International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 4615-4625
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Finetuning XLM-Roberta Pretrained Models For Question Answering In Hindi

Nirja D Shah[1*], Jyoti Pareek[2]

[1]Department of Computer Science, Gujarat University, Ahmedabad, Gujarat, India,
*** Corresponding author Email**: nirjashah@gujaratuniversity.ac.in - **ORCID**: 0009-0002-9649-0616

[2]Department of Computer Science, Gujarat University, Ahmedabad, Gujarat, India.
Email: drjyotipareek@yahoo.com - **ORCID**: 0000-0002-6825-4803

Article Info:

DOI: 10.22399/ijcesen.2838
Received : 05 March 2025
Accepted : 30 May 2025

Keywords

Hindi NLP
Reading Comprehension
RoBERTa
Question Answering
NLP Finetuning

Abstract:

The paper intends to explore the development of a Hindi QA system using XLM-RoBERTa, trained on the Hindi subset of the chaii dataset. It tries to bridge the performance gap existing between low resource languages such as Hindi and high resource counterparts like English in the QA systems domain. We validate the model with systematic experimentation over a set of hyperparameters. The results reveal that relatively smaller learning rates, especially 0.00002, even with batch size 8, greatly enhance the performance with average BERTScore of 88.11. On the contrary, higher learning rates uniformly resulted in decreases in model performance. The batch size also mattered to performance but much less so than learning rate as lower batch sizes did not significantly degrade performance at lower learning rates. Further extensions to the above metrics depict good performance for smaller values of learning rates, with cases up to 21.07% above a BLEU score greater than 80 and 37.72% of cases with ROUGE1 F1 above 80. Such cases emphasize fine-tuning to be very important in QA tasks in low-resource languages. This paper contributes to understanding how QA systems can be optimized for Hindi and provides a benchmark for future research in this area

1. Introduction

Question Answering systems have emerged as one of the hot areas within the horizon of information retrieval systems. Notable development has been observed in QA systems in most of the well-resourced languages, while still there is a considerable gap in the development of the QA system for low resource languages like Hindi. There is a pressing need to fill this gap, considering that Hindi happens to be one of the most widely spoken tongues in the world, with millions of native speakers using digital systems to access information. Yet, constructing strong QA systems is challenging because high-quality, annotated datasets and the linguistic complexity of Hindi are scarce. It focuses on enhancing the performance of QA systems in Hindi with such state-of-the art transformer models like XLM-RoBERTa that have been pre-trained on multilingual corpora and optimized for cross-lingual tasks. Liu et al. (2019) [1] introduced RoBERTa, a robustly optimized BERT model and presented evidence to demonstrate the impact of hyperparameter tuning and additional training data in improving model performance. Staliūnaitė et al. (2020) [2] critically tested the linguistic skills of RoBERTa, BERT, and DistilBERT and demonstrated the shortcomings of compositional and lexical semantics like arithmetic, negation, and semantic reasoning in them through a CoQA case study. To address these deficiencies, the authors suggest that multitask learning can be incorporated through enriching models with linguistic knowledge. This improves performance on complex question types. This study illustrates possibilities of adding more linguistic information during pre-training to better understand complex structures in language. McCarley et al. (2021) [3] focused on efficiency of the model by exploring structured pruning and task-specific distillation methods for BERT-based question answering models. Their results indicate a significant reduction in model size and inference

time without noticeable accuracy losses, which goes in the direction of more efficient NLP systems. This contribution further enriches the existing literature on model compression, which is highly relevant for deployment of NLP models in resource-constrained environments.Researchers find ways to optimize the performance of QA, and this includes optimizing the objectives of pre-training, leveraging linguistic knowledge, and innovating new model architectures. Jia et al. (2021) [4] proposed a Question Answering Infused Pre-training or QUIP, which showed that training a bi-encoder model on synthetic QA pairs may lead to good performance on a wide array of NLP tasks. Mayeesha et al. [5] overcame the challenge of low-resource languages by successfully applying transfer learning using multilingual BERT models for Bengali QA. Warstadt et al. [6] investigated acquisition of linguistic generalizations by PLMs and concluded that a great deal of training data is required to favour linguistic features over surface-level patterns. Together, these studies demonstrate the feasibility of PLMs for QA while indicating areas to further improve efficient methods for their own pre-training, ways to alleviate data scarcity, and increasing models' capacity to learn and apply linguistic knowledge.Integrating ensemble techniques into the transformer models ALBERT, RoBERTa, and ELECTRA increases the accuracy in span prediction. Bachina et al. [7] demonstrated in 2021 that fine-tuning these models on datasets such as SQuAD2.0, Natural Questions and CORD-19 with ensemble methods significantly improved the performance of QA systems with enhanced Exact Match and F1 scores. Such findings show the promise of ensemble techniques where strengths of multiple models can be leveraged to make more accurate predictions for QA tasks. Chaybouti et al. [8] worked on the efficiency and scalability aspects of QA systems for real world applications by using siamese networks along with RoBERTa-based models. The model has been fine-tuned on the SQuAD v1.1 dataset, and superior performance on the PIQA benchmark demonstrates that it is much more efficient than the previous state-of-the-art model, DENSPI. Comparing IndoBERT-lite and RoBERTa for QA applications in the Indonesian language, Richardson and Wicaksana [9] concluded that IndoBERT-lite-that is the model based 5 on the ALBERT architecture-excelled RoBERTa with accuracy and F-score when tested on Indonesian datasets that included TyDi QA and the Indonesian-translated SQuAD.The primary goal of the work is to build an extremely high performing QA system in Hindi on the chaii dataset. It focuses on the problem of considerable performance differences between QA systems that could potentially interact with languages like Hindi as opposed to English. Since state-of-the-art models for English have achieved superior performance on tasks such as machine reading comprehension and extractive QA, their counterpart in Hindi has received poor performance due to less availability of linguistic resources, inadequate annotated datasets, and the complexity of the language itself. Thus, the target would be to fill the gap by fine-tuning XLM-RoBERTa on the Hindi subset of the chaii dataset, testing the performance of the system on multiple metrics, and trying to analyse how such changes of hyperparameter impacts, for example, the learning rate and batch size. The paper comprehensively describes how XLM-RoBERTa performs in Hindi QA tasks and analysis on the hyperparameter tuning strategy. Secondly, it throws light on the challenges and nuances of building QA systems for Hindi language and steps into key linguistic complexities related to morphology and word inflections. The models were trained on high-quality, annotated data, realizing the cross-lingual capabilities that can make this project contribute to the rising body of work focused on improving NLP tools for Hindi and paving the way for more inclusive digital ecosystems that are resource-rich

2. Method

Data cleansing is part of the very first step involved in NLP pipelines after acquiring raw data to remove inconsistencies, noise, and irrelevant information. These involve steps like text processing that includes breaking down of the text into sub words called tokenization, removal of stop words that carry very little meaning (e.g. 'like', 'and', 'of'), lemmatizing of words to their root form, Part-of-Speech (POS) tagging where grammatical tags (e.g. noun, verb, adjective) are applied to words and named entity recognition (NER) to identify and classify named entities (e.g. persons, organizations, locations). The pre-processed text is then represented in numerical forms to be used for algorithms during the training

process. Traditional methods for textual representation include Bag-of-Words (BoW), Term frequency- inverse document frequency (TF-IDF), Word2Vec, and FastText, having its set of limitations towards representing the contextual content of the text. Unlike BoW and the TF-IDF, they focus on the context of word. As Word2Vec performs at the word level, FastText utilizes information at the character level to produce a word vector, which is beneficial in handling out-of-vocabulary words and morphologically diverse languages. Though the improvement over BoW has several constraints, both models fail to handle polysemy, or when a word has more than one meaning as both models represent each word as a single vector.

Also, it may not capture out-of-vocabulary words or words that may rarely occur within a text. As such, FastText mitigates somewhat such a problem by encompassing subword information; however, it does not do very well in case of totally novel words across the network.Words' contextualized representation is a situation where word embeddings will consider the relative position and context by processing the whole text, which supports better prediction abilities for machine learning algorithms on challenging tasks, such as the question answer prediction using NLP. Since the objective is to create a deep learning model for the Hindi question-answering task, then it would be best if a transformer-based model could be trained from scratch on the Hindi corpus, and then finetune the model and parameters for the question-answering task. As training from scratch is a resource-intensive process from both temporal and financial viewpoints, we will use the transfer learning process, which involves the technique of representing language in an unsupervised method over a large dictionary of words and modifying its architecture to suit question answering.

The method uses much fewer resources for training since the transformer models are pretrained on large corpus and need to be fine-tuned for the downstream tasks. We have chosen the RoBERTA model for performing the Hindi question answering tasks, the components of which are described in the below sections.

2.1 RoBERTA

RoBERT is the fine-tuned version of the BERT model by Facebook AI, aimed at overcoming some of the BERT model's limitations and optimizing the training efficiency of BERT in handling the core and advanced tasks of NLP. RoBERTa retains the base architecture of the BERT model, which is a multilayer transformer, but introduces very critical changes in its design from pre-training. These updates included training on more data, the application of longer sequences, and the use of a dynamic masking strategy in place of the static masking strategy applied in BERT. Dynamic masking points that masked positions in the input text change with each epoch of training, which would expose RoBERTa to more diverse contexts of each word. In addition, the RoBERTa model deleted the Next Sentence Prediction objective; this was the objective introduced in the BERT architecture. They did find that NSP added relatively little value to the performance and removal resulted in more focused learning on the MLM task, enhancing the efficiency of the model. The architecture is also pre-trained on a much larger dataset, over 160GB of text - several times bigger than what was used for BERT - thus further enhancing the generalizability and performance of RoBERTa across a wide variety of tasks.Suwarningsih et al. (2022) [10] proved the superiority of RoBERTa over competitive models in building an Indonesian QA system, given the flexibility of the model. This is consistent with Pearce et al. (2021) [11], which provides an inclusive comparison of many PLMs over different QA datasets, that makes RoBERTa a robust performer. In Yu et al. (2021) [12], authors detail new encoder-decoder architectures based on RoBERTa, that embeds attention mechanisms into it to be able to capture complex semantic and syntactic relationships between 4 words within text, showing the future prospect of hybridizing PLMs with more traditional NLP techniques.Applied to the task of QA, RoBERTa has been similarly used to BERT but with all the advantages bestowed by its strong regime of training and architectural improvements. RoBERTa, further to this, is fine-tuned on the QA dataset consisting of pairs of questions along with their contexts within which the model must predict the appropriate span of text within the passage that answers a question. Standard input format to the RoBERTa model in QA tasks is just a concatenation

of a question with the passage and going through special tokens: `[CLS] Question [SEP] Passage [SEP]`. At fine-tuning time, the RoBERTa model learns to predict the start position and the end position of the answer span within the passage. It will choose the token span that maximizes the product of the start and end probabilities-a formulation that essentially defines the most likely span as an answer to the question. Part of RoBERTa's excellence in QA tasks is because it employs a refined pre-training approach, which in turn allows it to better understand and retrieve relevant information from a passage.

2.2 Dataset Composition

Dataset for the Conversational AI for Indic languages come under the chaii competition on Kaggle that focuses on advancing question answering systems for low resource languages like Hindi and Tamil. It falls under the machine reading comprehension challenge, wherein one attempts to build models to read a passage and extract an answer specific to that question from context provided. Each of the examples comprises a context or passage and a question with its relevant answer(s). The answer is typically a stretch of text from the context itself, plus start and end indices to give the position of the answer in the passage. In NLP tasks, the Hindi component of the chaii dataset is an important resource, especially for use in question answering systems. It focuses on extractive QA, where models are trained to find and return exact spans of text that would respond to specific questions. With Hindi among these languages, it is highly relevant for developing and fine-tuning models for multilingual and low-resource languages as Hindi is one of the most widely spoken languages in the world, yet it has always missed substantial digital resources for NLP when compared to those of languages like English [13, 14]. Training on this dataset will boost not only Hindi QA performance but also general cross-lingual capacities of the model, thus making it better to understand and generate language in a wide context and across domains. Table 1 gives us the statistics of the token counts and character lengths for contexts in the dataset, we see the context to question ratio is 1.19 meaning most contexts just have 1 question along with it. Hence our objective is to train on 746 QA pairs as seen in Table 2. As visible in Table 3, most answers have an average of just 2-3 tokens.

Table 1. *Summary statistics of token counts and character lengths for contexts in chaii dataset*

Language	#Contexts	Token count			Character count		
		Avg	Min	Max	Avg	Min	Max
Hindi	623	1854.57	24	10,259	10,145.06	176	49,289
Tamil	301	1369.28	50	5,791	12,730.59	446	49,815

Table 2. *Summary statistics of token counts and character lengths for questions in chaii dataset*

Language	#Questions	Token count			Character count		
		Avg	Min	Max	Avg	Min	Max
Hindi	746	8.29	4	22	42.67	22	121
Tamil	368	4.77	3	9	39.59	19	79

Table 3. *Summary statistics of token counts and character lengths for answers in chaii dataset*

Language	Avg answers per context	Token count			Character count		
		Avg	Min	Max	Avg	Min	Max
Hindi	1.19	2.24	1	51	12.46	1	239
Tamil	1.22	1.92	1	32	13.33	1	286

2.3 Experimental Environment

To ensure efficient parallel computing of the deep learning models, the experiments were conducted within a computing environment comprising a system running Windows 11 and equipped with an 12th Gen Intel(R) Core(TM) i7-12700H 2.70 GHz processor. The system was configured with 32 GB of DDR5 RAM operating at 5200 MHz, providing ample memory bandwidth for computational tasks. Graphics processing was facilitated by an NVIDIA GeForce RTX 3070 Laptop GPU and DCH Driver with 5,888 CUDA cores featuring 8 GB of GDDR6 memory, enabling efficient parallel processing for

accelerated computations. The conda 23.1.0 distribution was employed to manage Python dependencies and packages, with the experiments executed using Python version 3.10. This robust experimental setup ensured optimal performance and compatibility, laying the foundation for rigorous and reliable experimentation throughout the study.

2.4 Data Preprocessing

Externally visible characters and words may be present within raw data, due to which the classification may be hindered. We do not provide raw data to the classifier directly. Instead, we preprocess the data and feed pre-processed data into the classifier with the hope that it will improve the performance of the model. Raw data may comprise a large number of special characters such as #, %, *, ., -, $ etc. This decreases the accuracy and thus needs to be removed from the dataset. The dataset contains many Hindi stop words that add nothing toward prediction tasks. These may pose an obstacle when searching for higher accuracy. There is a deletion of these stop words, which has proven to improve the model's performance. Hindi words sometimes spell differently. For example, '*खेल*' takes a spelling like '*खेलो*', '*खेलता*', '*खेली*', '*खेला*', '*खेलूँगा*', '*खेलेंगे*', etc. To handle the corpus effectively, we apply stemming and lemmatization techniques to process only the root word ('*खेल*'). Below are examples of raw and preprocessed data. The preprocessed version performs better than the raw data.

Raw Data: *भारतीय क्रिकेट टीम का गठन 1926 में हुआ था। टीम ने अपना पहला अंतरराष्ट्रीय मैच 1932 में इंग्लैंड के खिलाफ खेला। भारतीय टीम ने 1983 और 2011 में क्रिकेट विश्व कप जीता। भारतीय क्रिकेट टीम के वर्तमान कप्तान रोहित शर्मा हैं।*

Preprocessed Data: *भारतीय, क्रिकेट, टीम, गठन, 1926, मैच, 1932, इंग्लैंड, खेल, टीम, विश्व कप, 1983, 2011, जीता, भारतीय, कप्तान, रोहित, शर्मा.*

2.5 Hyperparameter Tuning

Hyperparameter tuning is key to getting good results in training deep learning models, particularly for complex tasks like natural language understanding and question answering [15, 16]. Examples of such important parameters are learning rate, batch size, and input sequence length, which affect model performance and stability and even control the efficiency of the training process. We fine-tuned XLM-RoBERTa models with varying hyperparameters: changing different batch sizes (4 and 8) and different learning rates that varied in a range from 2.00E-05 up to 2.00E-01, with a fixed maximum sequence length of 484 tokens. The learning rate will determine how fast or slow a model updates its weights while training. If the learning rate is too high, the model will converge too fast to a suboptimal solution or might not at all converge to anything, which will increase the instability of training. If the learning rate is too low, then training could take an interminably long period since the updates on the weights are very small; hence, it would get stuck in local minima. Thus, the choice of an optimal learning rate is expected for the suitable attainment in an appropriate time frame for good performance [17].We tested through learning rates from 2.00E-05 to 2.00E-01. The lowest value, 2.00E-05, is usually used to fine-tune large pre-trained models like XLM-RoBERTa since the model performs very small, controlled weight updates and thus avoids overfitting. On the other hand, large learning rates, such as 2.00E-01, are useful at early training stages with rapid convergence but are more likely to miss finer details in the data. In our experiments, we used a whole range of learning rates and saw where there is a trade-off between fast convergence and stable, fine-grained learning. For instance, as 2.00E-05 is the default tunable one, testing out more aggressive rates like 2.00E-03 or 2.00E-02 let us find whether we could reach comparable or even better results with faster training which would be of especially high interest for Hindi.The batch size corresponds to the number of training samples that are allowed to pass one forward and backward step through the model during training. Larger batch sizes generally enable more stable gradient estimates and faster training, by making fuller use of the parallel processing capability inherent in modern GPUs [18]. They require more memory and may become impractical for large models like XLM-RoBERTa. Conversely, smaller batch sizes result in noisier gradient estimates but require fewer memory allocations, so that there can be more iterations over the data, which

may yield better generalization. We tried batch sizes of 4 and 8. Having smaller batches, such as 4, allows the model to do the weight updates more frequently in these episodes, which might actually be good for fine-tuning since the model is forced to adapt more often, and it will lead to better performance over smaller datasets or when the task is highly complex, as for example in the case of multilingual QA [19, 20]. A small batch size hurts training speed even more, particularly for larger models, but doubling the batch size to 8 seems to have a positive effect on reducing the variance of updates of the gradient and stabilizing the process of learning at the cost of increased memory usage and possibly slower iteration convergence.Maximum sequence length is the number of tokens that this model views from each input context. With a set value too low, it may truncate some of the important parts of an input; with languages like Hindi, word counts may be relatively high to even get concepts out. The maximum length should not be set too large as it would result in over computation and increased memory utilization since the model has to consume more tokens than it needs. We have a maximum length of 484 tokens, which we determined based on the average context length on chaii Hindi. This way, most contexts and questions will fit within that number without sacrificing much performance due to the loss at the end of truncation.

3. Results and Discussion

3.1. Evaluation Criteria

In evaluating a question-answering (QA) model, it is essential to use appropriate evaluation metrics that not only quantify how well the model's predicted answers align with the ground truth but also offer insights into the quality of those answers. We employed six widely used metrics: BLEU score, METEOR score, BERTScore, ROUGE1_F1, ROUGE2_F1, and ROUGEL_F1. Each of these metrics is vital for capturing different aspects of the predicted answers, such as syntactic accuracy, semantic relevance, and linguistic variety.

3.1.1. BLEU (Bilingual Evaluation Understudy)

BLEU is a precision-based metric designed to evaluate the quality of machine-generated text by comparing it with human-generated reference texts. It was originally developed for machine translation, but it has since been widely adopted for other natural language processing (NLP) tasks [21], including question answering. BLEU helps measure how much of the predicted answer is present in the reference answer by comparing n-grams (word sequences).

3.1.2. METEOR (Metric for Evaluation of Translation with Explicit ORdering)

METEOR is another widely used metric originally designed for machine translation [22]. Unlike BLEU, which focuses solely on precision, METEOR incorporates both precision and recall. It also includes stemming, synonyms, and word-ordering features, making it a more linguistically grounded evaluation measure. METEOR is especially useful for evaluating QA models because it rewards synonym matches and accounts for word-order differences.

3.1.3 BERTScore

BERTScore is a more recent evaluation metric that leverages the power of transformer-based models (like BERT) to compute semantic similarity between the candidate and reference texts. Unlike BLEU and METEOR, which rely on n-gram matching, BERTScore uses contextualized embeddings to capture semantic similarities, making it highly suitable for complex language tasks. BERTScore is particularly effective for QA tasks where exact token matches may not fully capture the quality of the answer.

3.1.4 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

The ROUGE metric family is widely used for evaluating text summarization and question answering systems. ROUGE measures overlap between the n-grams of the candidate and reference texts, emphasizing recall over precision. There are several variants of ROUGE, but the most relevant for QA tasks are ROUGE-1, ROUGE-2, and ROUGE-L, each of which measures different types of overlap. ROUGE-1 measures the overlap of unigrams between the candidate and reference. ROUGE-2 measures the overlap of bigrams. ROUGE-L measures the longest common subsequence (LCS) between the candidate and reference, emphasizing fluency and word order.

Table 4. *Average BERTScore for predicted answers with 10,000 SQuAD data test cases*

Batch size	Learning rate	Max length	Average BERTScore
8	0.00002	484	88.11
4	0.00002	484	87.99
8	0.0002	484	83.00
8	0.002	484	67.36
4	0.002	484	66.98
8	0.02	484	65.67
4	0.02	484	55.46
4	0.2	484	46.74
8	0.2	484	46.74

3.2. Results

Table 4 highlights the average BERTScore for predicted answers across different batch sizes, learning rates, and a consistent max length of 484. From the table, we observe a general trend that learning rate plays a significant role in determining the quality of the predictions. Smaller learning rates, particularly 0.00002, yield higher BERTScores, with the highest being 88.11 for a batch size of 8. As the learning rate increases, the BERTScore consistently drops, with the lowest scores of 46.74 observed for a learning rate of 0.2. This suggests that larger learning rates may cause the model to converge too quickly, leading to suboptimal generalization. Additionally, the batch size also influences performance, but the effect appears less pronounced than the learning rate. For example, at a learning rate of 0.00002, the BERTScore remains similar between batch sizes of 4 and 8, indicating that smaller batches do not dramatically impact this metric at lower learning rates.Table 5 presents the percentage of cases achieving a BLEU score and METEOR score above 80, which is a critical threshold for high-quality question-answering performance. The results reveal important insights into the effects of varying batch sizes and learning rates on model performance. The highest percentages of cases with an 80+ BLEU score and METEOR score are observed for the smallest learning rate of 0.00002, with 21.07% of cases achieving a BLEU score above 80 for a batch size of 8 and 22.87% for METEOR with a batch size of 4. This demonstrates that a lower learning rate allows the model to make more accurate predictions, resulting in higher quality answers. However, as the learning rate increases, the percentage of cases with scores above 80 drastically decreases, indicating that larger learning rates may cause the model to overfit or converge too quickly, reducing its ability to generate high-quality answers. For instance, at a learning rate of 0.002 or higher, almost no cases reach the 80+ score threshold, with percentages as low as 0.01% for BLEU and 0.22% for METEOR. The batch size also plays a role, although its impact is less pronounced than the learning rate. The highest percentages are achieved with a batch size of 8 and a learning rate of 0.00002.

Table 5. *Percentage of 10,000 SQuAD data test cases with BLEU score and METEOR score above 80*

Batch size	Learning rate	Max length	% of cases with 80+ BLEU score	% of cases with 80+ METEOR score
8	0.00002	484	21.07	22.81
4	0.00002	484	20.93	22.87
8	0.0002	484	14.55	16.04
4	0.002	484	0.04	0.26
8	0.02	484	0.02	0.26
8	0.002	484	0.01	0.22
4	0.02	484	0	0.15
4	0.2	484	0	0
8	0.2	484	0	0

Table 6 illustrates the percentage of 10,000 SQuAD test cases where the model's predicted answers achieved an 80+ ROUGE F1 score across three different ROUGE variants: ROUGE1, ROUGE2, and ROUGEL. These metrics are critical in evaluating the performance of models trained to answer questions based on the SQuAD dataset, as they measure how well the predicted answer matches the reference answer, both in terms of word overlap and sentence structure. From the results in the table, it's clear that the model's performance, as reflected by the percentage of test cases exceeding the 80+ threshold in ROUGE F1 scores, significantly varies across different batch sizes and learning rates. The model trained with a batch size of 8, learning rate of 0.00002, and a max length of 484 performs the best, with 37.72%, 21.75%, and 37.53% of the cases achieving ROUGE1, ROUGE2, and ROUGEL F1 scores above 80, respectively. This suggests that smaller learning rates and larger batch sizes lead to more accurate and contextually appropriate answers that align well with the reference answers In contrast, models trained with higher learning rates, such as 0.02 and 0.2, achieve almost negligible percentages of cases with ROUGE scores above 80. This indicates that larger learning rates cause the model to diverge, producing lower-quality predictions that fail to capture the essential content or structure of the reference answers. For instance, the model trained with a batch size of 8 and a learning rate of 0.02 only achieved 0.08% for ROUGE1, 0.02% for ROUGE2, and 0.08% for ROUGEL, indicating a significant drop in performance. The importance of ROUGE scores in this evaluation is tied to their ability to measure different levels of answer quality. ROUGE1 emphasizes the inclusion of key words, while ROUGE2 adds a focus on short phrases and syntactic coherence. ROUGEL, on the other hand, provides a more holistic view of how well the predicted answer retains the sequence and structure of the reference answer. Achieving an 80+ score in these metrics is a strong indicator that the model is generating coherent, contextually accurate, and meaningful answers that closely resemble human-generated responses.

3.3 Discussion

In this study, we explored the performance of XLM-RoBERTa for Hindi question-answering (QA) tasks using the chaii dataset, focusing on the impact of various hyperparameters such as learning rate, batch size, and maximum sequence length. The primary objective was to assess how well the model could generate contextually appropriate answers for Hindi questions and provide insights into the optimization strategies that can enhance model performance for low-resource languages. Our results demonstrate that the choice of learning rate has the most significant impact on model performance. Specifically, smaller learning rates, such as 0.00002, consistently yielded better results across key evaluation metrics, including BERTScore, BLEU, METEOR, and ROUGE. For example, the highest BERTScore of 88.11 was achieved at a learning rate of 0.00002 with a batch

.

Table 6. *% of 10,000 SQuAD data test cases with ROUGE1, ROUGE2 and ROUGEL score above 80.*

Batch size	Learning rate	Max length	% of cases with 80+ Rouge1 F1 score	% of cases with 80+ Rouge1 F2 score	% of cases with 80+ RougeL F1 score
8	0.00002	484	37.72	21.75	37.53
4	0.00002	484	37.22	21.6	36.99
8	0.0002	484	26.73	15.14	26.5
8	0.02	484	0.08	0.02	0.08
4	0.002	484	0.05	0.04	0.05
4	0.2	484	0.03	0.01	0.03
8	0.002	484	0.03	0.02	0.03
8	0.2	484	0.03	0.01	0.03
4	0.02	484	0.02	0	0.02

size of 8, while larger learning rates like 0.2 resulted in a steep decline in performance, with BERTScore dropping as low as 46.74. This trend was consistent across all metrics, as larger learning rates led to poorer generalization and lower-quality predictions, suggesting that the model was converging too quickly and failing to capture nuanced relationships in the data. Batch size, although important, had a less pronounced effect than the learning rate. At the optimal learning rate of 0.00002, the model's performance remained relatively stable between batch sizes of 4 and 8. However, as the learning rate increased, the choice of batch size became more critical, with larger batch sizes exacerbating the decline in performance. The experiments showed that while increasing batch size improves computational efficiency, it must be balanced carefully with the learning rate to avoid overfitting or underfitting. The evaluation using various metrics highlighted important differences in how each metric assesses the model's output. BERTScore, which is based on contextual embeddings, consistently yielded higher scores than BLEU, METEOR, and ROUGE. This is because BERTScore is more forgiving of variations in word choice and syntax if the overall meaning of the answer is preserved. In contrast, BLEU and ROUGE, which focus on n-gram overlap, penalized the model for paraphrasing or using alternate but contextually appropriate words. This difference suggests that for question-answering tasks in low-resource languages like Hindi, metrics that emphasize semantic similarity, such as BERTScore, may be more appropriate for evaluating model performance than metrics like BLEU or ROUGE, which rely on exact word matches.

Table 7*. Average scores on test data for different metrics for model trained on batch size 8, learning rate 2.00E-05 and max length of 484.*

Metric	Average value
Bertscore	88.11
Rouge1_f1	52.96
Rougel_f1	52.79
Meteor_score	39.32
Bleu_score	33.97
Rouge2_f1	27.17

Table7 indicates that the model achieves a relatively high BERTScore of 88.11, while other metrics such as ROUGE1, ROUGEL, METEOR, ROUGE2, and BLEU have significantly lower averages. BERTScore relies on contextual embeddings generated by the BERT model, which means it captures semantic similarity between words in the reference and predicted answers. It focuses on meaning and word usage in context, making it more forgiving of minor variations in word choice or phrasing, if the overall meaning is preserved [23]. This makes it less sensitive to exact word overlap or structure, leading to higher scores in many cases. ROUGE (especially ROUGE1 and ROUGE2) and BLEU focus on n-gram overlap (unigrams, bigrams, etc.), where exact word matches between the predicted and reference answers are crucial. These metrics penalize models that paraphrase or use different but contextually appropriate words, which could explain why their scores are lower. Additionally, the lower ROUGE2 (27.17) indicates the model struggles more with capturing longer phrases (bigrams), which affects BLEU (33.97) as well, since it also emphasizes exact n-gram matches.

4. Conclusion and Future Work

While this study has provided valuable insights into the performance of XLM-RoBERTa for Hindi QA, there are several areas for further research that could enhance the model’s effectiveness and generalizability. One promising avenue is the exploration of alternative pre-trained models. Although XLM-RoBERTa has demonstrated strong results, other models such as mBERT, T5, or newer architectures like Mistral and XLM-T could offer different advantages [24]. These models may capture language nuances differently due to their unique pre-training objectives. For instance, T5’s text-to-text framework could be particularly useful in improving the generation and comprehension of QA systems in low-resource languages like Hindi.Another important area for future work is the incorporation of additional data sources. Given the limited availability of large-scale datasets for Hindi, expanding the dataset is crucial. Cross-lingual transfer learning, where models trained on high-resource languages are fine-tuned on Hindi, is one possible solution. Synthetic data generation and using multilingual corpora that include Hindi and other Indian languages could also significantly boost model performance [25, 26]. Additionally, domain adaptation techniques could allow the model to generalize better by training on datasets from

domains like the chaii dataset. Fine-tuning the model with advanced optimization techniques offers another potential improvement. This study used a standard fine-tuning process, but advanced methods like gradient accumulation, cyclic learning rates, or differential learning rates for different layers of the model could lead to better convergence and overall performance. These techniques could help stabilize the model during training, especially when dealing with challenging datasets, leading to improved predictions.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Y. Liu *et al. (2019).* RoBERTa: A Robustly Optimized BERT Pretraining Approach, *arXiv (Cornell University)*, vol. 1., doi: https://doi.org/10.48550/arxiv.1907.11692

[2] I. Staliūnaitė and I. Iacobacci, (2020). Compositional and Lexical Semantics in RoBERTa, BERT and DistilBERT: A Case Study on CoQA, *arXiv (Cornell University)*, doi: https://doi.org/10.48550/arxiv.2009.08257.

[3] J. S. McCarley, R. Chakravarti, and A. Sil, (2021). Structured Pruning of a BERT-based Question Answering Model, *arXiv.org*. https://arxiv.org/abs/1910.06360

[4] R. Jia, M. Lewis, and L. Zettlemoyer, (2021). Question Answering Infused Pre-training of General-Purpose Contextualized Representations, *arXiv (Cornell University)*. doi: https://doi.org/10.48550/arxiv.2106.08190

[5] T. Tahsin Mayeesha, A. Md Sarwar, and R. M. Rahman, (2020). Deep learning based question answering system in Bengali, *Journal of Information and Telecommunication*, doi: https://doi.org/10.1080/24751839.2020.1833136

[6] A. Warstadt, Y. Zhang, H.-S. Li, H. Liu, and S. R. Bowman, (2020). Learning Which Features Matter: RoBERTa Acquires a Preference for Linguistic Generalizations (Eventually), *arXiv (Cornell University)*. doi: https://doi.org/10.48550/arxiv.2010.05358

[7] Sony Bachina, Spandana Balumuri, and Sowmya Kamath S, (2021). Ensemble ALBERT and RoBERTa for Span Prediction in Question Answering. doi: https://doi.org/10.18653/v1/2021.dialdoc-1.9

[8] Sofian Chaybouti, Achraf Saghe, and Aymen Shabou, (2021). EfficientQA : a RoBERTa Based Phrase-Indexed Question-Answering System, *arXiv (Cornell University)*. doi: https://doi.org/10.48550/arxiv.2101.02157

[9] B. Richardson and A. Wicaksana, (2022). Comparison Of Indobert-Lite And Roberta In Text Mining For Indonesian Language Question Answering Application, *International Journal of Innovative Computing, Information and Control ICIC International c*, vol. 2022(6), doi: https://doi.org/10.24507/ijicic.18.06.1719

[10] Wiwin Suwarningsih, Raka Aditya Pramata, Fadhil Yusuf Rahadika, and Mochamad Havid Albar Purnomo, (2022). RoBERTa: language modelling in building Indonesian question-answering systems, *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 20(6), doi: https://doi.org/10.12928/telkomnika.v20i6.24248

[11] K. Pearce, T. Zhan, A. Komanduri, and J. Zhan, (2021). A Comparative Study of Transformer-Based Language Models on Extractive Question Answering, *arXiv.org*. https://arxiv.org/abs/2110.03142

[12] P. Yu and Y. Liu, (2021). Roberta-based Encoder-decoder Model for Question Answering System. doi: https://doi.org/10.1109/icaa53760.2021.00070

[13] B. S. Harish and R. K. Rangan, (2020). A comprehensive survey on Indian regional language processing, *SN Applied Sciences*, vol. 2(7), doi: https://doi.org/10.1007/s42452-020-2983-x

[14] K. Sourabh and V. Mansotra, (2012). An Experimental Analysis on the Influence of English on Hindi Language Information Retrieval, *International Journal of Computer*

Applications, vol. 41(11), doi: https://doi.org/10.5120/5587-7832.

[15] J. A. Ilemobayo *et al.*, (2024). Hyperparameter Tuning in Machine Learning: A Comprehensive Review, *Journal of Engineering Research and Reports*, vol. 26(6), doi: https://doi.org/10.9734/jerr/2024/v26i61188

[16] L. Liao, H. Li, W. Shang, and L. Ma, (2022). An Empirical Study of the Impact of Hyperparameter Tuning and Model Optimization on the Performance Properties of Deep Neural Networks, *ACM Transactions on Software Engineering and Methodology*, vol. 31(3), doi: https://doi.org/10.1145/3506695

[17] Y. Ding, (2021). The Impact of Learning Rate Decay and Periodical Learning Rate Restart on Artificial Neural Network, doi: https://doi.org/10.1145/3460268.3460270

18] S.-Y. Zhao, Y.-P. Xie, and W.-J. Li, (2020). Stagewise Enlargement of Batch Size for SGD-based Learning, *arXiv.org*, https://arxiv.org/abs/2002.11601

[19] D. Masters and C. Luschi, (2018). Revisiting Small Batch Training for Deep Neural Networks, *arXiv:1804.07612 [cs, stat]*, https://arxiv.org/abs/1804.07612

[20] C. Simionescu, G. Stoica, and R. Herscovici, (2022). Dynamic Batch Adaptation, *arXiv.org*. https://arxiv.org/abs/2208.00815v1

[21] A. Yang, K. Liu, J. Liu, Y. Lyu, and S. Li, (2018). Adaptations of ROUGE and BLEU to Better Evaluate Machine Reading Comprehension Task, *arXiv.org*, https://arxiv.org/abs/1806.03578

[22] S. Banerjee and A. Lavie, (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. https://aclanthology.org/W05-0909.pdf

[23] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, (2020). BERTScore: Evaluating Text Generation with BERT, *arXiv:1904.09675 [cs]*, https://arxiv.org/abs/1904.09675

[24] D. Amin, S. Govilkar, and S. Kulkarni, (2023). Question answering using deep learning in low resource Indian language Marathi, *arXiv.org*, https://arxiv.org/abs/2309.15779

[25] A. Prabhakar, G. S. Majumder, and A. Anand, (2022). CL-NERIL: A Cross-Lingual Model for NER in Indian Languages (Student Abstract), *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36(11), doi: https://doi.org/10.1609/aaai.v36i11.21652

[26] GeMQuAD : Generating Multilingual Question Answering Datasets from Large Language Models using Few Shot Learning, *Arxiv.org*, 2023. https://arxiv.org/html/2404.09163v1