers for machine translation. *CoRR*, abs/2104.08771.

Ping Guo, Yubing Ren, Yue Hu, Yunpeng Li, Jiarui Zhang, Xingsheng Zhang, and Heyan Huang. 2024. Teaching large language models to translate on low-resource languages with textbook prompting. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15685–15697, Torino, Italia. ELRA and ICCL.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751.

Marcin Junczys-Dowmunt, Alexandra Birch, Roman Grundkiewicz, Kenneth Heafield, Tomasz Dwojak, Hieu Hoang, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André Martins, and Alexandra F. Birch. 2018. Marian: Fast neural machine translation in c++. In *Proceedings of ACL 2018: System Demonstrations*, pages 116–121. Association for Computational Linguistics.

LCM-team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, Safiyyah Saleem, and Holger Schwenk. 2024. Large concept models: Language modeling in a sentence representation space. *Computing Research Repository*, arXiv:2412.08821. Version 2.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. pages 3045–3059.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M. Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. Crosslingual generalization through multitask finetuning. pages 15991–16111.

NLPH. 2019. Svlm-hebrew-wikipedia-corpus. https://github.com/NLPH/SVLM-Hebrew-Wikipedia-Corpus.

Norod78. 2023. Hebrewstageandlyricswithnewlines. https://huggingface.co/datasets/Norod78/HebrewStageAndLyricsWithNewLines.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterfusion: Non-destructive task composition for transfer learning. *CoRR*, abs/2005.00247.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Qwen-Team. 2024. Qwen2.5: A party of foundation models.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *CoRR*, abs/1705.08045.

Bibek Upadhayay and Vahid Behzadan. 2024. Taco: Enhancing cross-lingual transfer for low-resource languages in LLMs through translation-assisted chain-of-thought processes. In *5th Workshop on practical ML for limited/low resource settings*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Appendix

Model Training Details

Fine-Tuning Configuration

Our implementation utilized parameter-efficient fine-tuning with the following configuration:

- **Optimizer:** AdamW ($\eta = 2 \times 10^{-5}$, weight decay = 0.01)
- **Training:** Max sequence length = 128, gradient clipping = 1.0
- **Model-specific optimizations:** Customized padding token handling for BLOOMZ, and Qwen architectures

From-Scratch Training

For training models without pre-trained weights:

- **Optimizer:** AdamW ($\eta = 5 \times 10^{-4}$, weight decay = 0.01)
- **Training strategy:** Higher learning rate, increased gradient accumulation (8 steps)
- **Efficiency:** Mixed precision (FP16) with dynamic loss scaling

Soft Prompt Training

The soft prompt approach consisted of:

- **Architecture:** Trainable prompt embeddings ($n = 30$) with frozen LLM parameters
- **Optimization:** AdamW ($\eta = 5 \times 10^{-4}$) with cosine schedule and 100 warmup steps
- **Implementation:** Input embeddings concatenation ($E_{prompt} \oplus E_{input}$) with extended attention masks

TALL Training Process

The training of our TALL architecture employed specific techniques for efficient cross-lingual learning:

- **Token-Focused Loss:** Training focused exclusively on predicting the final token of target sequences, with a specialized loss function that isolates the last non-padded position in each sequence
- **Optimization:** AdamW optimizer with cosine annealing schedule (η decaying over epochs)
- **Evaluation Metrics:** Tracked per-token accuracy, loss, and perplexity with focused evaluation on final token prediction
- **Checkpoint Management:** Maintained best-model checkpoints based on evaluation loss, with gradient norm monitoring for training stability

Evaluation Framework

Our comparative evaluation used a unified framework for all models:

- **Task:** Single-token Hebrew word prediction at sentence endings
- **Approaches:** Direct Hebrew, Naive Translation, From-scratch, Fine-tuned, Soft Prompt, and TALL implementations
- **Inference:** Temperature-controlled sampling ($T = 0.7$) with top-k (50) and top-p (0.95) filtering
- **Metrics:** Per-token accuracy with progressive tracking across dataset samples

Each training approach prioritized parameter efficiency by optimizing only the minimal necessary subset of parameters. All evaluation results, along with the full evaluator code, are available in our repository.

Metric	bloomz TALL	Qwen TALL
Total Parameters	883,537,920	799,239,680
LLM Only Parameters	559,214,592 (63.3%)	494,032,768 (61.8%)
Trainable Parameters	126,786,048 (14.35%)	107,669,632 (13.47%)

Table 2: Overall Model Statistics for bloomz TALL and Qwen TALL. The “LLM Only Parameters” row represents the sum of LLM Embeddings and Main LLM parameters.

Module	Total Params	Trainable Params	Notes
HE-EN Encoder	138,341,376	0	Frozen encoder
LLM Embeddings	256,901,120	0	Frozen embedding layer
Autoencoder 1	4,203,520	4,203,520	Two-layer MLP (1024 $\rightarrow$ 2048, 2048 $\rightarrow$ 1024)
Custom Decoder 1	101,828,608	101,828,608	Trainable decoder module
Main LLM	302,313,472	0	Frozen main LLM (BloomModel)
Autoencoder 2	1,577,472	1,577,472	Two-layer MLP (1024 $\rightarrow$ 1024, 1024 $\rightarrow$ 512)
Custom Encoder 2	19,176,448	19,176,448	Trainable encoder module
EN-HE Decoder	59,195,904	0	Frozen decoder module
LM Head	33,709,568	0	Final linear mapping

Table 3: Module-wise Breakdown for bloomz TALL.

Module	Total Params	Trainable Params	Notes
HE-EN Encoder	138,341,376	0	Frozen encoder
LLM Embeddings	136,134,656	0	Frozen embedding layer
Autoencoder 1	3,448,704	3,448,704	Two-layer MLP (1024 $\rightarrow$ 1792, 1792 $\rightarrow$ 896)
Custom Decoder 1	83,598,080	83,598,080	Trainable decoder module
Main LLM	357,898,112	0	Frozen main LLM (Qwen2Model)
Autoencoder 2	1,446,400	1,446,400	Two-layer MLP (896 $\rightarrow$ 1024, 1024 $\rightarrow$ 512)
Custom Encoder 2	19,176,448	19,176,448	Trainable encoder module
EN-HE Decoder	59,195,904	0	Frozen decoder module
LM Head	33,709,568	0	Final linear mapping

Table 4: Module-wise Breakdown for Qwen TALL.