Bridging the Language Gap: Dynamic Learning Strategies for Improving Multilingual Performance in LLMs

Somnath Kumar[*]   **Vaibhav Balloli**[*]   **Mercy Ranjit**   **Kabir Ahuja**
Sunayana Sitaram   **Kalika Bali**   **Tanuja Ganu**   **Akshay Nambi**
Microsoft Research India
akshayn@microsoft.com

Abstract

Large language models (LLMs) have revolutionized various domains but still struggle with non-Latin scripts and low-resource languages. This paper addresses the critical challenge of improving multilingual performance without extensive fine-tuning. We introduce a novel dynamic learning approach that optimizes prompt strategy, embedding model, and LLM per query at runtime. By adapting configurations dynamically, our method achieves significant improvements over static, best and random baselines. It operates efficiently in both offline and online settings, generalizing seamlessly across new languages and datasets. Leveraging Retrieval-Augmented Generation (RAG) with state-of-the-art multilingual embeddings, we achieve superior task performance across diverse linguistic contexts. Through systematic investigation and evaluation across 18 diverse languages using popular question-answering (QA) datasets we show our approach results in 10-15% improvements in multilingual performance over pre-trained models and 4x gains compared to fine-tuned, language-specific models.

1 Introduction & Related Work

Large Language Models (LLMs), such as ChatGPT (OpenAI, 2023), Gemini (Team et al., 2023), and Claude (AI, 2023), have driven significant advancements in artificial intelligence (AI), setting new benchmarks for performance across a wide range of tasks (Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023). They excel in diverse applications, including search engines, office tools, and critical sectors like health, education, and agriculture (Shiksha, 2024; FarmerChat, 2024; KhanAcademy, 2024; M365Copilots, 2023). By transforming workflows, LLMs are rapidly becoming essential in real-world systems, revolutionizing approaches to complex tasks across domains.

However, despite their widespread success, LLMs remain predominantly optimized for English and Latin-script languages, creating significant limitations in non-English and multilingual environments (Ahuja et al., 2023a,b; Khanuja et al., 2021).

Although efforts have been made to extend LLM capabilities to low-resource languages through fine-tuning and smaller, specialized

Method	Accuracy
LLama2 70B	8.5
Mistral 7B instruct	29.6
Cohere	78.8
Palm2	76.5
GPT3.5	60.1
GPT4	71.5
TULR-XXL	**84.6**

Table 1: Performance comparison across various models for `TyDiQA`.

models (Gala et al., 2024; Abdin et al., 2024), their performance in multilingual tasks still lags behind state-of-the-art (SOTA) multilingual models like TULRv6 and XLMR (Goyal et al., 2021). A comparative analysis, shown in Table 1, highlights this performance gap across LLMs such as GPT-3.5, GPT-4, Palm2, and LLaMA2 on the `TyDiQA` multilingual QA dataset, where they consistently underperform relative to models specifically designed for multilingual tasks.

To bridge the performance gap in multilingual LLMs, two key research directions have emerged (Qin et al., 2024; Huang et al., 2024). The first focuses on enhancing foundational models with additional multilingual data, such as Cohere AI's Aya 101 (Üstün et al., 2024), which curates instructions across 99 languages. However, this approach has limitations. Data scarcity for low-resource languages remains critical, leading to suboptimal performance during pre-training (Hämmerl et al., 2022; Wang et al., 2020). Additionally, the computational cost of training models across multiple languages is prohibitive, making fine-tuning impractical for many researchers (Qin et al., 2024; Liu et al., 2024). Even after fine-tuning, models

---

[*]Equal Contributions

9209

often struggle to generalize beyond the languages or tasks they were trained on, as seen in Sarvam 2B (Sarvam, 2024; AI, 2024; Xu et al., 2024), which underperforms on Indian languages excluded from its training data.

The second direction aims to improve pre-trained LLMs through optimized external configurations, focusing on **(1) Prompt Strategies**, **(2) Embedding Models**, and **(3) Model Selection**. While various prompt strategies (e.g., Chain-of-Thought, cross-lingual prompts) have improved specific tasks (Wei et al., 2022; Shi et al., 2022; Nguyen et al., 2024; Li et al., 2024), no single approach works consistently across all languages (Zhao and Schütze, 2021; Huang et al., 2022; Fu et al., 2022; Lin et al., 2021). For instance, Chain-of-Thought prompting improves reasoning in English (Wei et al., 2022; Lai and Nissim, 2024) but struggles with languages like Finnish or Tamil.

Embedding models like OpenAI's text-embedding-3 and Cohere's multilingual v3.0 (OpenAI, 2024; Cohere, 2024) have significantly boosted multilingual performance in question-answering tasks, yet selecting the right embedding remains challenging as performance varies across languages. Furthermore, the release of new LLMs exacerbates the **Model Selection Dilemma**, where model performance varies widely across languages and retraining models for each version is impractical due to resource constraints.

Most prior work relies on **static configurations**—where a single prompt strategy or embedding is applied to specific tasks (Qin et al., 2024; Huang et al., 2024). This falls short in multilingual contexts due to linguistic diversity. For example, strategies optimized for English may fail in languages like Japanese or Arabic, while embeddings designed for Indo-European languages may struggle with tonal languages like Mandarin (Ahuja et al., 2023a,b). These challenges emphasize the need for real-time, dynamic approaches that adapt to each language's unique requirements without requiring costly retraining. This is crucial in multilingual settings where a one-size-fits-all configuration is unlikely to succeed.

Our work addresses this gap with a dynamic runtime selection framework. Unlike static configurations, our approach dynamically selects the best combination of prompt, model, and embedding for each query based on the task and language. For instance, a French query may use a model fine-tuned for Western European languages

and a prompt strategy that handles gendered nouns, while a Hindi query might employ a strategy suited for free word order and compound verbs. This real-time adaptability ensures each query receives an accurate, context-aware response tailored to its linguistic structure.

Our key contributions are twofold:

1. **Hybrid Approach:** We integrate LLM-generated responses with multilingual embeddings in a Retrieval-Augmented Generation (RAG) setup. This hybrid model improves document retrieval and text generation, enhancing coherence and relevance, while addressing performance gaps in multilingual tasks. By using language-specific embeddings, we bridge cross-lingual understanding, achieving superior results, especially in low-resource languages.

2. **Dynamic Learning Framework:** We introduce a dynamic configuration framework that optimizes runtime selection of prompts, LLMs, and embeddings. Powered by a lightweight transformer, this framework generalizes across tasks, languages, and datasets without retraining for each domain, reducing computational overhead. By selecting optimal configurations in real time, it ensures adaptability to new LLMs, embedding models, or prompt strategies as they emerge.

Our hybrid dynamic learning architecture combines LLMs with convolutional layers, supporting both offline and online learning. It addresses three key needs: **(i) Offline Learning**, leveraging ground-truth data for optimal configuration in controlled settings; **(ii) Online Adaptability**, adjusting dynamically to real-time inputs and distribution shifts; and **(iii) Language and Dataset Flexibility**, maintaining high performance across diverse linguistic and contextual variations.

We validate our approach using the `IndicQA` and `TyDiQA` QA datasets, which encompass 18 languages. Our framework demonstrates a 10-15% improvement in multilingual performance compared to existing pre-trained LLMs, and significantly outperforms fine-tuned models optimized for specific languages, such as Ambari (HuggingFace, b), Airavata (HuggingFace, a), and Navarasa (HuggingFace, c), with performance gains exceeding 4x. These results demonstrate the superiority of our dynamic approach, which outperforms both static models and fine-tuned, language-specific solutions.

While we evaluate on QA tasks, our dynamic

Language: Kn (Kannada)
Question: ನೇಪಾಳವು ಯುದ್ಧದಲ್ಲಿ ಯಾರಿಂದ ಸೋಲಿಸಲ್ಪಟ್ಟಿತ್ತು?
Translate_En: Nepal was defeated in war by whom?
GT Answer: ಆಂಗ್ಲ   **Translate_En_Answer:** English
Generated answers & F1 score: ಬ್ರಿಟಿಷ್ (0), ಬ್ರಿಟೀಷರಿಂದ (0), ಆಂಗ್ಲರು (0)

Language: Mr (Marathi)
Question: चांद्रयान १ हे कुठल्या संस्थेचे चंद्रावर पहिले मोहीम आहे?
Translate_En: Chandrayaan 1 is the first mission to the moon by which organization?
GT Answer: इसोने   **Translate_En_Answer:** ISRO
Generated answers & F1 score: इसरो ने (0), इसो (0), भारतीय अंतराळ संशोधन संस्था (इसो) (0), भारतीय अंतराळ संशोधन संस्थेच्या (0), इसोने केले (0.67)

Figure 1: Examples showing the limitations in the GT answer in `IndicQA` dataset.

framework is versatile and extends to other multilingual applications. By decoupling task performance from any single model, prompt, or embedding, it provides an efficient, scalable solution for overcoming LLM limitations in non-English and low-resource languages.

2 Multilingual Tasks, Datasets & their Limitations

In this work, we focus on RAG-based Question Answering (QA) tasks, demonstrating the model's ability to deliver accurate responses by leveraging external text context.

2.1 Dataset
We utilize two prominent multilingual QA datasets that includes 18 diverse languages (see Table 2) from high to medium to low resource including Latin and Non-Latin scripts (we follow ISO 639-1 language code standards in the remaining of the paper (Wikipedia)):

Table 2: Datasets

IndicQA		TyDiQA	
Lang	# Q	Lang	# Q
as	1789	bn	180
bn	1763	te	874
gu	2017	fi	1031
hi	1547	ko	414
kn	1517	ru	1079
ml	1589	ar	1314
mr	1604	en	654
or	1680	id	773
pa	1542	sw	596
ta	1804		
te	1734		

1.IndicQA (AI4Bharat, 2022): A curated dataset in 11 Indic languages sourced from Wikipedia on topics related to Indic culture and history, comprising over 18,000 questions.

2.TyDiQA (Clark et al., 2020): This dataset covers 9 typologically diverse languages. Our experiments focus on the Gold-P task, where only the gold answer passage is provided rather than the entire Wikipedia article.

2.2 Evaluation Metrics for Multilingual QA
F1 score is the commonly used metric in QA tasks (Rajpurkar et al., 2016), compares individual words in predictions to the True Answer. While `SQuAD-F1` is standard for English QA evaluation, `MLQA-F1` (Lewis et al., 2019) offers additional preprocessing for fair multilingual evaluation, including stripping Unicode punctuations and stand-

alone articles. Hence, we adopt `MLQA-F1` as our evaluation metric.

2.3 Limitations of Current Datasets & Evaluation Approach
Many multilingual evaluation datasets were developed before the advent of Large Language Models (LLMs), posing two key challenges:

`Challenge 1: Limited Ground Truth (GT)`. These datasets usually contain only one answer per question, though multiple semantically correct answers may exist, particularly in real-world and conversational contexts.

`Challenge 2: Strict Evaluation Metrics`. The standard F1 scoring at the word level leads to significant penalties for minor variations in answers, especially when only a single GT is available.

Figure 1 demonstrates these challenges using the `IndicQA` dataset for Kannada and Marathi. Although generated responses are factually correct, they differ slightly from the single GT answer, resulting in low or zero `MLQA-F1` scores. This underscores the limitations of both the dataset's GT and the evaluation method.

One solution is to enrich GTs by including all valid alternatives, but this requires extensive and costly data collection. To overcome this, we introduce `GPTAnnotator`, leveraging an LLM (e.g., GPT-4) to validate predicted answers. This builds on previous work where GPT models are used as evaluators and annotators for diverse tasks. `GPTAnnotator` assesses predicted responses by comparing them to the original GT and outputs three options: *YES* for semantically correct answers, *NO* for mismatches, and *PARTIAL* for partial matches. `GPTAnnotator` enriches the GT with correct answers, creating a more comprehensive reference set (see Appendix 11 for prompts).

To further refine evaluations, we propose `GPTAnnotator-F1`, an F1 score calculated against the enriched GT with multiple valid answers. In contrast, the traditional `MLQA-F1` score compares predictions against the original, limited GT. Both are F1 metrics but differ in the number

9211

of answers they evaluate against (`MLQA-F1` uses a single-answer GT, while `GPTAnnotator-F1` considers multiple correct answers).

We validated GPTAnnotator-F1 by selecting 100 questions from the `IndicQA` dataset (across six languages) and comparing the results with human annotations

Figure 2: Comparison of `MLQA-F1` and `GPTAnnotator-F1`.

(`HumanAnnotator-Score`). Human annotators were native speakers and were given clear instructions for annotations. As shown in Figure 2 `MLQA-F1` scores differed from human annotations by an average of 25% (with a maximum of 51%), exposing the limitations of current GTs. In contrast, `GPTAnnotator-F1` reduced the error difference by 30%, aligning more closely to human judgment. Thus, providing a more accurate reflection of LLM performance. In the subsequent sections, we present results using both `MLQA-F1` and `GPTAnnotator-F1` metrics.

Limitations of `GPTAnnotator`: GPTAnnotator's quality depends on the LLM's performance in the target language. Designers should check benchmarks/baselines or run small-scale human evaluations to compare annotations. If discrepancies are significant, the LLM may not be suitable. Despite some limitations, our evaluations show GPTAnnotator aligns well with human annotators, proving effective across languages.

3 Prompt Strategies for Polyglot LLMs

Effective prompt design is critical for improving generative models, especially in multilingual tasks (Sahoo et al., 2024). Crafting prompts is already challenging in monolingual English (Yang et al., 2022), and becomes more complex across diverse languages due to differences in syntax, grammar, and lexicon. Various prompt strategies have been proposed, each with its advantages and limitations across languages (Ahuja et al., 2023a,b).

Chain-of-Thought prompting (Wei et al., 2022; Lai and Nissim, 2024) excels in reasoning tasks but struggles with morphologically complex languages like Korean. Self-Translation (Gao et al.,
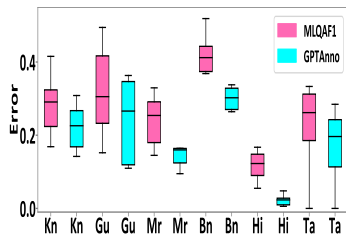
2024), where models refine responses across languages, can cause inconsistencies, particularly in low-resource languages. Linguistic Feature Prompting (Nie et al., 2024; Messina et al., 2023) encodes syntactic or semantic features directly into prompts, aiding models in languages with complex grammar. Finally, Aggregation strategies, which combine responses from multiple prompt types, offer a way to mitigate prompt-specific weaknesses (Wang et al., 2023; Lin et al., 2021).

However, no single strategy works best across all languages. Success depends on factors such as language-specific traits, task complexity, and resource availability.

Selected Strategies from Prior Work. From the variety of prompt strategies available, we selected five that consistently showed the best performance across tasks and languages. These strategies may not be universally optimal, but they provide strong results in our multilingual experiments.

1. Monolingual (Mono): Instruction, context, and examples are provided in the source language. This works well for high-resource languages but underperforms for low-resource ones (Ahuja et al., 2023a).

2. Translate-Test (Trans): Instructions and contexts are translated into English, leveraging the model's strengths in English before back-translating the output to the source language. However, translation errors can affect accuracy in low-resource languages (Agrawal et al., 2024; Ghafoor et al., 2021).

3. Similar High-Resourced Language (Sim): Roundtripping through a linguistically similar high-resource or medium-resource language (chosen based on lang2vec (Littell et al., 2017)) improves performance by capturing linguistic similarities better than direct English translation, especially for related languages. More details in Appendix 7.

4. Aggregation Source (Agg_Src): Combines responses from multiple strategies (Mono, Trans, Sim) to form a final answer in the source language. Though computationally expensive, this leads to more coherent, accurate answers (Wang et al., 2023).

5. Aggregation Translate (Agg_Trans): Aggregates responses in English before back-translating to the source language. While translation challenges exist, high-quality translation systems make this approach effective.

Other approaches like self-translation and linguistic feature-based prompting are viable but

	MLQA-F1			GPTAnnotator-F1		
	GPT-4Turbo	GPT3.5Turbo	Mixtral	GPT-4Turbo	GPT3.5Turbo	Mixtral
Mono	**0.51**	**0.43**	0.15	0.71	0.71	0.31
Trans	0.36	0.37	**0.33**	**0.80**	**0.80**	**0.68**
Sim	0.30	0.28	0.19	0.70	0.70	0.44
Agg_Src	**0.51**	**0.43**	0.20	0.73	0.73	0.39
Agg_Trans	0.35	0.38	**0.33**	0.79	0.79	**0.68**

Table 3: Performance of different Prompting strategies for `IndicQA`.

Metrics	Models	Ada	Adav3	XLMR	Cohere
MLQA-F1	GPT4T	0.51	0.5	0.54	**0.58**
	GPT3.5	0.43	0.43	0.39	**0.44**
GPTAnno	GPT4T	0.8	0.8	0.8	**0.82**
	GPT3.5	0.8	0.8	0.8	**0.81**

Table 4: Hybrid approach performance on `IndicQA`.

	MLQA-F1		GPTAnnotator-F1	
Lang	GPT3.5	GPT4T	GPT3.5	GPT4T
as	Ada	Cohere	Ada	Ada
bn	Cohere	Cohere	Ada	Ada
gu	Ada	Cohere	Cohere	Cohere
hi	Cohere	Cohere	Ada	Cohere
kn	Ada	Cohere	Cohere	Cohere
ml	Cohere	Cohere	Cohere	Cohere
mr	Ada	Cohere	Cohere	Cohere
or	Adav3	Ada	Ada	Ada
pa	Adav3	Adav3	Cohere	Ada
ta	Cohere	Cohere	Cohere	Cohere
te	Cohere	Cohere	Cohere	Cohere

Table 5: Embedding preference `IndicQA`.

	MLQA-F1		GPTAnnotator-F1	
Lang	GPT3.5	GPT4T	GPT3.5	GPT4T
ar	Ada	Adav3	Ada	Adav3
bn	Ada	Ada	Ada	Ada
en	Cohere	XLMR	Cohere	XLMR
fi	Ada	Ada	Ada	Ada
id	Ada	Adav3	Ada	Adav3
ko	Ada	XLMR	Ada	XLMR
ru	Ada	Ada	Ada	Ada
sw	Ada	Adav3	Ada	Adav3
te	Ada	Cohere	Ada	Cohere

Table 6: Embedding preference `TyDiQA`.

didn't perform consistently. We tested both zero-shot and few-shot setups, finding that few-shot examples consistently improved performance. Future advancements in example selection and in-context learning will further enhance these strategies.

Prompting Strategies Results. Our results highlight three key findings:

1. No Universal Best Strategy: No single prompt strategy works best across all models and languages. For `GPT-4Turbo` and `GPT3.5Turbo`, `Mono` and `Agg_Src` excel, while `Mixtral` favors `Trans` and `Agg_Trans`. Translation-based strategies work better for low-resource languages like Tamil and Telugu due to limited data availability.

2. Strategy Sensitivity to Metrics: Performance varies based on the evaluation metric. For example, `GPTAnnotator-F1` favors `Trans` and `Agg_Trans`, while `MLQA-F1` shows better results with `Mono` and `Agg_Src`.

3. Comparable Performance with Metric Variation: While `MLQA-F1` suggests `GPT-4Turbo` outperforms `GPT3.5Turbo`, enriching ground truth and using `GPTAnnotator-F1` reveals comparable performance, with a 28% overall improvement in accuracy for `GPT3.5Turbo`. This underscores the importance of metric selection when evaluating models.

Summary: *Prompt strategies significantly boost multilingual model performance, but no single approach is universally superior across models, metrics, or languages. The `GPTAnnotator-F1` met-ric, in particular, levels the performance gap between `GPT3.5Turbo` and `GPT-4Turbo`.*

4 Hybrid Approach: Synthesizing LLM Generation with Multilingual Embeddings

While LLMs excel in response synthesis, improving multilingual performance requires robust multilingual embeddings. GPT models, primarily trained on English data, use the default embedding model (text-embedding-ada-002, or ada), which underperforms in multilingual contexts. In contrast, state-of-the-art multilingual models like XLMR-XXL (Goyal et al., 2021) and Cohere (embed-multilingual-v3.0) (Cohere, 2024) demonstrate superior results due to their diverse language training.

We leverage a hybrid approach that combines the cross-lingual understanding of multilingual embeddings with the text-generation abilities of LLMs. We experiment with GPT's default ada embeddings, an improved variant (adav3) (OpenAI, 2024), and state-of-the-art multilingual embeddings like XLMR-XXL (Goyal et al., 2021) and Cohere v3 (Cohere, 2024).

Performance Analysis. Table 4 illustrates the maximum performance achieved by each embedding (ada, adav3, xlmr, cohere) for `GPT-4Turbo` and `GPT3.5Turbo` models across all languages and prompt strategies for `IndicQA`. Cohere, a multilingual embedding, enhances `GPT-4Turbo` performance by up to 7% and 2% compared to de-

fault ada embeddings when using `MLQA-F1` and `GPTAnnotator-F1` metrics. This indicates a significant improvement in multilingual task performance with multilingual embeddings coupled with LLM generation. While marginal improvements are observed in `GPT3.5Turbo` with multilingual embeddings, mainly due to poor LLM generation with `GPT3.5Turbo` rather than multilingual content retrieval.

Additionally, Table 5 and 6 indicates the preferred embedding for each language that yields the best performance. Generally, multilingual embeddings, particularly Cohere, are preferred for `IndicQA`. Similar trends are observed in `TyDiQA`, as detailed in Appendix 9.

Summary: *The hybrid approach boosts performance by up to 7% on the `GPT-4Turbo` model. However, there's no universal best prompt strategy, model, or embedding that performs optimally across datasets and languages.*

5 Dynamic Learning Approach to Improve Multilingual Performance

Motivation: A one-size-fits-all solution does not exist for selecting the best combination of prompt strategy, embeddings, and LLM for different languages. This raises the key question: Can we dynamically determine the optimal configuration for each query to maximize multilingual performance?

To address this, we propose a learning approach that dynamically selects the optimal configuration per query, meeting three key requirements: (i) Offline Learning: It learns the best configuration using ground truth data offline, (ii) Online Learning: It adapts in real-time, adjusting for new data and distribution shifts, and (iii) Language and Dataset Adaptability: It remains flexible across languages and datasets, ensuring robust performance.

Hybrid Architecture: Our solution combines LLMs with convolutional layers to dynamically select the best configuration across LLM models, embeddings, and prompt strategies. LLMs generate high-dimensional representations, which are fed into ND convolutional layers that extract features across dimensions, predicting task accuracy (F1 score) per query. By comparing predicted scores, we select the optimal configuration for each task and language, for both offline and online learning.

Prior efforts like LOVM (Zohar et al., 2024), (Liu et al., 2023) and HuggingGPT (Shen et al., 2024) focus on optimizing model selection for a single parameter. In contrast, our approach selects

the optimal combination of LLM model, prompt strategy, and embeddings, tackling a complex, high-dimensional search space.

In our architecture, we predict F1 scores for each configuration, generating a SoftMax output as a probability distribution. Sampling configurations from this distribution in online settings allows for controlled entropy and exploration of diverse configurations, helping mitigate bias, especially with out-of-distribution data.

Architecture details. The architecture leverages the LLaMa-2-70B-hf model for embedding generation. The traditional sampling head is replaced by a set of Conv-ND layers (Vizcaíno et al., 2021), denoted as $\mathcal{H}$, which predict the F1 Score for each configuration. The LLaMa-2-70B-hf (Touvron et al., 2023) backbone, $\mathcal{B}$, embeds the Task Description $\mathcal{T}$, into task embeddings $\mathcal{E}_T$ and configuration embeddings $\mathcal{C}_i$ into $\mathcal{E}_{C_i}$.

These embeddings are then arranged into an ND array of size $\mathcal{R}^{e \times n_1 \times n_2 \times n_3 \ldots n_m}$, where m is the number of parameters (e.g., language model, embedding model, prompt strategies, so $m = 3$). Each n_i represents the number of possibilities for each parameter (e.g., three LLMs (`GPT-4Turbo`, `GPT3.5Turbo`, `Mixtral`), four embedding models (adav2,adav3,XLMR,cohere), five prompt strategies (`Mono`, `Trans`, `Sim`, `Agg_Src`, `Agg_Trans`)). The embedding projection size e for $\mathcal{B}$ is 8192. The task embedding is broadcasted and concatenated with configuration embeddings to form a matrix of size $\mathcal{R}^{2e \times n_1 \times n_2 \times n_3 \ldots n_m}$.

We treat the embedding dimension as the number of input channels to $\mathcal{H}$ and reduce it to 1 while preserving the remaining dimensions, resulting in a matrix of size $\mathcal{R}^{1 \times n_1 \times n_2 \times n_3 \ldots n_m}$ or $\mathcal{R}^{n_1 \times n_2 \times n_3 \ldots n_m}$, representing the predicted F1 for all configurations.

$$\mathcal{E}_{T_j} \leftarrow \mathcal{B}(\mathcal{T}_j) \tag{1}$$

$$\mathcal{E}_{C_i} \leftarrow \mathcal{B}(\mathcal{C}_i) \tag{2}$$

$$\mathcal{E}_j \leftarrow \mathcal{E}_{T_j} \parallel \mathcal{E}_{C_i} \tag{3}$$

$$\hat{y} \leftarrow \mathcal{H}(\mathcal{E}_j) \tag{4}$$

Using the above equations, we obtain $\hat{y}$, which is the predicted F1 score for all combinations. To select the configuration, we either take the *argmax* or apply *softmax* and sample a particular configuration. Figure 3 illustrates inference pipeline, given the Task Description $\mathcal{T}_j$ and Configurations $\mathcal{C}_i$ to obtain $\hat{c}$ for sampled configuration.
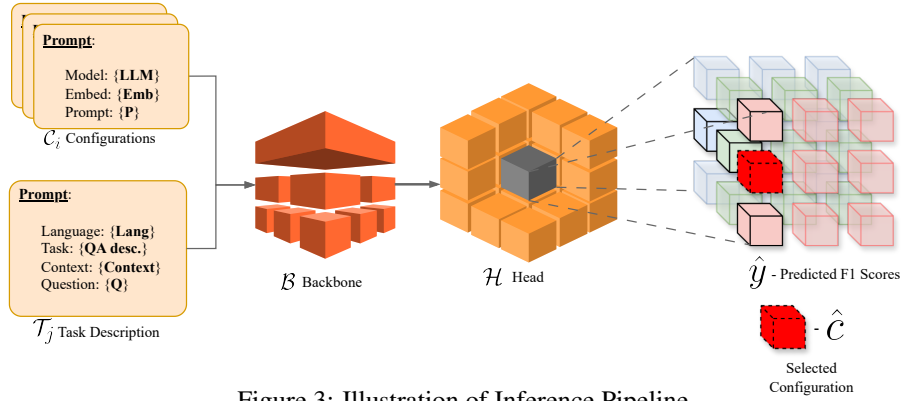
Figure 3: Illustration of Inference Pipeline

Evaluation	Datasets	Acc @top1	Acc @top5	F1 @top1	F1 @top5	Max F1	Random F1	Best single F1
MLQA-F1	IndicQA	0.41	0.83	0.60	**0.64**	0.64	0.46	0.51
	TyDiQA	0.57	0.78	0.52	**0.54**	0.54	0.43	0.50
GPTAnno	IndicQA	0.32	0.48	0.59	0.68	**0.69**	0.49	0.58
	TyDiQA	0.62	0.55	0.56	0.69	**0.72**	0.51	0.54

Table 7: Offline performance.

Evaluation	Datasets	Acc @top1	Acc @top5	F1 @top1	F1 @top5	Max F1	Random F1	Best single F1
MLQA-F1	IndicQA	0.29	0.73	0.60	**0.63**	0.63	0.46	0.51
	TyDiQA	0.62	0.85	0.51	**0.52**	0.52	0.41	0.45
GPTAnno	IndicQA	0.52	0.57	0.62	0.66	**0.69**	0.52	0.61
	TyDiQA	0.62	0.67	0.73	0.74	**0.76**	0.54	0.69

Table 8: Online performance.

5.1 Training the Model for Both Online and Offline Setups.

We train the Backbone $\mathcal{B}$ and the Head $\mathcal{H}$ using different loss functions for offline and online settings.

1. Offline Setting: In the offline setting, we have the advantage of knowing the F1 scores for all possible configurations for each given sample. This complete information allows us to obtain the ground truth F1 scores for all samples, denoted as y. We can then use these ground truth F1 scores to train the backbone $\mathcal{B}$ and the head $\mathcal{H}$ effectively.

(a) **Infer F1 Scores for All Configurations**: For each sample, infer the F1 scores for all possible configurations. For example, if there are three configurations for each parameter (e.g., three language models (GPT-4Turbo, GPT3.5Turbo, Mixtral), four embedding models (adav2,adav3,XLMR,cohere), five prompt strategies (Mono,Trans, Sim,Agg_Src, Agg_Trans)), we would infer F1 scores for $3 \times 4 \times 5 = 60$ configurations per sample.

(b) **Obtain Ground Truth F1 Scores**: Collect the actual F1 scores for all configurations, which serve as the ground truth y. Thus, for each sample, we gather the F1 scores for all 60 configurations.

(c) **Train Using MSE Loss**: Use the Mean Squared Error (MSE) loss to train the model. The MSE loss is computed between the predicted F1 scores $\hat{y}$ and the ground truth F1 scores y $MSELoss = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$, where N is the number of samples.

2. Online Setting: In the online setting, we only have the ground truth F1 score for the configuration that was selected and inferred. This results in a sparse matrix of F1 scores, as we do not compute the F1 scores for all configurations to avoid the computational cost.

(a) **Infer F1 Score for Selected Configuration**: For each sample, infer the F1 score for only the selected configuration. This selected configuration is chosen based on the model's predictions or a sampling strategy. For example, if the model predicts a specific configuration out of 60, we only compute the F1 score for that particular configuration.

(b) **Obtain Ground Truth F1 Score**: Compute the actual F1 score for the selected configuration, which serves as the ground truth $y_{selected}$.

(c) **Update Using Sparse Matrix**: Update the model using the sparse matrix of predicted F1 scores $\hat{y}$. Only the score corresponding to that configuration is updated, leaving other entries unaffected, thus reducing computational overhead.

(d) **Adjust Loss Function**: The loss function must account for the sparsity. Instead of a straightforward MSE loss, we use a modified loss function that updates only the predicted F1 score for the selected configuration, $SparseMSELoss = (\hat{y}_{selected} - y_{selected})^2$, where $\hat{y}_{selected}$ is the predicted F1 score for the selected configuration, and $y_{selected}$ is the ground truth F1 score for the same configuration. This approach optimizes the model

Evaluation	Languages	Acc@top1	Acc@top5	F1@top1	F1@top5	Max- F1	Random-F1	Best Single-F1
Language Adaptation	Kn	0.29	0.75	0.44	**0.46**	0.47	0.37	0.45
	Ta	0.28	0.74	0.48	**0.50**	0.53	0.43	0.46
	Te	0.28	0.74	0.51	**0.55**	0.57	0.43	0.49
Dataset Adaptation	`TyDiQA` on `IndicQA` base	0.56	0.67	0.43	**0.52**	0.52	0.41	0.45

Table 9: Learning approach performance on adaptation to unseen languages and datasets.

without needing to compute F1 scores for all configurations, reducing computational overhead. Implementation details of the above pipeline is explained in Appendix 10.

Train-Test Split. The datasets are divided into three parts: 60% for offline training, 20% for online adaptation, and 20% for testing. Evaluation is performed using the `MLQA-F1` and `GPTAnnotator-F1` metrics.

In our online setup evaluation, we use ground truth due to the difficulty of collecting real-time user feedback, however this setup mirrors an online active learning environment where feedback is gathered on specific samples. As this solution is deployed in QA chatbots or copilots, user feedback would allow it to adapt to new scenarios over time.

5.2 Evaluation of Learning Approach

1. Offline Training Results. We evaluated our model against two baselines: (i) Random Configuration Selection, and (ii) Best Single Configuration (the highest-scoring configuration for all samples). Performance was measured using Accuracy (Acc@Top1, Acc@Top5) and F1 score at top 1 and top 5 configurations.

As shown in Table 7 our model outperforms random selection by 17% and the best single configuration by 11%, consistently across both `MLQA-F1` and `GPTAnnotator-F1` scores. Notably, our approach achieves a top 5 accuracy that matches the maximum achievable accuracy, underscoring its robustness in generating correct answers.

2. Online Training results: We evaluated our model's adaptability to new data distributions by further training it for 10 epochs using parameters from the offline phase (epoch 100). In online adaptation, our model achieved top 1 and top 5 F1 scores of 60% and 63%, respectively, closely approaching the maximum accuracy (63%) (see Table 8). It outperformed random selection by 15% and the best single configuration by 7%, demonstrating effectiveness even with minimal fine-tuning on new or out-of-distribution data.

3. Adaptation Efficacy: *(i) Adaptation to Unseen Languages:* We tested the model's ability to adapt to languages not encountered during offline training. We trained the backbone and the head on the `IndicQA` dataset, excluding Kannada, Tamil, and Telugu languages. The excluded languages were then used for online training, simulating scenarios where the model encounters new languages during inference. Results in Table 9 show the

model generalizing effectively, achieving F1 scores close to the maximum, and outperforming baselines across all languages, proving its adaptability in multilingual scenarios. *(ii) Adaptation to Different Datasets:* We also assessed adaptation to different datasets by training on the `IndicQA` dataset and testing on `TyDiQA`. Despite limited language overlap, our model exceeded random selection by 11% and the best single configuration by 7%. With just 15 fine-tuning epochs on 20% of the `TyDiQA` dataset, it achieved the maximum F1 score, reinforcing its ability to handle diverse datasets and query distributions.

Summary: *Our approach demonstrates substantial improvements in dynamically selecting configurations and adapting to new languages and datasets, showcasing its effectiveness and adaptability in real-world multilingual applications.*

5.3 Comparing with Language specific fine-tuned model

We conducted extensive experiments comparing our dynamic learning approach with state-of-the-art (SOTA) fine-tuned language-specific models. Remarkably, our approach outperforms these fine-tuned models by over 4x in terms of F1 scores.

For example, the Navarasa and Aryabhatta models, fine-tuned on over 10 Indian languages, achieve an average F1 score of just 10% on the Indic dataset across all languages. In contrast, our dynamic approach, as shown in Tables 7 and 8, consistently achieves 60-70% F1 scores. Similarly, we evaluated bi-lingual models like Ambari (fine-tuned for Kannada) and Airavata (fine-tuned for Hindi) on the Indic QA dataset, where their F1 scores were below 5%. This highlights the limitations of fine-tuned models in handling real-world QA tasks.

In contrast, our dynamic approach, without language-specific fine-tuning, achieves F1 scores of over 50-60% across various languages, even when the model was not trained on those specific languages. For instance, as shown in Table 9, our

model achieved an F1 score of 46% on Kannada (KA) without prior training, while Ambari scored less than 2%. This demonstrates the broad applicability and superior performance of our approach across diverse languages, applications, and tasks.

Practical usage: To use our dynamic algorithm, users provide three inputs: (a) base LLM models, (b) multilingual embeddings, and (c) prompt strategies. Our system then dynamically selects the best combination of these for the given language and task, optimizing multilingual performance without manual fine-tuning.

6 Conclusions

In this work, we introduced a dynamic learning framework to enhance multilingual LLM performance without extensive training or fine-tuning. Our findings show that prompting strategies are not universally effective, requiring dynamic, language-specific approaches to optimize performance across datasets, models, and languages. Second, our hybrid use of multilingual embeddings, particularly with Cohere, achieved up to a 7% performance boost on the `GPT-4Turbo` model, highlighting the importance of embedding selection in cross-lingual understanding. Most notably, our dynamic runtime configuration framework demonstrated 10-15% improvements in multilingual task performance and up to 4x gains over language-specific fine-tuned models. Our framework outperformed static, best configurations and baseline models, proving its effectiveness in both offline and online settings. This dynamic adaptability not only enhances LLMs' multilingual capabilities but also future-proofs them, allowing seamless integration with emerging models and strategies. Future research directions include exploration of learning techniques, scalability to larger datasets, and the generalization of our approach to other tasks.

Limitations and Broader Research: While our work takes a first step towards improving multilingual performance, the system is still not fully inclusive, and as a community, we must explore ways to ensure LLMs are accessible to all. Finally, while our key contributions including learning algorithms are generalizable, the optimal strategies and embeddings may differ from one dataset to another. With the growing demand for multilingual language models, our findings pave the way for future advancements in Polyglot LLM performance.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Ashish Sunil Agrawal, Barah Fazili, and Preethi Jyothi. 2024. Translation errors significantly impact low-resource languages in cross-lingual learning. *arXiv preprint arXiv:2402.02080*.

Kabir Ahuja, Rishav Hada, Millicent Ochieng, Prachi Jain, Harshita Diddee, Samuel Maina, Tanuja Ganu, Sameer Segal, Maxamed Axmed, Kalika Bali, et al. 2023a. Mega: Multilingual evaluation of generative ai. *arXiv preprint arXiv:2303.12528*.

Sanchit Ahuja, Divyanshu Aggarwal, Varun Gumma, Ishaan Watts, Ashutosh Sathe, Millicent Ochieng, Rishav Hada, Prachi Jain, Maxamed Axmed, Kalika Bali, et al. 2023b. Megaverse: benchmarking large language models across languages, modalities, models and tasks. *arXiv preprint arXiv:2311.07463*.

Anthropic AI. 2023. Model card for claude 3. Accessed: 2024-05-21.

Sarvam AI. 2024. Announcing openhathi series. Sarvam AI Blog. Accessed: 2024-09-15.

AI4Bharat. 2022. Indicqa: A multilingual question answering dataset for 12 indic languages. https://huggingface.co/datasets/ai4bharat/IndicQA.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*.

Cohere. 2024. Introducing embed v3. Cohere Blog. Accessed: 2024-05-21.

Microsoft corporation. Azure openai service. https://azure.microsoft.com/en-us/products/cognitive-services/openai-service.

Digital Green FarmerChat. 2024. Farmer chat. Website. Accessed: 2024-05-21.

Jinlan Fu, See-Kiong Ng, and Pengfei Liu. 2022. Polyglot prompt: Multilingual multitask promptraining. *arXiv preprint arXiv:2204.14264*.

Jay Gala, Thanmay Jayakumar, Jaavid Aktar Husain, Mohammed Safi Ur Rahman Khan, Diptesh Kanojia, Ratish Puduppully, Mitesh M Khapra, Raj Dabre, Rudra Murthy, Anoop Kunchukuttan, et al. 2024. Airavata: Introducing hindi instruction-tuned llm. *arXiv preprint arXiv:2401.15006*.

Dehong Gao, Kaidi Chen, Ben Chen, Huangyu Dai, Linbo Jin, Wen Jiang, Wei Ning, Shanqing Yu, Qi Xuan, Xiaoyan Cai, et al. 2024. Llms-based machine translation for e-commerce. *Expert Systems with Applications*, 258:125087.

Abdul Ghafoor, Ali Shariq Imran, Sher Muhammad Daudpota, Zenun Kastrati, Rakhi Batra, Mudasir Ahmad Wani, et al. 2021. The impact of translating resource-rich datasets to low-resource languages through multi-lingual text processing. *IEEE Access*, 9:124478–124490.

Naman Goyal, Jingfei Du, Myle Ott, Giri Anantharaman, and Alexis Conneau. 2021. Larger-scale transformers for multilingual masked language modeling. *arXiv preprint arXiv:2105.00572*.

Kaiyu Huang, Fengran Mo, Hongliang Li, You Li, Yuanchi Zhang, Weijian Yi, Yulong Mao, Jinchen Liu, Yuzhuang Xu, Jinan Xu, et al. 2024. A survey on large language models with multilingualism: Recent advances and new frontiers. *arXiv preprint arXiv:2405.10936*.

Lianzhe Huang, Shuming Ma, Dongdong Zhang, Furu Wei, and Houfeng Wang. 2022. Zero-shot cross-lingual transfer of prompt-based tuning with a unified multilingual prompt. *arXiv preprint arXiv:2202.11451*.

HuggingFace. a. Airavata: Introducing hindi instruction-tuned llm. https://huggingface.co/ai4bharat/Airavata.

HuggingFace. b. Ambari: A series of open source bilingual kannada-english large language models. https://huggingface.co/collections/Cognitive-Lab/ambari-65a2678d1051c2b0db3e01fe.

HuggingFace. c. Navarasa 2.0 models: Collection of models navarasa 2.0 models finetuned with gemma on 15 indian languages. https://huggingface.co/collections/Telugu-LLM-Labs/navarasa-20-models-65f7c72addf0619cb0943c0a.

Katharina Hämmerl, Björn Deiseroth, Patrick Schramowski, Jindřich Libovický, Alexander Fraser, and Kristian Kersting. 2022. Do multilingual language models capture differing moral norms? *Preprint*, arXiv:2203.09904.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

KhanAcademy. 2024. Khanmigo. Website. Accessed: 2024-05-21.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, et al. 2021. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*.

Huiyuan Lai and Malvina Nissim. 2024. mcot: Multilingual instruction tuning for reasoning consistency in language models. *arXiv preprint arXiv:2406.02301*.

Patrick S. H. Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2019. MLQA: evaluating cross-lingual extractive question answering. *CoRR*, abs/1910.07475.

Bryan Li, Tamer Alkhouli, Daniele Bonadiman, Nikolaos Pappas, and Saab Mansour. 2024. Eliciting better multilingual structured reasoning from llms through code. *arXiv preprint arXiv:2403.02567*.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2021. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*.

Patrick Littell, David R Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 8–14.

Xiangyan Liu, Rongxue Li, Wei Ji, and Tao Lin. 2023. Towards robust multi-modal reasoning via model selection. *arXiv preprint arXiv:2310.08446*.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. 2024. Understanding llms: A comprehensive overview from training to inference. *Preprint*, arXiv:2401.02038.

M365Copilots. 2023. Introducing microsoft 365 copilot – your copilot for work. Microsoft Blog. Accessed: 2024-05-21.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Conference on Empirical*

Methods in Natural Language Processing (EMNLP), Copenhagen, Denmark.

Cara Marta Messina, Cherice Escobar Jones, and Mya Poe. 2023. Prompting reflection: Using corpus linguistic methods in the local assessment of reflective writing. *Written Communication*, 40(2):620–650.

Xuan-Phi Nguyen, Sharifah Mahani Aljunied, Shafiq Joty, and Lidong Bing. 2024. Democratizing llms for low-resource languages by leveraging their english dominant abilities with linguistically-diverse prompts. *Preprint*, arXiv:2306.11372.

Ercong Nie, Shuzhou Yuan, Bolei Ma, Helmut Schmid, Michael Färber, Frauke Kreuter, and Hinrich Schütze. 2024. Decomposed prompting: Unveiling multilingual linguistic structure knowledge in english-centric large language models. *arXiv preprint arXiv:2402.18397*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. New embedding models and api updates. OpenAI Blog. Accessed: 2024-05-21.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Libo Qin, Qiguang Chen, Yuhang Zhou, Zhi Chen, Yinghui Li, Lizi Liao, Min Li, Wanxiang Che, and Philip S Yu. 2024. Multilingual large language model: A survey of resources, taxonomy and frontiers. *arXiv preprint arXiv:2404.04925*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Sarvam. 2024. Sarvam 2b model. Sarvam 2B Model. Accessed: 2024-09-15.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Microsoft Research Shiksha. 2024. Teachers in india help microsoft research design ai tool for creating great classroom content. Microsoft Research Blog. Accessed: 2024-05-21.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Josué Page Vizcaíno, Federico Saltarin, Yury Belyaev, Ruth Lyck, Tobias Lasser, and Paolo Favaro. 2021. Learning to reconstruct confocal microscopy stacks from single light field images. *IEEE Transactions on Computational Imaging*, 7:775–788.

Siyuan Wang, Jianming Zheng, Wanyu Chen, Fei Cai, and Xueshan Luo. 2023. Multiple: Multilingual prompt learning for relieving semantic confusions in few-shot event detection. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2676–2685.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. *CoRR*, abs/2004.06748.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Wikipedia. List of iso 639 language codes. https://en.wikipedia.org/wiki/List_of_ISO_639_language_codes.

Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. 2024. A survey of resource-efficient llm and multimodal foundation models. *arXiv preprint arXiv:2401.08092*.

Hao Yang, Junyang Lin, An Yang, Peng Wang, Chang Zhou, and Hongxia Yang. 2022. Prompt tuning for generative multimodal pretrained models. *arXiv preprint arXiv:2208.02532*.

Mengjie Zhao and Hinrich Schütze. 2021. Discrete and soft prompting for multilingual models. *arXiv preprint arXiv:2109.03630*.

Orr Zohar, Shih-Cheng Huang, Kuan-Chieh Wang, and Serena Yeung. 2024. Lovm: Language-only vision model selection. *Advances in Neural Information Processing Systems*, 36.

Ahmet Üstün, Viraat Aryabumi, Zheng-Xin Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. 2024. Aya model: An instruction finetuned open-access multilingual language model. *Preprint*, arXiv:2402.07827.

Appendix

7 Similar Language Algorithm

Section 3 introduced various prompt strategies and prompt templates that we have optimized for polyglot LLMs. One of the prompt strategies defined is round-tripping the input in source language through "Similar high-resourced language (Sim)". In this section, we present the algorithm for identifying the right set of similar high-resourced languages for a given source language. For every language, we associate its class attribute between 0-5 based on the classes defined in (Joshi et al., 2020). Here, class 5 represents very high-resourced languages like English, whereas 0 represents very low-resourced languages like Gondi, Mundari, etc. We use the language similarity metrics based on language feature similarities(Malaviya et al., 2017) captured in lang2vec (Littell et al., 2017). We give higher preference to the languages with Latin script since the languages with Latin script have shown better performance on GPTx models(Ahuja et al., 2023a).

Algorithm 1: Get language relevance score based on language similarity distance, the class of related language and whether the related language has Latin script.

$w_{Latin} \leftarrow 0.9$;
Function GetRelevanceScore(d, l_{cls}, $isLatin$):
  $w \leftarrow 1$;
  if $isLatin$ **then**
    $\lfloor\ w \leftarrow w_{Latin}$
  $score \leftarrow w \times d/l_{cls}$;
  return $score$

---

8 Prompt Strategies Results

Performance of prompts for `TyDiQA`.

In this section we present the performance of our Prompts on `TyDiQA` dataset, We report `MLQA-F1` and `GPTAnnotator-F1` for each prompt Averaged across all 9 languages. The numbers are reported for `GPT-4Turbo` and `GPT3.5Turbo`

	MLQA-F1		GPTAnnotator-F1	
	GPT-4Turbo	GPT3.5Turbo	GPT-4Turbo	GPT3.5Turbo
Mono	**0.64**	**0.64**	**0.71**	**0.71**
Tans	0.49	0.51	0.61	0.63
simi	0.47	0.47	0.58	0.58
Aggsrc	0.62	0.63	0.69	0.70
aggtrans	0.49	0.52	0.60	0.63

Table 10: Performance of different Prompt strategies for `TyDiQA`

with text-embedding-ada-002 embeddings In Table.10 we observe similar trends to experiment with `IndicQA`, i.e., Each model has different trends across different prompting strategies and the choice of the metrics also favours different model making it difficult to find a suitable choice of prompt for a generalized pipeline.

Algorithm 2: Identifying similar high-resourced languages for a given language

---

Data: Source language l_s
Result: A set of similar high-resourced languages $L_{similar}$
$L_{similar} \leftarrow \emptyset$
Language class threshold $cls_{threshold} \leftarrow 3$
Languages similarity distance threshold $dist_{threshold} \leftarrow 0.5$
for $l \in L$ **do**
  if $class(l) \geq cls_{threshold}$ **then**
    Similarity distance between l and l_S:
    $d \leftarrow lang2vec\_distance$
    (['syntactic','genetic',
    'geographic'],l,l$_s$);
    $RelevanceScore \leftarrow$
    GetRelevanceScore(average(d),
    class(l), isLatin(l);
    if $RelevanceScore \leq dist_{threshold}$
    then
      $\lfloor\ L_{similar}.add(l)$

Per language performance for `GPT-4Turbo` and `GPT3.5Turbo` for `IndicQA`

Table. 11, 12 presents the performance of `GPT-4Turbo` and `GPT3.5Turbo` respectively with text-embedding-ada-002 embeddings, across all 11 languages and 5 prompts that we propose. Here we observe strong patterns for Agg_Sim performing the best across majority of the languages ($\frac{7}{11}$ for `GPT-4Turbo` and $\frac{5}{11}$ for `GPT3.5Turbo`), Mono performs better and comes very close to Agg_sim in these languages. For languages such as "ta", "te" translate is prefered. With the limited

9220

Table 11: GPT-4Turbo on IndicQA

Lang	Mono	Translate	Similar	AggSim	AggTrans
as	**0.58**	0.33	0.33	**0.58**	0.32
bn	**0.62**	0.38	0.36	**0.62**	0.36
gu	**0.59**	0.31	0.30	**0.59**	0.30
hi	**0.67**	0.54	0.42	**0.68**	0.51
kn	**0.48**	0.31	0.25	**0.48**	0.29
ml	**0.32**	0.30	0.19	**0.32**	0.29
mr	**0.58**	0.33	0.30	**0.57**	0.32
or	**0.57**	0.29	0.27	**0.57**	0.27
pa	**0.61**	0.46	0.43	0.60	0.46
ta	0.31	**0.40**	null	0.34	0.39
te	0.25	0.36	0.17	0.28	**0.37**
AVG	**0.51**	0.36	0.30	**0.51**	0.35

Table 12: GPT3.5Turbo on IndicQA

Lang	Mono	Translate	Similar	AggSim	AggTrans
as	**0.40**	0.34	0.33	**0.45**	0.37
bn	**0.54**	0.39	0.34	**0.54**	0.46
gu	**0.48**	0.32	0.30	**0.49**	0.33
hi	**0.63**	0.52	0.38	**0.64**	0.52
kn	**0.47**	0.32	0.21	**0.46**	0.31
ml	0.23	**0.32**	0.13	**0.26**	0.31
mr	**0.48**	0.34	0.30	**0.47**	0.36
or	**0.40**	0.29	0.27	**0.39**	0.32
pa	**0.54**	0.46	0.40	**0.54**	0.44
ta	0.31	**0.40**	null	0.24	**0.39**
te	0.25	**0.36**	0.17	0.25	**0.34**
AVG	**0.43**	0.37	0.28	**0.43**	0.38

Table 13: GPT-4Turbo on TyDiQA

Lang	Mono	Translate	Similar	AggSim	AggTrans
ar	**0.50**	0.43	null	**0.50**	0.40
bn	**0.69**	0.46	0.43	**0.69**	0.43
en	**0.65**	null	0.60	0.62	0.58
fi	**0.63**	0.49	0.48	0.59	0.49
id	**0.66**	0.58	0.53	0.63	0.54
ko	**0.64**	0.48	0.43	0.63	0.47
ru	**0.51**	0.45	0.46	0.50	0.44
sw	**0.80**	0.63	null	0.78	0.65
te	**0.67**	0.42	0.39	0.66	0.43
AVG	**0.64**	0.49	0.47	0.62	0.49

Table 14: GPT3.5Turbo on TyDiQA

Lang	Mono	Translate	Similar	AggSim	AggTrans
ar	0.53	0.45	null	0.52	0.42
bn	0.65	0.46	0.41	0.65	0.48
en	0.66	null	0.61	0.65	0.64
fi	0.68	0.53	0.50	0.65	0.54
id	0.67	0.61	0.53	0.66	0.59
ko	0.65	0.49	0.46	0.66	0.51
ru	0.52	0.46	0.45	0.51	0.44
sw	0.76	0.64	null	0.74	0.64
te	0.66	0.44	0.36	0.67	0.43
AVG	**0.64**	0.51	0.47	0.63	0.52

languages the variance in the trend is high and a rule based system would fail with inclusion of more languages.

Per language performance for GPT-4Turbo and GPT3.5Turbo for TyDiQA In Table. 13, 14 performance of GPT-4Turbo and GPT3.5Turbo along with text-embedding-ada-002 embeddings are presented across all 9 languages and 5 proposed prompts. Here contrary to IndicQA experiments Mono is preferred over Agg_sim making a significant change in distribution. The optimal prompt doesn't depend only on the language or model but also on the distribution of the question, this statement is supported by the fact TyDiQA and IndicQA share 2 languages "bn" and "te", while in IndicQA Agg_sim was prefered for "bn" and Translate for "te" it has completely shifted to Mono for both "bn" and "te" in TyDiQA. Hence prompt selection is depends on the language and also the distribution of the dataset or sample.

9 Hybrid approach

In this section we evaluate the performance of our Hybrid Approach across text-ada-002-embedding, Adav3, XLMRXXL and Cohere

embed_multilingual_v3. We use TyDiQA as the dataset and average the MLQA-F1 and GPTAnnotator-F1 across all 9 languages and all 5 prompts. In Table. 15 we present the values for both GPT-4Turbo and GPT3.5Turbo, while the trend is completely different to that of IndicQA which could be primarily attributed to the languages typology and derivations.

10 Detailed Training Procedure & Implemenation Details

The algorithm employs separate strategies for inference and training tailored to different operational conditions. During inference, the algorithm selects the optimal configuration based on F1 score predictions from task and configuration embeddings as described in Algorithm 3. In the Offline Setting, configuration selection is deterministic, using an argmax function for precise, data-rich environments. Conversely, the Online Setting uses a probabilistic softmax function to adapt to data-scarce situations, enabling dynamic exploration and refinement of configurations.

For training, the offline mode applies a Mean Squared Error (MSE) loss across all configurations, ensuring comprehensive learning. In contrast, the online mode implements a sparse MSE loss, updating only the evaluated configurations through a masking technique. This approach reduces compu-

Metrics	Models	Ada	Adav3	XLMRXXL	Cohere
MLQA-F1	GPT-4Turbo	**0.64**	**0.64**	0.60	**0.61**
	GPT3.5Turbo	**0.64**	0.60	0.57	**0.59**
GPTAnnotator-F1	GPT-4Turbo	**0.71**	**0.71**	0.65	**0.68**
	GPT3.5Turbo	**0.71**	0.66	0.63	**0.66**

Table 15: Hybrid approach performance - `TyDiQA`.

tational load and accelerates adaptation to new data, optimizing performance in real-time applications as outlined in Algorithm 3.

The sparse MSE Loss employs a Mask $\mathcal{M}$ which is defined as, Let $\mathbf{C}$ be a tensor of order m with dimensions $n_1 \times n_2 \times \cdots \times n_m$. Suppose $\hat{c} = C_{i_1,i_2,...,i_m}$ is a selected element from $\mathbf{C}$, where $(i_1, i_2, \ldots, i_m)$ are the indices of $\hat{c}$ in $\mathbf{C}$. Define the tensor $\mathcal{M}$ as follows:

$$\mathcal{M}_{j_1,j_2,...,j_m} = \begin{cases} 1 & \text{if } (j_1, j_2, \ldots, j_m) = (i_1, i_2, \ldots, i_m) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

10.1 Implementation Details

In this work, we use Azure OpenAI models (corporation) for all our LLM and embedding models including `GPT-4Turbo`, `GPT3.5Turbo` and `Mixtral`. For the given configurations of base LLM models, embeddings and prompt strategies, the training needs to be performed only once and can be shared with different multilingual applications and use-cases. For the learning model, we train the Llama model on GPU with A100 80 GB, CPU with 96 cpu cores at 2.2GHz and 1024 GB RAM. The duration of training 100 offline epochs is 1.42 Hrs. The duration of training 25 online epochs is 0.74 Hrs. The inference and evaluation is dependent on the rate limits imposed by Azure OpenAI APIs (corporation). **Model Version**: For LLMs we use `GPT-4Turbo`- 0125-preview, `GPT3.5Turbo`- 0125, Mixtral - Mixtral-8x7B-Instruct-v0.1; For Embeddings we use ada - text-ada-002-embedding, ada3 - text-ada-003-embedding, XLMR-XXL - facebook/xlm-roberta-xxl and Cohere - embed_multilingual_v3;.

11 `GPTAnnotator` Setup and details

11.1 Human Annotation Task Details

We build a simple human annotation interface using Streamlit[1] where the context, the question

[1]https://streamlit.io/

Algorithm 3: Learning Strategy Algorithm for Inference and Training

Data: Task descriptions $\mathcal{T}$, configuration options $\mathcal{C}_i$

Result: Optimal configuration $\hat{c}$ and its corresponding F1 score

$\mathcal{B}$ - LLaMa-2-70B backbone for embedding generation

$\mathcal{H}$ - Conv-ND layers for F1 score prediction

e - embedding projection size, $e = 8192$

m - number of parameters, $m = 3$ (e.g., language model, embedding model, prompt strategies)

$\mathcal{R}^{n_1 \times n_2 \times \cdots \times n_m}$ - size of the N-dimensional array for configurations

bs - Batch size of Task Definitions.

for $\mathcal{T}_j \leftarrow \{\mathcal{T}_0, ...\mathcal{T}_{bs}\}$ **do**

$\quad \mathcal{E}_{Tj} \leftarrow \mathcal{B}(\mathcal{T}_j) \,; \mathcal{E}_{Ci} \leftarrow \mathcal{B}(\mathcal{C}_i)$

$\quad \mathcal{E}_j \leftarrow \text{Concatenate}(\mathcal{E}_{Tj}, \mathcal{E}_{Ci})$

$\quad \hat{y} \leftarrow \mathcal{H}(\mathcal{E}_j)$

$\quad$ /* Inference for selecting configuration */

$\quad$ **if** *Offline Setting* **then**

$\quad\quad \hat{c} \leftarrow \arg\max(\hat{y})$

$\quad$ **else if** *Online Setting* **then**

$\quad\quad \hat{c} \sim \text{Softmax}(\hat{y})$

$\quad$ /* Training to update $\mathcal{H}$ & $\mathcal{B}$ */

$\quad$ **if** *Offline Setting* **then**

$\quad\quad y \leftarrow$ Ground truth F1 scores $\forall \mathcal{C}_i$

$\quad\quad Loss_{\text{off}} \leftarrow \text{MSE}(\hat{y}, y)$

$\quad$ **else if** *Online Setting* **then**

$\quad\quad y_{\text{sparse}} \leftarrow$ Ground truth F1 score for $\hat{c}$

$\quad\quad \mathcal{M} \leftarrow$ Mask matrix using eq. 5

$\quad\quad Loss_{\text{on}} \leftarrow \text{MSE}(\mathcal{M} \odot \hat{y}, y_{\text{sparse}})$

$\quad$ Update $\mathcal{H}$ & $\mathcal{B}$ using $Loss$

related to the context, and the ground truth answer for each record are fetched from the IndicQA dataset(AI4Bharat, 2022). In this evaluation task, the annotators are first presented with a passage that acts as the context required to answer the question which is shown along with the ground truth answer. The annotators are then asked to evaluate the answers generated by the LLM using different strategies based on the ground truth answer provided, by answering one of the following options: "Yes", "No" or "Partial". Here is the instruction provided to the annotators.

> First, select your language and go through the context under the title "Context GT" once. Then, look at the question and try to answer this question and compare it with the ground truth answer. Next, for all the available answers, choose:
>   1. "Yes" if the answer is absolutely correct(minor punctuation errors are allowed)
>   2. "Partial" if the answer captures some part of the core answer, but has grammatical mistakes or minor errors(spelling, etc.) that make the answer partially correct.
>   3. "No" if the answer is completely wrong

Based on the human annotations for each question, we then recompute the F1 score. The updated F1 scores are calculated using Algorithm 4, where *evals* contains evaluations for all the strategies annotated by the human annotator.

Algorithm 4: Evaluation Algorithm when using Human Annotator or GPTAnnotator

Data: $ground\_truth$, $gpt\_answers$, $evals$
Result: $eval\_scores$
$eval\_scores \leftarrow []$; $valid\_answers \leftarrow []$;
$evals = get\_eval(gpt\_answers)$;
$valid\_answers.append(ground\_truth)$;
for $i \leftarrow 0$ **to** $len(gpt\_answers)$ **do**
 if $evals[i] = "Yes"$ **then**
  $valid\_answers.append(gpt\_answers[i])$;

for $i \leftarrow 0$ **to** $len(gpt\_answers)$ **do**
 $eval\_f1.append(compute\_score$
 $(gpt\_answers[i], valid\_answers))$;

11.2 GPT Eval process

In Section 2.3, we introduced GPTAnnotator, where GPT models perform the evaluation of the answer generated when compared to the ground truth. Similar to the human evaluation task described in the previous subsection 11.1, the GPTAnnotator is tasked to evaluate the LLM responses based on the available ground truth for the given record. The prompt below is used for GPT3.5Turbo in order to evaluate the answers.

> You are a multilingual evaluation assistant. Users will send in a query, context text, the correct answer for the query based on the context text, and also an answer that needs to be evaluated. You will evaluate the answer based on the context text and the correct answer that the user has sent and respond with Yes, No, or Partial based on the below evaluation instructions. Instructions: 1. Yes if the answer is absolutely correct. 2. Partial if the answer captures some part of the correct answer, but has minor errors like grammatical or spelling mistakes, etc. 3.No if the answer is completely wrong.

The updated F1 Scores for each of the strategy is calculated using Algorithm 4 where *evals* contains "Yes", "No" or "Partial" evaluations as judged by the GPTAnnotator.