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

Anonymous ACL submission

Abstract

Large Language Models (LLMs) excel in high-resource languages but struggle with low-resource languages due to limited training data and linguistic diversity. This paper presents *TALL* (**T**rainable **A**rchitecture for Enhancing **LL**M Performance in **L**ow-Resource Languages), a novel approach designed to bridge this gap. The key innovation of *TALL* lies in its integration of three pre-trained models: a high-resource LLM and two bilingual translation models. By transforming inputs from low-resource languages into high-resource language representations, *TALL* leverages the robust reasoning capabilities of the LLM. Subsequently, it refines the output through dimensional alignment layers and custom transformers, enabling accurate decoding into the target low-resource language.

We validate *TALL* through experiments on Hebrew, chosen for its rich morphology, complex syntax, and limited annotated datasets. To ensure a realistic evaluation, the experiments utilized models with different levels of exposure to Hebrew. Results demonstrate significant improvements in accuracy, showcasing the effectiveness of *TALL*'s modular design and trainable alignment layers. This architecture offers a scalable and adaptable framework for cross-lingual transfer and improved processing in low-resource language settings, with potential applicability to a wide range of languages and NLP tasks. Furthermore, *TALL* employs a parameter-efficient training strategy, freezing pre-trained components while training only lightweight modules, thus balancing computational efficiency with performance gains.[1]

1 Introduction

Large Language Models (LLMs) have significantly advanced the field of natural language processing (NLP), particularly for high-resource languages like English. However, their performance in low-resource languages, remains limited due to challenges like sparse data and linguistic complexity. These challenges often result in uncertainty in model predictions and reduced accuracy.

Low-resource languages, which lack extensive annotated datasets and robust pre-trained models, present unique hurdles for LLMs. As illustrated in Figure 1, there is a significant imbalance between the number of speakers of a language and the amount of digital content available, such as Wikipedia articles. This disparity highlights the data scarcity problem that limits the effectiveness of LLMs in low-resource languages.

Existing methods, such as transfer learning and data augmentation, have attempted to bridge the gap, but they often fail to fully exploit the full potential of the LLM.

This paper addresses this gap by introducing *TALL* (**T**rainable **A**rchitecture for Enhancing **LL**M Performance in **L**ow-Resource Languages), a novel architecture that systematically harnesses the strengths of LLMs trained on high-resource languages. The key idea behind *TALL* is to leverage the strengths of three pre-trained models: the LLM, which excels in processing high-resource languages, and two translators, which are already trained on extensive bilingual data. By integrating these models, *TALL* transforms inputs in a low-resource language into an intermediate high-resource representation, utilizes the LLM for reasoning and prediction, and then refines the output through the second translator to generate accurate and fluent text in the low-resource language. By integrating a sequence of carefully designed transformation steps, *TALL* ensures that the rich linguistic knowledge of high-resource language models is leveraged while bridging representational gaps through trainable alignment components.

TALL employs a modular pipeline consisting of few stages, including pre-trained translators, di-

[1] Code is available in an anonymous repository: https://anonymous.4open.science/r/TALL-28B2/.

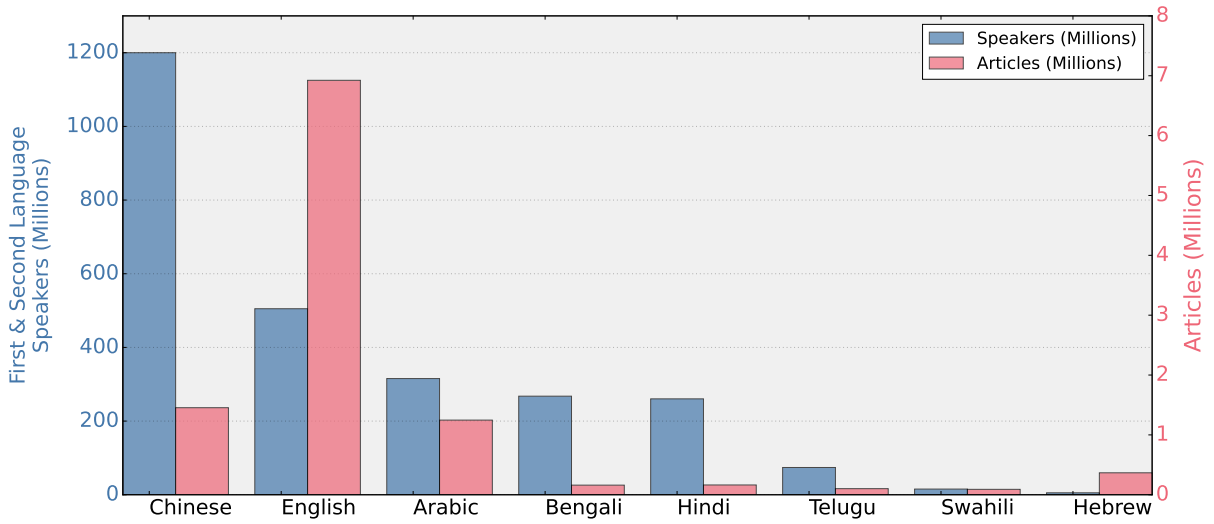

Figure 1: **Speakers vs. Wikipedia articles by language.** A comparison of the number of speakers and the number of Wikipedia articles for various languages. This figure highlights the significant disparity between the number of native speakers and the amount of digital content available in different languages. This imbalance underscores the challenges faced by low-resource languages in the digital space and emphasizes the need for advanced architectures like *TALL* Data represents the total number of first and second language speakers and Wikipedia articles. Source: (Wikimedia Foundation, 2024).

mension alignment autoencoders, and custom transformer components. Notably, these components are designed to minimize computational overhead by freezing large pre-trained models and only training lightweight adapter layers.

We present *TALL* as a scalable and adaptable solution for improving LLM performance in low-resource languages, validated through experiments on Hebrew tasks. We chose Hebrew as a representative low-resource language due to its rich morphology, complex syntax, and limited availability of annotated datasets. For our experiments, we utilized models with different levels of exposure to Hebrew data, ensuring a realistic evaluation of *TALL*'s effectiveness in low-resource language settings. We demonstrate the effectiveness of trainable dimension alignment and custom transformers in facilitating cross-lingual information flow and improving task-specific accuracy. The paper is organized as follows: Section 2 reviews related work, while Section 3 details the *TALL* architecture. Section 4 focuses on its implementation and Hebrew-specific training. Experimental results are presented in Section 5, followed by the conceptual foundations in Section 6. Section 7 discusses future research directions, and Section 8 concludes with key findings and implications.

2 Related Work

Improving the performance of Large Language Models (LLMs) in low-resource languages is a critical challenge in the field of natural language processing (NLP). Several approaches have been explored to address this issue, ranging from fine-tuning techniques to in-context learning strategies.

One prominent approach is few-shot in-context learning (ICL) (Cahyawijaya et al., 2024), which allows LLMs to perform tasks in low-resource languages by utilizing a few examples provided in the input context. Recent work has shown that cross-lingual in-context learning (X-ICL) can leverage examples from high-resource languages to improve performance in low-resource languages. This method demonstrates that while ICL can be effective, its success is closely tied to the quality of alignment between the languages involved.

Another approach involves fine-tuning multilingual models specifically for low-resource languages, as explored in works like adaptM-LLM (Lankford et al., 2024). This method tailors the model to the target language pair, offering a more direct adaptation compared to general in-context learning strategies.

A recent approach introduced in PolyLM (Wei et al., 2023) explores the use of curriculum learning to enhance the multilingual capabilities of LLMs. This method involves a two-stage training pro-

cess where the model first learns general language patterns from a large dataset dominated by high-resource languages, and then fine-tunes on a curated subset with an increased proportion of low-resource languages. This strategy facilitates knowledge transfer from high-resource languages to low-resource ones, resulting in improved performance across a wide range of multilingual tasks.

A recently introduced approach that tackles multilingual language modeling at a higher level of abstraction is the Large Concept Model (LCM) (LCM-team et al., 2024). Instead of processing text on a token-by-token basis, LCM operates on sentence-level "concepts," leveraging the SONAR embedding space (Duquenne et al., 2023). SONAR provides a single vector space covering 200 languages—including text and limited speech capabilities—where semantically similar sentences are mapped to nearby points. Building on these embeddings, LCM is trained to autoregressively predict the next sentence embedding, then uses SONAR's multilingual decoder to reconstruct the output sentence in a target language. This design effectively shifts the language modeling problem from token prediction to concept-level reasoning.

LCMs emphasize broad language coverage and have demonstrated strong zero-shot generalization in tasks like abstractive summarization (XLSum), outperforming comparably sized models on languages in which LCM received no explicit instruction. By virtue of SONAR's multilingual foundation, LCM can also process low-resource languages without any additional fine-tuning.

LCMs differ from *TALL* in several ways. First, the LCM pipeline operates at a sentence-level "concept" space derived from the SONAR encoder-decoder (Duquenne et al., 2023), while *TALL* uses a more traditional token-level flow, relying on two pre-trained translation models and a central LLM. Instead of predicting the next token, LCM employs a diffusion-based generation process where sentence embeddings are gradually denoised from an initial noisy state. This allows the model to learn a probabilistic distribution over possible continuations rather than committing to a single token sequence at each step. In contrast, *TALL* processes hidden states through its transformers in a sequential manner, maintaining the standard way used in token-based language models. Second, *TALL* is relatively straightforward in terms of computational overhead, freezing large pre-trained components and introducing only lightweight alignment layers, while LCMs involve more extensive end-to-end training of the concept space. Overall, both approaches highlight the promise of multilingual architectures, but *TALL* focuses on a parameter-efficient design suitable for low-resource deployments, whereas LCMs adopt a richer sentence-level representation that can naturally generalize to multiple modalities. Continued research into LCMs' high-level, concept-based generation holds the potential for further breakthroughs in multilingual and multimodal modeling.

These studies highlight the diversity of strategies being developed to enhance LLM performance in low-resource settings, whether through in-context learning, fine-tuning, or cross-lingual methods. The architecture proposed in this paper builds on these approaches by leveraging high-resource language capabilities through a dual translation process, aiming to refine the cross-linguistic processing abilities of LLMs and further improve their performance in low-resource languages.

3 Model Architecture

3.1 Overview

In traditional encoder-decoder translation models, a sentence is translated from one language to another through an auto-regressive process. First, the encoder processes the input sentence and converts it into a sequence of hidden states, which serve as an intermediate representation of the source language. The decoder then generates the output sequence token by token in an auto-regressive manner, where each predicted token is conditioned on the previously generated tokens. Throughout this process, the decoder employs cross-attention mechanisms to attend to the encoder's hidden states, allowing it to incorporate relevant contextual information from the source sentence at each decoding step.

In *TALL*'s architecture, the first stage processes the input sentence by tokenizing it in three ways: (1) using the first translator's tokenizer for the source language (during training, the last word removed), (2) translating the input sentence and tokenizing it with the LLM's tokenizer, and (3) tokenizing the input sentence with the second translator's tokenizer as the target language.

The intermediate hidden states from the first translator (the encoder outputs) are first processed through an autoencoder to align their dimensions with the LLM. Next, the custom decoder obtains the LLM embeddings of the translated sentence to-

kenized by the LLM's tokenizer and applies cross-attention with the aligned hidden states from the autoencoder. While decoders traditionally operate in an autoregressive manner to generate output sequences, our custom decoder serves a distinct purpose: it projects inputs into the LLM's embedding space while maintaining cross-attention to the encoded low-resource language representations. This repurposing of the decoder architecture enables direct interaction between the LLM's native embedding space and the source language encodings. This process aims to provide the LLM with a high-resource language representation of the sentence while retaining attention to the original low-resource language input. Once the LLM generates its predictions, a custom encoder transforms the output into a format suitable for the high-resource-to-low-resource translator's decoder.

During training, teacher forcing is applied by providing the ground-truth target tokens from the low-resource language as inputs to the second translator's decoder, ensuring semantic alignment and accurate decoding. Crucially, the model is trained only to predict the final missing token of the source sentence. Clearly, this token is never included in the input, ensuring that the loss function explicitly optimizes for its correct generation without direct exposure during training.

3.2 Overall Structure and Trainable Components

As illustrated in Figure 2, *TALL*'s architecture consists of seven main stages, designed to leverage existing pre-trained components while minimally introducing new, lightweight layers that are trainable. This modular structure enables efficient adaptation to low-resource language tasks while preserving the linguistic knowledge embedded in pre-trained models.

The seven main stages of *TALL*'s architecture are as follows:

1. **Source Language Encoding:** The input sentence in the source low-resource language is passed through a pre-trained low-resource-to-high-resource (LR-HR) encoder. This encoder generates hidden states that represent the intermediate semantic form of the input sentence in a format interpretable by the following components.

2. **Dimension Alignment (Source to LLM):** The hidden states produced by the LR-HR encoder are passed through a dimension alignment autoencoder. This trainable layer adapts the hidden states to the dimensional requirements of the LLM, ensuring seamless integration with the following custom decoder.

3. **First Custom Decoder (LR-HR Decoder):** The custom decoder receives the LLM-tokenized translated sentence embeddings and applies cross-attention to the aligned hidden states from the autoencoder. This ensures that the LLM receives a high-resource language representation enriched with information from the original low-resource input.

4. **LLM Integration:** The processed sequence is passed to the frozen LLM, which generates the next-token prediction.

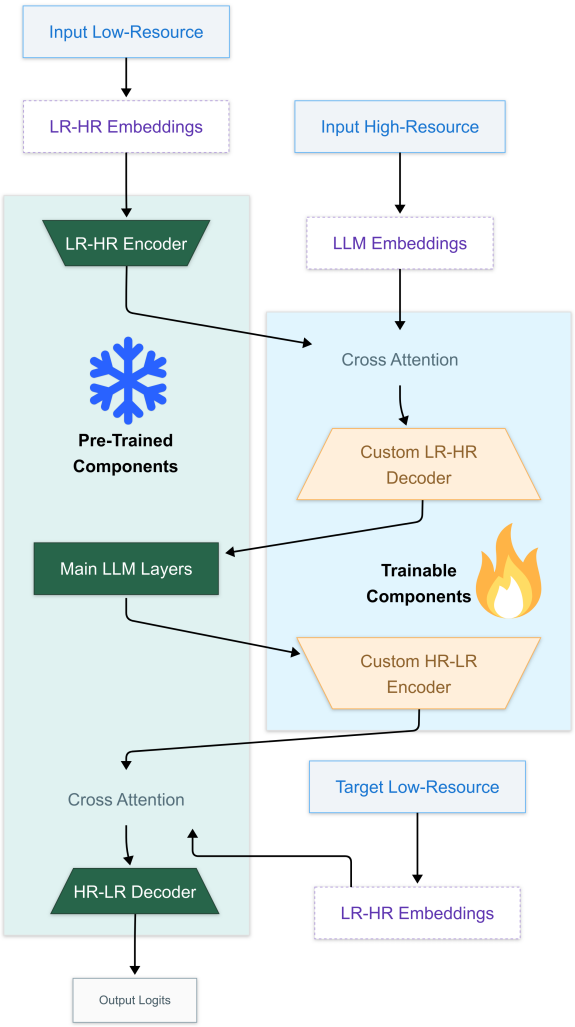

Figure 2: A conceptual overview of the *TALL* architecture, and its training data flow with teacher forcing, omitting the dimension alignment autoencoders for simplicity.

5. **Dimension Alignment (LLM to Target):** The LLM's hidden states are transformed via a second autoencoder to align with the dimension requirements of the second translator's encoder.

6. **Second Custom Encoder:** This encoder processes the aligned hidden states from the LLM, encoding the high-resource language representation into a structured format suitable for the second translator's decoder.

7. **Target Language Decoding:** The output from the second custom encoder is passed to the HR-LR decoder, which generates the final text in the target low-resource language. This decoder leverages its pre-trained capabilities to produce fluent and accurate translations by applying cross-attention between the encoded representation and the target low-resource language embeddings.

In summary, *TALL* leverages pre-trained, *frozen* models for their rich linguistic knowledge, while only fine-tuning lightweight *trainable* adapter layers. This approach keeps training efficient and manageable, enabling the model to better capture the nuanced complexities inherent in low-resource languages.

4 Implementation and Training Details

To implement *TALL* for Hebrew, we combined pre-trained Marian MT models for Hebrew-English and English-Hebrew translation with a frozen LLM, integrating the custom trainable components.

Hebrew-English Encoder: We use a pre-trained Marian MT model (Junczys-Dowmunt et al., 2018), leveraging its encoder to create a robust intermediate representation of the Hebrew input. This step provides a language-agnostic semantic representation that the subsequent components can more readily process.

Dimension Alignment Autoencoders: Two distinct autoencoders (one for the Hebrew-English to LLM alignment and another for the LLM to English-Hebrew alignment) ensure a smooth interaction between models with different architectural dimensions. Each autoencoder is composed of an encoder and a decoder stack of fully-connected layers, layer normalization, and GELU activations, enabling projection of input hidden states into the target dimensionality. These autoencoders can be pre-trained independently as regular autoencoders are trained; once trained, only their encoder components need to be retained for alignment within the *TALL* pipeline. As a result, the representations remain semantically meaningful after transformation, while allowing for efficient integration and fine-tuning within the overall architecture.

Custom Transformers: *TALL*'s architecture integrates two custom transformer-based modules derived from Marian configurations:

- **Custom Decoder 1:** Positioned after the dimension alignment from Hebrew-English space to LLM space, this Marian Decoder conditions on English tokens and cross-attends to the dimension-aligned encoder states.

- **Custom Encoder 2:** After being processed by the LLM and aligned into the English-Hebrew dimension space, this Marian Encoder further refines the representations before passing them into the English-Hebrew Marian decoder.

LLM Integration: The central LLM component is embedded between the two custom transformer modules. By positioning the LLM after the input has been converted into English-like representations, we leverage the LLM's extensive training in English, enabling it to perform complex reasoning and linguistic manipulations. The LLM's improved hidden states are then adapted back into a representation suitable for the English-Hebrew decoder.

Final Decoding and Output: The final Hebrew output is generated by the English-Hebrew Marian decoder (Junczys-Dowmunt et al., 2018). This ensures that the final tokens are produced in fluent, contextually appropriate Hebrew, benefiting from the entire chain of transformations and enhancements provided by the preceding components.

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.

4.1 Training Procedure

The training process for *TALL* focuses on refining the model to generate Hebrew output while leveraging English intermediate representations. The training data consisted of 256,000 Hebrew sentences sourced from news articles found online.

The dataset underwent standard preprocessing, including the removal of duplicate sentences, extraneous symbols, and filtering for sentence lengths between 5 and 30 words.

Batched data samples are prepared by tokenizing Hebrew input sentences, their corresponding English translations, and the Hebrew target outputs. The model is trained using teacher forcing, where the ground-truth target tokens are provided as decoder inputs. The loss is computed using cross-entropy, optimizing only for the final missing token in the sequence.

Training is performed using the AdamW optimizer with a cosine annealing learning rate scheduler. The initial learning rate is set to 5×10^{-5} for the main model and 1×10^{-3} for the autoencoders. Gradient clipping and mixed precision training are applied to stabilize training. The best model is selected based on evaluation loss, ensuring optimal performance.

Figure 3 (see the appendix) shows the training dynamics, illustrating the improvement in accuracy and reduction in perplexity over time.

5 Comparative Experiment and Results

To test the performance of *TALL*, we conducted a comparative experiment against alternative approaches. The objective was to assess our model's ability to accurately predict a missing Hebrew word at the end of a given sentence.

5.1 Experimental Setup

The evaluation was conducted on two datasets, each consisting of 50,000 Hebrew sentences: WIKI-SLVM (from the SVLM Hebrew Wikipedia Corpus (svl)) and NEWS, a test dataset originating from the same domain and source as the training data but containing unseen sentences. For each sentence, the final word was removed, and the task was to predict the missing Hebrew word. We compared *TALL* with Direct Hebrew, Naive, Soft Prompt, and a Fine-Tuned Baselines using bloomz-560m (Muennighoff et al., 2023) and QWEN2.5-0.5b (Qwen-Team, 2024). These models have different levels of exposure to Hebrew, as reflected in the Direct Hebrew baseline. Bloomz-560m has very little prior exposure, making it a strong benchmark for testing performance in a very low-resource setting. QWEN2.5-0.5b has high exposure to Hebrew, allowing us to evaluate *TALL*'s effectiveness across varying degrees of low-resource language proficiency. By evaluating *TALL* on these models, we gain insights into its adaptability across languages with minimal or partial training data availability.

Direct Hebrew Approach: This baseline approach attempts to predict the missing Hebrew word directly using the unadapted LLM. The model simply receives the truncated Hebrew sentence as input and generates a continuation. Since Bloomz-560m has never seen Hebrew, its performance is minimal, serving as a lower bound for Hebrew token prediction in a zero-resource setting

Naive Approach: The naive pipeline first translates the truncated Hebrew sentence (with the final word removed) into English, then uses the English LLM to predict the next token, thereby extending the English sentence. This entire, now-completed English sentence is then translated back into Hebrew, and the final token of the resulting Hebrew sequence is taken as the predicted word. While this approach is straightforward, it overlooks critical linguistic differences between Hebrew and English. For instance, Hebrew adjectives typically follow the nouns they modify, whereas English adjectives often precede nouns, leading to mismatched word orders and unnatural predictions. Moreover, other morphological and syntactic discrepancies between the two languages further degrade the quality and relevance of the predicted Hebrew output.

Soft Prompt Approach: In this method, we train a set of soft (learned) parameters—known as a *soft prompt*—to guide the English LLM towards producing English words that translate correctly into Hebrew. (Lester et al., 2021) Unlike a naive zero-shot pipeline, this approach fine-tunes a minimal set of parameters so that the LLM can generate English outputs that are semantically aligned with the missing Hebrew word. This approach leverages parameter-efficient tuning techniques to better adapt the model to the final prediction task.

Fine-Tuned Approach: To provide a stronger baseline, we fine-tuned the models on the same Hebrew dataset used for *TALL*, consisting of news articles formatted for causal language modeling. Each model was fine-tuned using supervised training with teacher forcing and cross-entropy loss, following the standard approach for causal LLMs.

The training was conducted with a learning rate of 2×10^{-8} and a batch size of 6, ensuring stable optimization while adapting the models to Hebrew

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
	TALL	**5.59**	**5.15**
NEWS (Test Set)	Direct Hebrew	0.08	2.83
	Fine-tuned	0.42	4.14
	Naive	2.27	3.02
	Soft prompt	2.80	2.24
	TALL	**6.10**	**4.93**

Table 1: Comparison of approaches on predicting the missing Hebrew word across datasets. *TALL*, soft prompt tuning, and fine-tuning were trained on the News train dataset. Therefore, fine-tuned models show improved performance only on the NEWS test set, but, in fact, perform worse on WIKI-SLVM. However, *TALL* achieves strong results across both datasets, highlighting its generalization ability.

text. The models were trained using standard efficiency techniques to prevent overfitting.

***TALL* Approach:** This represents our proposed *TALL* architecture. By fully integrating a Hebrew-English encoder, English LLM layers, dimension alignment components, and an English-Hebrew decoder, the custom model is specifically trained to produce accurate Hebrew completions. The architecture is designed to leverage the high-resource English competence of the LLM, while ensuring that the outputs align with the Hebrew target domain through the dual-translation mechanism and integrated alignment strategies.

5.2 Results and Analysis

The results, summarized in Table 1, highlight key differences between the evaluated approaches and demonstrate the effectiveness of *TALL*. Several important insights emerge from the comparison:

1. **Necessity of Cross-Lingual Strategies:** The Direct Hebrew approach, which relies on models with varying levels of Hebrew exposure, produces varying results. Bloomz-560m, which has never seen Hebrew, performs poorly, while QWEN2.5-0.5b, with some exposure, performs slightly better. This highlights the effectiveness of *TALL*'s cross-lingual approach, demonstrating that well-structured architectures can enhance performance even for models with minimal or moderate exposure to the target language.

2. **Incremental Gains with Translation and Soft Prompts:** The Naive and Soft Prompt approaches show slight but limited improvements. The Naive approach benefits from leveraging an English LLM's stronger token prediction abilities but introduces structural mismatches due to differences in Hebrew and English syntax. While we attempted to optimize soft prompts to guide the LLM toward better Hebrew predictions, we did not achieve substantial improvements. This suggests that soft prompts alone may not be an effective method for adapting models to low-resource languages without additional fine-tuning or structural modifications.

3. **Fine-Tuning Insights** All models improved on the News test dataset, demonstrating that fine-tuning effectively helped them adapt to this specific domain. However, their performance on the Wikipedia dataset declined slightly, likely due to a domain shift or mild forgetting. Since these models were probably pretrained on Wikipedia, fine-tuning on the News dataset may have interfered with their ability to generalize back to Wikipedia-based examples.

6 Conceptual Foundations

The theoretical underpinning of the *TALL* architecture draws upon established principles in natural language processing (NLP), transfer learning, and cross-lingual modeling. Its design leverages the following key theoretical foundations:

Multilingual and Cross-Lingual Transfer: A central idea behind *TALL* is to harness the substantial linguistic knowledge embedded in Large

Language Models (LLMs) trained predominantly on high-resource languages (e.g., English). Previous research in multilingual representations (Conneau et al., 2020; Li et al., 2024) has shown that representations learned for one language can facilitate learning in another language, especially if the languages share underlying semantic structures or scripts.

Representation Learning and Alignment: *TALL* relies on dimension alignment autoencoders to project encoder hidden states between different model architectures and representation spaces. The theoretical motivation for such intermediate layers comes from the field of representation learning (Bengio et al., 2013), where learned vector spaces encode semantic information. By learning a mapping function (autoencoder) that aligns these spaces, we ensure that different model components can interchange information without loss of semantic fidelity.

Teacher Forcing: Teacher forcing (Williams and Zipser, 1989) is employed to improve stability and convergence during training by providing the ground-truth target tokens. Although originally developed for traditional language models and sequence-to-sequence models, the concept is equally applicable here.

Collectively, these theoretical considerations guide the design of *TALL*, informing the decisions on the model architecture, training objectives, and evaluation strategies.

7 Future Work

While *TALL* demonstrates promising results in enhancing LLM performance on low-resource languages like Hebrew, several avenues remain open for future exploration:

Extending to Other Low-Resource Languages: Future work can apply *TALL* to a broader set of languages, including morphologically rich and agglutinative languages. By experimenting with different language pairs and families, researchers can assess the generality and scalability of *TALL's* approach.

Incorporating Additional Modalities: Beyond text-only representations, future research could integrate multimodal inputs (e.g., audio or images) to enrich the language representations. For instance, using speech or image annotations could further ground language understanding and potentially enhance performance in low-resource settings, where textual data may be scarce.

Refining Dimension Alignment Mechanisms: Although our dimension alignment autoencoders effectively map between representation spaces, future work could explore more advanced alignment techniques.

8 Conclusion

This work introduced *TALL*, a strategy for improving the performance of Large Language Models in low-resource languages by leveraging high-resource language capabilities. The key innovation lies in the dual-translation pipeline, where input is transformed into a high-resource language space—capitalizing on the strengths of well-trained English-centric LLMs—before being re-translated into the target low-resource language.

Through careful integration of custom transformer components, dimension alignment autoencoders, and teacher forcing, *TALL* demonstrates improved performance on Hebrew tasks, as measured by reduced perplexity, increased accuracy. The presented results validate the theoretical foundations of cross-lingual transfer, representation alignment, and incremental learning, and highlight the importance of leveraging existing LLM capabilities to bridge resource gaps.

While challenges remain—particularly in extending the approach to a wider array of languages and exploring multimodal enhancements—*TALL* offers a meaningful step towards more inclusive and accessible NLP technologies. By continuing to refine this approach, and by engaging in broader evaluations and more diverse linguistic landscapes, the field can push closer to a future where high-quality language models support all languages, regardless of their resource availability.

Limitations

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

Limited Support for Key-Value Caching: Typical LLM-based pipelines leverage key-value caching to speed up inference during token-by-token generation, enabling efficient autoregressive decoding. In the current design, however, the integration of multiple translation and alignment mod-

ules complicates straightforward caching of intermediate states. This results in less efficient inference times compared to standard LLMs. Nonetheless, future work could address this concern by exploring partial caching techniques or optimizing the custom components to make them more cache-friendly.

Reduced Per-Token Learning Signal: Conventional LLM training benefits from predicting every next token in a sequence, reinforcing language modeling abilities at all time steps. By contrast, *TALL*'s setup focuses on predicting only the final token of the low-resource sentence after the dual-translation process. While this design choice effectively zeroes in on specific task objectives, it constrains the model's training signal to a single token per example. The consequence is a more limited reinforcement of internal representations. Potential solutions include curriculum learning strategies or multi-stage training protocols, where initial training phases could incorporate additional per-token predictions.

Adapter Layer Bottleneck: *TALL* is heavily dependent on dimension alignment layers and other adapter modules to integrate multiple pretrained models. Although these adapters are designed to be lightweight, they could introduce an information bottleneck, limiting the free flow of semantic features through the pipeline. Empirically, this may reduce maximum achievable performance. Future refinements may focus on more expressive adapter architectures, non-linear alignment techniques, or even joint fine-tuning strategies that partially relax the strict boundaries between pre-trained components.

In summary, while these limitations highlight areas that may impact efficiency and learning capacity, they also point to clear and tractable research directions. By addressing caching techniques, enhancing the learning signal, and refining adapter designs, subsequent iterations of *TALL* could further improve speed, adaptability, and accuracy in low-resource language settings.

References

Svlm hebrew wikipedia corpus. https://github.com/NLPH/SVLM-Hebrew-Wikipedia-Corpus. [Accessed December, 2024].

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. pages 8440–8451.

Paul-Ambroise Duquenne, Holger Schwenk, and Benoît Sagot. 2023. SONAR: sentence-level multimodal and language-agnostic representations. *CoRR*, abs/2308.11466.

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.

Séamus Lankford, Haithem Afli, and Andy Way. 2024. adaptmllm: Fine-tuning multilingual language models on low-resource languages with integrated LLM playgrounds. *CoRR*, abs/2403.02370(12):638.

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.

Mingda Li, Abhijit Mishra, and Utkarsh Mujumdar. 2024. Bridging the language gap: Enhancing multilingual prompt-based code generation in llms via zero-shot cross-lingual transfer. *CoRR*, abs/2408.09701.

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.

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

Xiangpeng Wei, Haoran Wei, Huan Lin, Tianhao Li, Pei Zhang, Xingzhang Ren, Mei Li, Yu Wan, Zhiwei Cao, Binbin Xie, Tianxiang Hu, Shangjie Li, Binyuan Hui, Bowen Yu, Dayiheng Liu, Baosong Yang, Fei Huang, and Jun Xie. 2023. Polylm: An open source polyglot large language model. *CoRR*, abs/2307.06018.

Wikimedia Foundation. 2024. List of wikipedias by speakers per article. [Accessed December, 2024].

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Appendix

Training Performance Analysis

To analyze the training process, we track key performance metrics, including accuracy and perplexity, over multiple training steps. Figure 3 illustrates the progression of training:

- **Accuracy:** The accuracy of the final token prediction steadily improves, indicating that the model is learning to make more confident and correct predictions.
- **Perplexity:** The perplexity decreases over time, with occasional fluctuations. These fluctuations may correspond to points where the model updates its internal representations significantly, adapting to new patterns in the training data.

Overall, these results demonstrate that the training process effectively leverages English intermediate representations and teacher forcing, leading to a robust ability to predict missing Hebrew tokens with increasing accuracy.

Figure 4 presents the accuracy progression during the evaluation of 50,000 sentences from the WIKI-SLVM dataset using the QWEN model.

All evaluation results, along with the full evaluator code, are available in our repository[2].

[2]The repository, including evaluation scripts and full experimental results, is available at `https://anonymous.4open.science/r/TALL-28B2/`.

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.

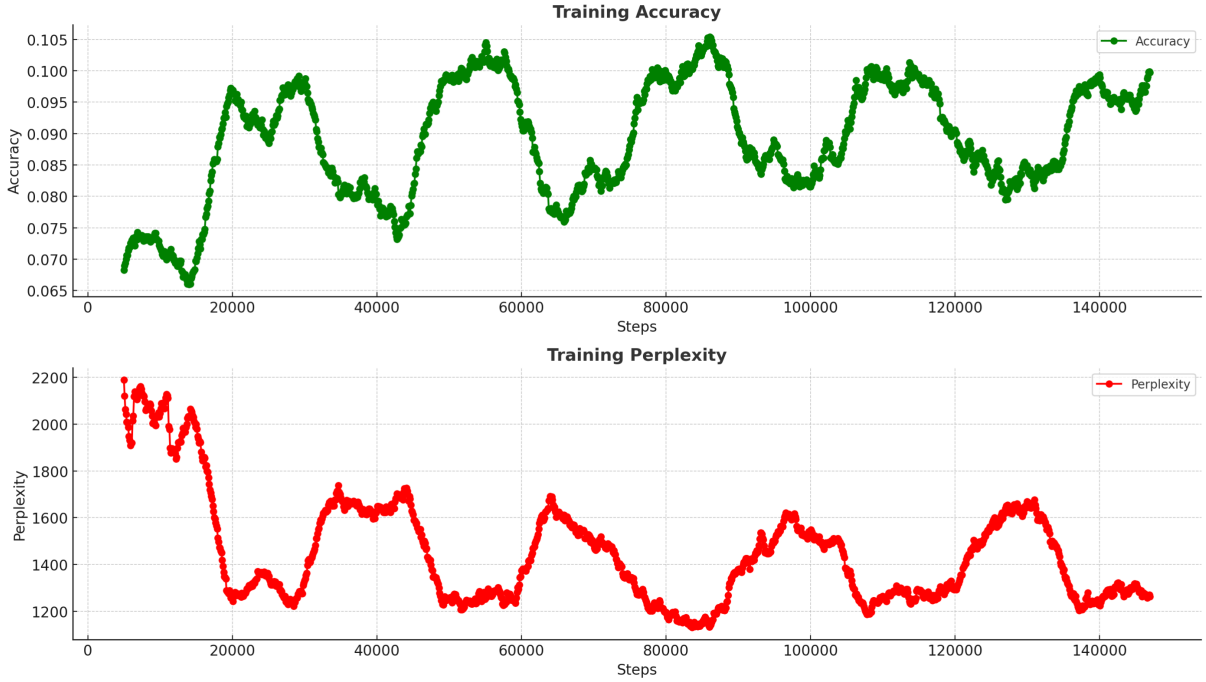

Figure 3: Training dynamics over time, showing accuracy (top) and perplexity (bottom) as functions of training steps. Metrics are smoothed using a moving average. The model increasingly improves its predictions of the final token, demonstrating the effectiveness of the training procedure.

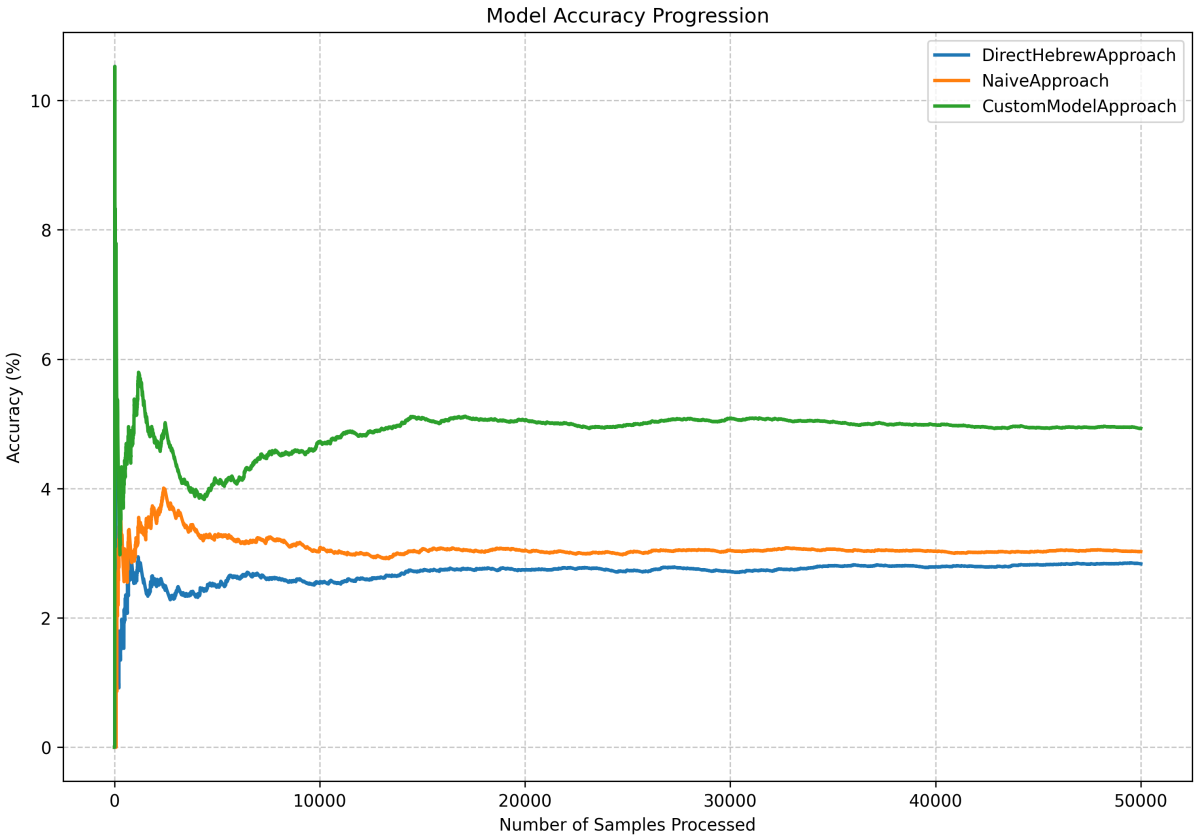

Figure 4: Accuracy progression during the evaluation of 50,000 sentences from WIKI-SLVM on QWEN. The steady improvement of predictions across the evaluation set highlights the effectiveness of the evaluation method.