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

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

Abstract

Large language models (LLMs) are at the forefront of transforming numerous domains globally. However, their inclusivity and effectiveness remain limited for non-Latin scripts and low-resource languages. This paper tackles the imperative challenge of enhancing the multilingual performance of LLMs without extensive training or fine-tuning. Through systematic investigation and evaluation of diverse languages using popular question-answering (QA) datasets, we present novel techniques that unlock the true potential of LLMs in a polyglot landscape. Our approach encompasses three key strategies that yield significant improvements in multilingual proficiency. First, by meticulously optimizing prompts tailored for polyglot LLMs, we unlock their latent capabilities, resulting in substantial performance boosts across languages. Second, we introduce a new hybrid approach that synergizes LLM Retrieval Augmented Generation (RAG) with multilingual embeddings and achieves improved multilingual task performance. Finally, we introduce a novel learning approach that dynamically selects the optimal prompt strategy, LLM model, and embedding model per query at run-time. This dynamic adaptation maximizes the efficacy of LLMs across languages, outperforming best static and random strategies. Additionally, our approach adapts configurations in both offline and online settings, and can seamlessly adapt to new languages and datasets, leading to substantial advancements in multilingual understanding and generation across diverse languages.

1 Introduction

Large Language Models (LLMs) such as ChatGPT [35], Gemini [45], and Claude [5] have revolutionized AI research, demonstrating vast advancements across various tasks [8, 37, 35]. These models serve as intelligent assistants in enterprise applications (e.g., search engines, office suites), and several practical domains like healthcare, education, and agriculture [40, 20, 2, 33]. They are transforming teaching methods, agricultural practices, and many professional interactions with AI systems.

However, the current landscape primarily favors LLMs optimized for English and Latin script languages, limiting their effectiveness in non-English contexts [3, 4, 25]. Despite recent advancements like fine-tuned LLMs [17] and smaller language models [1], their performance remains subpar, especially in multilingual scenarios. Numerous studies highlight a noticeable performance gap between LLMs (including proprietary models like GPTx and Gemini, fine-tuned variants, and smaller models like OpenHathi [6]) and state-of-the-art

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`.

*Equal Contributions

Preprint. Under review.

(SOTA) multilingual models such as TULRv6 and XLMR [19].
Table 1 compares the performance of various LLMs, including
LLama2 70B, Mistral 7B instruct, Cohere, Palm2, GPT3.5, and GPT4, against SOTA models like TULR-XXL on the multilingual QA dataset `TyDiQA` (covering 9 diverse languages) [10]. Although GPT4 and Palm2 have improved over GPT3.5, a significant gap remains to SOTA models. This pattern persists across other multilingual QA benchmarks, such as MLQA, IndicQA, and AfriQA[34].

To bridge this gap, two main approaches are being explored. The first involves enhancing foundational model training, which faces several challenges: **1) Scarcity of Training Data:** Limited high-quality data for non-English and low-resource languages hinders optimal multilingual LLM pre-training [23, 48]. **2) Lack of Access and Resources:** Many existing models are not open-source, and the high computational costs of training/fine-tuning restrict customization for specific languages [38, 31]. **3) Limited Adaptability:** Fine-tuned or smaller models often struggle to understand languages outside the targeted subset, despite performance improvements for specific languages [44].

The alternative approach enhances the performance of pre-trained LLMs through external configurations: **1) Prompt Tuning:** Research explores strategies like native language, cross-lingual, and Chain-of-Thought prompting, improving performance for specific languages/tasks [49, 43]. However, no single strategy consistently outperforms others across all tasks and languages [29]. **2) Optimizing Model Embeddings:** Tasks like question answering benefit from Retrieval Augmented Generation (RAG), incorporating external knowledge [18]. Improved text-embedding models enhance performance by retrieving relevant information. For example, OpenAI's text-embedding-3 improves multilingual performance over its predecessor, text-embedding-ada-002 (MIRACL score from 31.4% to 54.9%[36]), and Cohere's embed-multilingual-v3.0 achieves a MIRACL of 67%[11]. Choosing the right embedding model remains challenging. **3) Model Selection Dilemma:** New LLM versions, proprietary (e.g., OpenAI GPTx, Gemini, Claude) and open-source (e.g., Llama, OpenHaithi), are continuously released. Identifying the best model for specific tasks and languages remains uncertain.

The lack of a definitive configuration (prompt strategy, embeddings, LLM model) drives our work, aiming to enhance multilingual LLMs performance without training or fine-tuning. We introduce three techniques to improve multilingual LLM performance: (i) optimizing prompting strategies, (ii) integrating multilingual embeddings with LLMs for better document retrieval and generation, and (iii) dynamically selecting prompts, language models, and embeddings at runtime to optimize performance across languages and tasks. Our key contributions are:

1. **Optimizing Prompts for Polyglot LLMs:** Crafting prompts tailored to the unique characteristics of LLMs results in significant performance enhancements across languages.
2. **Hybrid Approach combining LLM Generation with Multilingual Embeddings:** Combining LLM response generation with *multilingual embeddings* in a Retrieval Augmented Generation (RAG) setting enhances the coherence and context relevance in text retrieval and generation, thereby enhancing multilingual task performance.
3. **Dynamic Learning Approach for Performance Optimization:** Our approach dynamically selects the best prompt strategy, LLM model, and multilingual embedding model at runtime, maximizing efficacy across languages and surpassing best static and random strategies.

We assessed the effectiveness of our techniques on two widely used Question Answering (QA) datasets: `IndicQA` and `TyDiQA`, covering 18 languages. Our findings reveal significant limitations in current datasets and evaluation approach, highlighting the need for improved task evaluation (see Section 2.3). Through extensive experimentation, **our results demonstrate over 15-20% improvement in multilingual performance across diverse languages**. These results underscore the effectiveness of our approach, which can be applied to various prompting techniques, tasks, and models described in existing literature. We also provide the source code for the community[2].

2 Multilingual Tasks, Datasets & their Limitations

In this work, we prioritize Question Answering (QA) tasks, showcasing the model's ability to deliver accurate responses. Here, we focus on the RAG based QA tasks that leverage the text information provided as external knowledge-base for answering the question.

2.1 Dataset

We utilize two prominent multilingual QA datasets:

[2]The source code will be released soon.

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: इस्रोने **Translate_En_Answer:** ISRO
Generated answers & F1 score: इसरो ने (0), इस्रो (0), भारतीय अंतराळ संशोधन संस्था (इस्रो) (0), भारतीय अंतराळ संशोधन संस्थेच्या (0), इस्रोने केले (0.67)

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

1.IndicQA [7]: 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 [10]: This dataset covers 9 typologically diverse languages and includes two sub-tasks: passage selection and minimum answer span (Gold-P). 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 Task

F1 score is the commonly used metric in QA tasks [39], compares individual words in predictions to the True Answer. While SQuAD-F1 is standard for English QA evaluation, MLQA-F1 [27] 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 public datasets for evaluating multilingual performance were developed before the Large Language Model (LLM) era. Evaluating LLMs on these datasets poses two main challenges:

Challenge 1: Limited Ground Truth (GT). GT in these datasets typically includes only one answer per question, whereas in reality, there could be multiple semantically equivalent variants more suitable in conversational or real-world application settings.
Challenge 2: Strict Evaluation Approach. The F1 score, often computed at the individual word level, poses a challenge especially in cases where there is only one GT. Even minor differences between ground truth and predicted answers lead to notable score reductions.

Figure 1 illustrates the above challenges on the IndicQA dataset for Kannada (kn) and Marathi (mr) languages. Each question is accompanied by the GT answer and generated results from different strategies. In the left example for Kannada, although the generated answers differ, they remain factually accurate, providing an alternative reference to "British". Similarly, in the right, the answer is an abbreviation for an organization, while the generated answers include a mix of abbreviated and fully expanded versions. As the dataset includes only one GT answer, the MLQA-F1 scores for these cases would be exceptionally low, even zero. These examples highlight the limitations of current datasets and evaluation approach.

One potential solution is to enrich the ground truth with all possible alternatives. However, this would require significant data collection effort, proving both expensive and cumbersome. Recently, LLMs have proven to be proficient annotators for various tasks across domains [21]. Verifying or validating answers is simpler than generating them [26]. To address the GT limitation, we propose GPTAnnotator, utilizing an LLM to validate a set of predicted answers. GPTAnnotator employs an LLM like GPT4, where for each question, it assesses numerous predicted answers generated using different strategies (prompts and models) for their correctness (see Appendix 11 for prompts).

GPTAnnotator offers three options: *YES* for any semantically correct answer that matches with the GT answer, *NO* for no match, and *PARTIAL* for partial match. When GPTAnnotator yields a *YES*, predicted answers are added to the original GT, enhancing it with multiple equivalent answers. For instance in Figure 1, all generated answers would now be marked as correct and added to GT. This addresses challenge 1 by ensuring a more comprehensive evaluation. To tackle challenge 2, we introduce GPTAnnotator-F1 score, computing F1 against this new comprehensive ground truth with multiple semantically correct answers. Thus, we evaluate LLMs' multilingual performance for QA tasks using two metrics: MLQA-F1, comparing the predicted answer to the original GT, and GPTAnnotator-F1, evaluating against a derived, comprehensive GT. Note both are F1 scores but evaluated with single answer GT (MLQA-F1) and multiple answers GT (GPTAnnotator-F1).

Our goal is to ensure that the GPTAnnotator-F1 score closely mirrors a human annotator's score. To compare GPTAnnotator-F1 and MLQA-F1, we randomly selected 100 questions from the IndicQA

dataset across six languages (bn, hi, ta, kn, mr, gu) and presented them to native speakers in our research group. Each annotator received the context paragraph, question, GT answer, and a set of predicted answers generated using various strategies, akin to `GPTAnnotator`. Annotators selected *YES* for a complete match (Score = 1), *NO* for no match (Score = 0), and *PARTIAL* for a partial match (Score = 0.5 or F1 against GT), termed `HumanAnnotator-Score`.

Figure 2 illustrates the difference between `HumanAnnotator-Score` and `MLQA-F1`, and `HumanAnnotator-Score` and `GPTAnnotator-F1` for the six languages across all strategies. On these 100 samples (across 6 languages), `MLQA-F1` scores showed an average difference of 25% and a maximum difference of 51% compared with `HumanAnnotator-Score`, indicating significant gap. This discrepancy arises due to the limited original GT provided in the datasets and the generative models' ability to produce responses with variations. In contrast, `GPTAnnotator-F1` differences are substantially lower compared to `MLQA-F1` differences when compared to `HumanAnnotator-Score`. Additionally, across all languages, `GPTAnnotator-F1` scores reduce the difference by 30% compared to `MLQA-F1` scores and align more closely with `HumanAnnotator-Score`. Thus, with the assistance of `GPTAnnotator`, we can address current limitations in the dataset and evaluation approach, and importantly, evaluate performance more akin to human assessment. In the subsequent sections, we present results obtained using both `MLQA-F1` and `GPTAnnotator-F1` metrics.

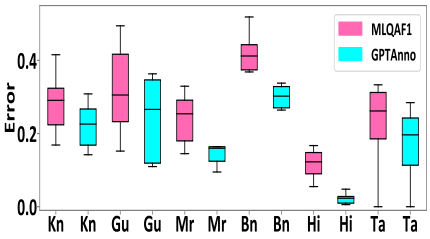

Figure 2: Comparison of `MLQA-F1` and `GPTAnnotator-F1`.

3 Prompt Strategies for Polyglot LLMs

Generative models' performance heavily relies on prompt tuning [41]. Crafting effective prompts, even for specific English tasks, is challenging [50]. Although tools like PromptBreeder [16] and MedPrompt [9] aid prompt creation for generic English QA tasks, defining prompts for multilingual scenarios lacks clear strategies [3, 4]. Through extensive experiments and analysis, we've identified LLM strengths and designed optimized templates for polyglot LLMs.

Our notation is as follows: Each prompt comprises an Instruction and Context, with contexts being passages like in reading comprehension for answering questions in QA scenarios. We employ two approaches - *zero-shot*, where no examples are passed, and *few-shot*, where a few random examples are passed based on the strategy. The language in which the query is issued is termed the source language, which is also the language for the final response. We devise the following prompting strategies to enhance the model's performance in multilingual tasks:

1. Monolingual (Mono): In this technique, the Instruction and Context are in the source language. Few-shot examples are also from the source language.

2. Translate-Test (Trans): This approach translates the Instruction and Context into English using an automatic Machine Translation system [15]. The model is then queried in English, and the result is back-translated into the source language (Roundtripping through English).

3. Similar high-resourced language (Sim): Similar to translate-test, this method involves roundtripping through another pivot language, typically in the high or medium-resource category. The pivot language is selected based on its proximity to the source language in terms of language feature similarities [32] captured in lang2vec [28]. The rationale is that selecting a pivot language based on language similarities allows for better capture of linguistic aspects compared to direct translation to English. Initially, multiple similar languages are identified using various feature similarity metrics, from which the pivot language is chosen. Preference is given to high-resource languages with Latin script, as evidence suggests they perform better [29]. This selection process ensures the pivot language, similar to the source language, enhances the strategy's effectiveness. More details in Appendix 7.

4. Aggregation Source (Agg_Src): This strategy aggregates responses from previous strategies such as mono, trans, and sim. These responses are presented to the LLM to determine the best answer in the source language based on input from other strategies. While this increases the number of calls to the LLM, it effectively combines the strengths of different prompting strategies and leverages information from multiple languages to generate a single response.

	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 2: Performance of different Prompting strategies for IndicQA. Bold indicates the strategies that give best performance for that model.

5. Aggregation Translate (Agg_Trans) Similar to Agg_Src, this strategy collects responses from each of the other strategies, such as mono, trans, and sim. These responses are translated into English. The three English responses are then aggregated using the LLM to generate a single aggregated response in English. This aggregated response in English is subsequently translated back into the source language to obtain the final response.

While we emphasize the above five primary multilingual prompting strategies, we also experimented with other popular approaches like self-translation and Chain-Of-Thought, which showed inconsistent performance. All strategies are executed with both zero-shot and few-shot (in-context examples). We use randomly selected fixed few-shot examples for consistency, aiming to demonstrate performance improvement. Advancements in in-context learning and example selection will further enhance all strategies and are complementary.

Prompting Strategies Results. We found that few-shot examples consistently improve performance across all models compared to zero-shot. Therefore, we present only few-shot results going forward. Table 2 shows average performance of the individual prompting strategies across all languages in the IndicQA dataset across GPT-4Turbo [14], GPT3.5Turbo [13], and Mixtral [12], considering both MLQA-F1 and GPTAnnotator-F1 metrics with text-embedding-ada-002 default embeddings. Three main takeaways emerge:

1. No Universal Best Strategy: There is no single prompt strategy that universally performs best across all languages for each model. For GPT3.5Turbo and GPT-4Turbo, Mono and Agg_Src strategies generally excel across most languages, while Mixtral tends to favor Trans and Agg_Trans. Notably, languages like 'ta' and 'te' consistently prefer translate strategies due to low-resource language constraints. Thus, the optimal strategy varies for each model and language. Appendix 8 provides more details regarding per-language performance across prompt strategies.

2. Strategy Sensitivity to Metrics: The choice of prompt strategies changes when different metrics are considered. When GPTAnnotator-F1 scores are utilized for GPT-4Turbo and GPT3.5Turbo, the best strategies for all languages include Trans and Agg_Trans, whereas Mono and Agg_Src perform better when MLQA-F1 is used. Similar takeaways arise for the TyDiQA dataset see Appendix 8.

Summary: *Prompt strategies enhance multilingual performance, but there's no one-size-fits-all solution across datasets, metrics, models, and languages. Notably, with GPTAnnotator-F1 scores, GPT3.5Turbo shows comparable performance to GPT-4Turbo.*

4 Hybrid Approach: Synthesizing LLM Generation with Multilingual Embeddings

Currently, most LLMs are trained predominantly on English and a few high-resource languages, limiting their understanding of medium and low-resource languages [22]. Tasks like question answering and summarization often require LLMs to incorporate non-parametric knowledge from private knowledge bases or contextual information. Retrieval Augmented Generation (RAG) addresses these limitations by retrieving relevant documents or information chunks for a given query and then synthesizing responses [18]. The process involves: 1) encoding knowledge-base documents using a text embedding model and indexing them for retrieval speed, 2) encoding the query using the same model, 3) conducting a similarity search between the query embedding and the indexed document embeddings to retrieve top k similar documents, and 4) forwarding the retrieved documents to the LLM for response synthesis. Better text-embedding models enhance task performance, especially in multilingual scenarios, by fetching the most relevant information based on the query. This improves response generation, even in languages with limited training data.

While LLMs excel in response synthesis, enhancing multilingual task performance relies on a robust multilingual text-embedding model. Since GPT models are primarily trained on English data, their

Metrics	Models	Ada	Adav3	XLMR	Cohere
MLQA-F1	GPT-4Turbo	0.51	0.5	0.54	**0.58**
	GPT3.5Turbo	0.43	0.43	0.39	**0.44**
GPTAnno	GPT-4Turbo	0.8	0.8	0.8	**0.82**
	GPT3.5Turbo	0.8	0.8	0.8	**0.81**

Table 3: Hybrid approach performance on IndicQA.

	MLQA-F1		GPTAnnotator-F1	
Lang	GPT3.5Turbo	GPT-4Turbo	GPT3.5Turbo	GPT-4Turbo
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 4: Embedding preference IndicQA.

	MLQA-F1		GPTAnnotator-F1	
Lang	GPT3.5Turbo	GPT-4Turbo	GPT3.5Turbo	GPT-4Turbo
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 5: Embedding preference TyDiQA.

default embedding model (text-embedding-ada-002, or ada) leads to suboptimal performance. In contrast, state-of-the-art multilingual models like XLMR-XXL [19], and Cohere (embed-multilingual-v3.0) [11] demonstrate superior performance due to their training on diverse language data.

In this paper, we introduce a **hybrid approach** that integrates the cross-lingual semantic understanding of multilingual embeddings with the text generation capabilities of LLMs. We experiment with GPT's default embedding (ada), its improved variant, text-embedding-3-large (adav3) [36], which offers enhanced multilingual performance, and state-of-the-art multilingual embeddings such as XLMR-XXL [19] and Cohere multilingual embeddings v3 (Cohere) [11].

Performance Analysis. Table 3 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 default 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 4 and 5 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 incorporating multilingual embeddings enhances task 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 Learning Approach to Improve Multilingual Task Performance

As demonstrated in previous sections, it is evident that a one-size-fits-all approach does not exist when it comes to selecting the optimal configuration of prompt strategy, embeddings, and LLM model for diverse multilingual languages. This leads us to the crucial question: *Can we dynamically determine the most optimal configuration for each query to maximize multilingual task performance?*

To tackle this challenge, we propose a learning approach that identifies the most optimal configuration for a given query. This approach should fulfill several key requirements: (i) **Offline Learning:** It should be capable of learning the optimal configuration using ground truth data in an offline setup, (ii) **Online Learning:** It should also be able to adapt and learn in real-time, allowing the base model to fine-tune itself to accommodate new data and distribution shifts between training and testing, and (iii) **Language and Dataset Adaptability:** Additionally, it should be flexible enough to adapt to different languages and datasets in an online setup, ensuring robust performance across diverse linguistic contexts and data sources.

Hybrid architecture. Our proposed learning approach combines the capabilities of LLMs with convolutional layers to dynamically select the most optimal configuration per language and task. We utilize LLMs to process input data and generate high-dimensional representations. These representations form an ND array, with dimensions corresponding to LLM model, prompt strategy, and embeddings. Convolutional layers with ND operate on this array, extracting relevant features across dimensions and learning indicative representations for optimal configurations. Subsequently, the output is the prediction of task accuracy (F1 Score) for the particular query across all configurations. This hybrid architecture integrates LLMs and convolutional layers, achieving superior performance in configuration selection. By comparing predicted accuracies, we dynamically select the optimal configuration for each query, tailored to specific task and language requirements. This approach

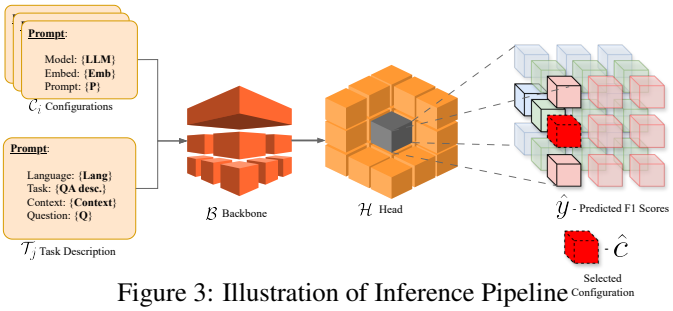

Figure 3: Illustration of Inference Pipeline

satisfies offline and online learning needs and adapts to various languages and datasets, ensuring effective configuration selection for diverse multilingual tasks.

Prior efforts like LOVM [51], [30]and HuggingGPT [42] have mainly concentrated on optimizing the model selection for a task. These methods typically consider only a single parameter selection at a time. In contrast, our approach concurrently selects the optimal configuration across three parameters: LLM model, prompt strategy, and embeddings. This introduces a complex and high-dimensional search space, posing challenges beyond the scope of iterative processes in prior efforts.

In our hybrid architecture, which combines LLMs with Convolutional ND layers, we predict the F1 score for each configuration per query. These scores form the basis for generating a *SoftMax* output, which serves as a probability density function. Sampling configurations from this distribution is crucial, especially in an online learning setting. This probabilistic sampling enables the control of entropy and encourages the exploration of diverse configurations, particularly with out-of-distribution data, thus helping to mitigate bias.

Architecture details. The architecture utilizes LLaMa-2-70B-hf model to produce embeddings. The traditional sampling head is replaced by a set of Conv-ND layers [47], denoted as $\mathcal{H}$, which predict the F1 Score for each configuration. The LLaMa-2-70B-hf [46] backbone, $\mathcal{B}$, embeds the Task Description $\mathcal{T}$, generating embeddings $\mathcal{E}_T$. The backbone $\mathcal{B}$ is also used to embed the individual configurations $\mathcal{C}_i$ into embeddings $\mathcal{E}_{C_i}$.

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 language models (`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 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 scores for all configurations.

$$\mathcal{E}_{T_j} \leftarrow \mathcal{B}(\mathcal{T}_j); \mathcal{E}_{C_i} \leftarrow \mathcal{B}(\mathcal{C}_i); \mathcal{E}_j \leftarrow \mathcal{E}_{T_j} \parallel \mathcal{E}_{C_i}; \hat{y} \leftarrow \mathcal{H}(\mathcal{E}_j) \tag{1}$$

Using the above, 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.

5.1 Training the Model for Both Online and Offline Setups.

To train the Backbone $\mathcal{B}$ and the Head $\mathcal{H}$, we employ two different loss functions based on the specific scenario: 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.

1. **Infer F1 Scores for All Configurations**: For each sample, infer the F1 scores for all possible configurations. This means computing the F1 scores for each combination of the language model, embedding model, and prompt strategies. For example, if there are three configurations for each parameter (e.g., three language models (`GPT-4Turbo`, `GPT3.5Turbo`,

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 6: 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 7: Online performance.

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.

2. **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.
3. **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.

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.

1. **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 or selects a specific configuration out of 60, we only compute the F1 score for that particular configuration.
2. **Obtain Ground Truth F1 Score**: Compute the actual F1 score for the selected configuration, which serves as the ground truth $y_{selected}$.
3. **Update Using Sparse Matrix**: Update the model using the sparse matrix of predicted F1 scores $\hat{y}$. Only the F1 score corresponding to the selected configuration is updated, leaving the other entries unaffected, thus reducing the computational overhead.
4. **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 loss function ensures the model is updated based on the selected configuration without needing the complete F1 score matrix. Implementation details of the above pipeline is explained in Appendix 10.

Train-Test Split. We split both datasets into three subsets, Offline training (60%), Online adaptation (20%), test set (20%), respectively. Furthermore, we train and evaluate the models using the two evaluation approaches with metrics MLQA-F1 and GPTAnnotator-F1.

5.2 Evaluation of Learning Approach

1. Offline Training Results. To evaluate our offline training phase, we compare against two baselines to gauge the effectiveness of our approach: (i) Randomly Selecting Configurations, representing sample variance, (ii) The Best Single Configuration, where the highest-scoring configuration across all samples is chosen. This ideal scenario reflects the performance when an optimal configuration is chosen as default. We evaluate the performance using multiple metrics, Accuracy for selecting configurations, F1 score for task performance with the selected configurations, i.e., **Acc@Top1** (how often the correct configuration is predicted by our model), **Acc@Top5** (the accuracy of correct configuration in the top 5 predictions) and F1 score accuracy at top 1 and top 5 (what is the F1 score obtained by executing the configurations selected with top1 and top5).

Table 6 shows the performance of our model against random and best single configuration with both MLQA-F1 and GPTAnnotator-F1 scores. Our model consistently outperforms random selection (by 17%) and best single configuration (by 11%) baselines for both scores (MLQA-F1 and GPTAnnotator-F1), indicating its ability to dynamically adapt and select configurations that optimize performance. Notably, our approach achieves a top 5 accuracy that matches the maximum achievable accuracy, underscoring its robustness in capturing diverse and potentially correct answers.

Evalaution	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 8: Learning approach performance on adaptation to unseen languages and datasets.

2. Online Training results: In online training, we assessed our model's ability to adapt to new data distributions. We used the same baseline methods as in offline training and compared their performance with our approach on 20% online adaptation set. For online adaptation, we employed parameters ($\mathcal{B}$ and $\mathcal{H}$) from the offline training at epoch 100 and further trained the model for 10 epochs on the online adaptation set. The performance in the online test phase (Table 7) showcases our model's remarkable adaptability, achieving F1 score accuracies at top 1 (60%) and top 5 (63%) closely matching the maximum achievable accuracy (63%). This indicates its consistency in capturing optimal configurations across diverse datasets and languages. Furthermore, our approach significantly outperformed baseline methods, surpassing random selection by 15% and the best single configuration by 7%. Even with minimal fine-tuning epochs, our approach demonstrates its effectiveness in adapting to new or out-of-distribution data.

3. Adaptation Efficacy: We now evaluate the adaptability of our approach.
(i) Adaptation to Unseen Languages: To assess our model's adaptability to newer downstream languages, we conducted experiments on languages not encountered during offline training. This is essential for evaluating its ability to generalize across diverse languages, crucial for real-world applications. 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. Our results, summarized in Table 8, demonstrate the model's remarkable adaptation capability. It achieves performance levels comparable to maximum achievable scores across Kannada, Tamil, and Telugu, showcasing effective generalization across diverse linguistic contexts. Compared to single best and random baselines, our model consistently outperforms them in terms of F1 scores across all evaluated languages, indicating its effectiveness in adapting to new languages.
(ii) Adaptation to Different Datasets: Adapting to diverse datasets with varying languages and query distributions is crucial for real-world applications. We trained the backbone and head of our model on the IndicQA dataset across all languages and tested on data from TyDiQA. Despite the small intersection of common languages between IndicQA and TyDiQA, our model demonstrated remarkable effectiveness, outperforming random selection by 11% and the best single strategy by 7% (see Table 8). With only 15 epochs of fine-tuning on 20% of TyDiQA, we achieved the maximum F1 score, emphasizing the importance of adapting to diverse datasets, languages, and query distributions.

Summary: *Our learning approach showcases significant improvements, underscoring its efficacy in dynamically selecting configurations, and adapting to new languages and datasets.*

6 Conclusions

In this work, we introduced a dynamic learning approach aimed at improving the performance of multilingual LLMs without requiring extensive training or fine-tuning. Our findings highlight several important insights. Firstly, we observed that prompting strategies lack universality, necessitating tailored approaches for different datasets, metrics, models, and languages. Additionally, our investigation into hybrid embeddings demonstrated performance enhancements, particularly with Cohere, which improved task performance by up to 7% on the GPT-4Turbo model. This underscores the significance of leveraging multilingual embeddings to augment LLM performance across varied linguistic contexts. Lastly, through offline and online assessments, our learning model surpassed baseline approaches by 15-20% in dynamically selecting configurations and adapting to new data distributions. Overall, our dynamic learning approach presents a promising avenue for enhancing the multilingual capabilities of LLMs, with implications for various real-world applications. Future research directions include further exploration of learning techniques, scalability to larger datasets, and the generalization of our approach to other NLP 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. As

the demand for multilingual language models continues to rise, our findings lay the foundation for future advancements for enhancing Polyglot LLMs performance in diverse linguistic contexts.

References

[1] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[2] K. Academy. Khanmigo. Website, 2024. URL https://www.khanmigo.ai/. Accessed: 2024-05-21.

[3] K. Ahuja, R. Hada, M. Ochieng, P. Jain, H. Diddee, S. Maina, T. Ganu, S. Segal, M. Axmed, K. Bali, et al. Mega: Multilingual evaluation of generative ai. *arXiv preprint arXiv:2303.12528*, 2023.

[4] S. Ahuja, D. Aggarwal, V. Gumma, I. Watts, A. Sathe, M. Ochieng, R. Hada, P. Jain, M. Axmed, K. Bali, et al. Megaverse: benchmarking large language models across languages, modalities, models and tasks. *arXiv preprint arXiv:2311.07463*, 2023.

[5] A. AI. Model card for claude 3, 2023. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. Accessed: 2024-05-21.

[6] S. AI. Announcing openhathi series. Sarvam AI Blog, 2024. URL https://www.sarvam.ai/blog/announcing-openhathi-series. Accessed: 2024-05-21.

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

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

[9] X. Chen, C.-M. Pun, and S. Wang. Medprompt: Cross-modal prompting for multi-task medical image translation. *arXiv preprint arXiv:2310.02663*, 2023.

[10] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 2020.

[11] Cohere. Introducing embed v3. Cohere Blog, 2024. URL https://cohere.com/blog/introducing-embed-v3. Accessed: 2024-05-21.

[12] M. corporation. Azure openai service. https://azure.microsoft.com/en-us/products/cognitive-services/openai-service, 2024.

[13] M. corporation. Gpt 3.5 turbo: Azure openai service. https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models#chatgpt-gpt-35-turbo, 2024.

[14] M. corporation. Gpt-4: Azure openai service. https://learn.microsoft.com/en-us/azure/cognitive-services/openai/concepts/models#gpt-4-models, 2024.

[15] M. corporation. Azure cognitive services translator services. https://learn.microsoft.com/en-us/azure/cognitive-services/translator/, 2024. URL https://learn.microsoft.com/en-us/azure/cognitive-services/translator/.

[16] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

[17] J. Gala, T. Jayakumar, J. A. Husain, M. S. U. R. Khan, D. Kanojia, R. Puduppully, M. M. Khapra, R. Dabre, R. Murthy, A. Kunchukuttan, et al. Airavata: Introducing hindi instruction-tuned llm. *arXiv preprint arXiv:2401.15006*, 2024.

[18] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[19] N. Goyal, J. Du, M. Ott, G. Anantharaman, and A. Conneau. Larger-scale transformers for multilingual masked language modeling. *arXiv preprint arXiv:2105.00572*, 2021.

[20] D. Green. Farmer chat. Website, 2024. URL `https://farmerchat.digitalgreen.org/`. Accessed: 2024-05-21.

[21] X. He, Z. Lin, Y. Gong, A. Jin, H. Zhang, C. Lin, J. Jiao, S. M. Yiu, N. Duan, W. Chen, et al. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.

[22] H. Huang, T. Tang, D. Zhang, W. X. Zhao, T. Song, Y. Xia, and F. Wei. Not all languages are created equal in llms: Improving multilingual capability by cross-lingual-thought prompting. *arXiv preprint arXiv:2305.07004*, 2023.

[23] K. Hämmerl, B. Deiseroth, P. Schramowski, J. Libovický, A. Fraser, and K. Kersting. Do multilingual language models capture differing moral norms?, 2022.

[24] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury. 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, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.560. URL `https://aclanthology.org/2020.acl-main.560`.

[25] S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave, et al. Muril: Multilingual representations for indian languages. *arXiv preprint arXiv:2103.10730*, 2021.

[26] M. Kuchnik, V. Smith, and G. Amvrosiadis. Validating large language models with relm. *Proceedings of Machine Learning and Systems*, 5, 2023.

[27] P. S. H. Lewis, B. Oguz, R. Rinott, S. Riedel, and H. Schwenk. MLQA: evaluating cross-lingual extractive question answering. *CoRR*, abs/1910.07475, 2019. URL `http://arxiv.org/abs/1910.07475`.

[28] P. Littell, D. R. Mortensen, K. Lin, K. Kairis, C. Turner, and L. Levin. 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, 2017.

[29] C. Liu, W. Zhang, Y. Zhao, A. T. Luu, and L. Bing. Is translation all you need? a study on solving multilingual tasks with large language models. *arXiv preprint arXiv:2403.10258*, 2024.

[30] X. Liu, R. Li, W. Ji, and T. Lin. Towards robust multi-modal reasoning via model selection. *arXiv preprint arXiv:2310.08446*, 2023.

[31] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong, Y. Pan, S. Xu, Z. Wu, Z. Liu, X. Zhang, S. Zhang, X. Hu, T. Zhang, N. Qiang, T. Liu, and B. Ge. Understanding llms: A comprehensive overview from training to inference, 2024.

[32] C. Malaviya, G. Neubig, and P. Littell. Learning language representations for typology prediction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark, September 2017.

[33] Microsoft. Introducing microsoft 365 copilot – your copilot for work. Microsoft Blog, 2023. URL `https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/`. Accessed: 2024-05-21.

[34] O. Ogundepo, T. R. Gwadabe, C. E. Rivera, J. H. Clark, S. Ruder, D. I. Adelani, B. F. Dossou, A. A. Diop, C. Sikasote, G. Hacheme, et al. Afriqa: Cross-lingual open-retrieval question answering for african languages. *arXiv preprint arXiv:2305.06897*, 2023.

[35] OpenAI. Gpt-4 technical report, 2023.

[36] OpenAI. New embedding models and api updates. OpenAI Blog, 2024. URL https://openai.com/index/new-embedding-models-and-api-updates/. Accessed: 2024-05-21.

[37] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[38] L. Qin, Q. Chen, Y. Zhou, Z. Chen, Y. Li, L. Liao, M. Li, W. Che, and P. S. Yu. Multilingual large language model: A survey of resources, taxonomy and frontiers, 2024.

[39] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL http://arxiv.org/abs/1606.05250.

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

[41] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[42] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

[44] S. Sitaram, M. Choudhury, B. Patra, V. Chaudhary, K. Ahuja, and K. Bali. Everything you need to know about multilingual LLMs: Towards fair, performant and reliable models for languages of the world. In Y.-N. V. Chen, M. Margot, and S. Reddy, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 21–26, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-tutorials.3. URL https://aclanthology.org/2023.acl-tutorials.3.

[45] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[46] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[47] J. P. Vizcaíno, F. Saltarin, Y. Belyaev, R. Lyck, T. Lasser, and P. Favaro. Learning to reconstruct confocal microscopy stacks from single light field images. *IEEE Transactions on Computational Imaging*, 7:775–788, 2021. doi: 10.1109/TCI.2021.3097611.

[48] X. Wang, Y. Tsvetkov, and G. Neubig. Balancing training for multilingual neural machine translation. *CoRR*, abs/2004.06748, 2020. URL https://arxiv.org/abs/2004.06748.

[49] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[50] H. Yang, J. Lin, A. Yang, P. Wang, C. Zhou, and H. Yang. Prompt tuning for generative multimodal pretrained models. *arXiv preprint arXiv:2208.02532*, 2022.

[51] O. Zohar, S.-C. Huang, K.-C. Wang, and S. Yeung. Lovm: Language-only vision model selection. *Advances in Neural Information Processing Systems*, 36, 2024.

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 [24]. 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[32] captured in lang2vec [28]. We give higher preference to the languages with Latin script since the languages with Latin script have shown better performance on GPTx models[3].

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**
    $w \leftarrow w_{Latin}$
  $score \leftarrow w \times d/l_{cls}$;
  return $score$

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**
      $L_{similar}.add(l)$

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` with text-embedding-ada-002 embeddings In Table.9 we observe similar trends to experiment with `IndicQA`, i.e., Each model have different trend across the different

	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 9: Performance of different Prompt strategies for `TyDiQA`

Table 10: 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 11: 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 12: 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 13: 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

prompt strategy and the choice of the metrics also favours different model making the it difficult to find a suitable choice of prompt for a generalized pipeline.

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

Table. 10, 11 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 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. 12, 13 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. 14 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

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 14: Hybrid approach performance - TyDiQA.

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 computational 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 $\hat{c} = C_{i_1,i_2,\ldots,i_m}$ 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:

$$\mathcal{M}_{j_1,j_2,\ldots,j_m} = \begin{cases} 1 & \text{if } (j_1, j_2, \ldots, j_m) = (i_1, i_2, \ldots, i_m) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

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**

$\mathcal{E}_{Tj} \leftarrow \mathcal{B}(\mathcal{T}_j)$; $\mathcal{E}_{Ci} \leftarrow \mathcal{B}(\mathcal{C}_i)$
$\mathcal{E}_j \leftarrow$ Concatenate$(\mathcal{E}_{Tj}, \mathcal{E}_{Ci})$
$\hat{y} \leftarrow \mathcal{H}(\mathcal{E}_j)$

/* Inference for selecting configuration */
if *Offline Setting* **then**
  $\hat{c} \leftarrow \arg\max(\hat{y})$
else if *Online Setting* **then**
  $\hat{c} \sim$ Softmax$(\hat{y})$

/* Training to update $\mathcal{H}$ & $\mathcal{B}$ */
if *Offline Setting* **then**
  $y \leftarrow$ Ground truth F1 scores $\forall \mathcal{C}_i$
  $Loss_{\text{off}} \leftarrow$ MSE$(\hat{y}, y)$
else if *Online Setting* **then**
  $y_{\text{sparse}} \leftarrow$ Ground truth F1 score for $\hat{c}$
  $\mathcal{M} \leftarrow$ Mask matrix using eq. 2
  $Loss_{\text{on}} \leftarrow$ MSE$(\mathcal{M} \odot \hat{y}, y_{\text{sparse}})$

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

10.1 Implementation Details

In this work, we use Azure OpenAI models [12] for all our LLM and embedding models including `GPT-4Turbo`, `GPT3.5Turbo` and `Mixtral`. Furthermore, for the learnign model, we train the Llama model on GPU with 4 x 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 [12].
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[3] where the context, the question related to the context, and the ground truth answer for each record are fetched from the IndicQA dataset[7]. 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.

[3] https://streamlit.io/

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`.