Indic-Transformers: An Analysis of Transformer Language Models for Indian Languages

Kushal Jain[1]*, **Adwait Deshpande**[2]*, **Kumar Shridhar**[1], **Felix Laumann**[1], **Ayushman Dash**[1]
[1]NeuralSpace, [2]Reverie Language Technologies,
{kushal,kumar,felix,ayushman}@neuralspace.ai,
adwait.deshpande@reverieinc.com

Abstract

Language models based on the Transformer architecture [1] have achieved state-of-the-art performance on a wide range of natural language processing (NLP) tasks such as text classification, question-answering, and token classification. However, this performance is usually tested and reported on high-resource languages, like English, French, Spanish, and German. Indian languages, on the other hand, are underrepresented in such benchmarks. Despite some Indian languages being included in training multilingual Transformer models, they have not been the primary focus of such work. In order to evaluate the performance on Indian languages specifically, we analyze these language models through extensive experiments on multiple downstream tasks in Hindi, Bengali, and Telugu language. Here, we compare the efficacy of fine-tuning model parameters of pre-trained models against that of training a language model from scratch. Moreover, we empirically argue against the strict dependency between the dataset size and model performance, but rather encourage task-specific model and method selection. Finally, we present effective strategies for handling the modeling of Indian languages and we release our model checkpoints for the community : https://huggingface.co/neuralspace-reverie.

1 Introduction

Natural Language Processing (NLP) has witnessed a paradigm shift from employing task-specific architectures to fine-tuning the same pre-trained language models for various downstream tasks [2, 3]. These language models are trained on large corpora of unlabelled text in an unsupervised manner. With the advent of Transformer-based architectures and various pre-training techniques in the last two years [4, 5, 6, 7], the state-of-the-art results have improved on various downstream tasks. However, much of this research has been limited to high-resource languages such as English, French, Spanish, and German [8]. There are 7000+ languages spoken around the world and yet most of the NLP systems are largely evaluated on a handful of high-resource languages. Hence, there is a dire need to work on NLP beyond resource-rich languages [9]. Our work contributes to filling this gap, as we focus on NLP for Indian languages.

India is a multilingual country with only 10 percent of its population speaking English [10]. There have been some concerted and noteworthy efforts to advance research in Indian

* Equal contribution

NeurIPS 2020 Workshop: ML Retrospectives, Surveys & Meta-Analyses (ML-RSA).

languages [11], but very little work has been done with respect to the recently proposed Transformer-based models [4, 12]. Most of the recent works that have tried using these models for their applications have preferred using a multilingual version of Bidirectional Encoder Representations from Transformers (BERT), proposed as mBERT [4], rather than training monolingual language models from scratch, i.e., without transferring any parameters from other pre-trained models. However, [8] shows that mBERT does not have high quality representations for all languages. For example, Multilingual BERT (mBERT) performs worse than non-BERT models on a number of downstream tasks for the bottom 30 percent of languages in terms of dataset-size upon which mBERT is trained [8]. Therefore, we see it as important to evaluate language models in the monolingual setting for low-resource languages. In this paper, we evaluate Transformer-based architectures for three Indian languages (Hindi, Bengali, and Telugu). To this end, our contributions are as follows:

- We train four variants of contextual monolingual language models, namely BERT, DistilBERT, RoBERTa and XLM-RoBERTa, for three Indian languages (Hindi, Bengali and Telugu), which are spoken by more than 60 percent of the Indian population [10]. These languages have different scripts based on an alphasyllabary system [13].
- We present an exhaustive analysis of these models by evaluating them on multiple NLP tasks: Question Answering (QA), Parts-of-Speech (POS) Tagging, and Text Classification. We conduct a wide range of experiments with three different setups that enable us to compare our language model variants with their multilingual counterparts and understand different factors that lead to better results on downstream tasks.
- For each setup, we evaluate models under two training settings which include fine-tuning the entire model and using the language model output as contextual embeddings. We use different layers, namely Long Short-Term Memory (LSTM) [14], BiLSTM, fully connected and Transformer, on top of these embeddings.
- We present a correlation between the available dataset size for training vs the results obtained and empirically demonstrate that using Transformers as feature extractors can lead to competitive results in several downstream tasks.
- We plan to release our language model checkpoints[2] to aid further research in NLP for Indian languages. Finally, we also intend to open-source *mergedQuAD*[3], a combination of XQuAD and MLQA datasets for Hindi, that allows training and comparison of QA models for Hindi.

Our paper is structured as follows. In the next Section 2, we place our research in context with related work in the downstream tasks we examine here. In Section 3, we provide the necessary theoretical background to the various Transformer-based models we anticipate to employ, and the subsequent Sections 4 and 5 explain our experimental setup and our results, respectively. We conclude our work with a brief discussion in Section 6.

2 Background & Related Work

In this section, we briefly discuss some relevant work in NLP for Indian languages, separately for each downstream task. We start with QA before we review existing methods for POS tagging and text classification for the three Indian languages we investigate.

2.1 Question Answering

The QA task involves extracting the answer for a given question from a corresponding passage. Traditional QA systems for Indian languages were largely based on multiple moving components, each designed for a specific task. For instance, [15] outlined a pipeline for QA systems in Hindi with separate components for query generation/processing, document retrieval, and answer extraction. The question types were limited to four and had separate answer selection algorithms for each. A similar component-based method was proposed by

[2] `https://huggingface.co/neuralspace-reverie`

[3] `https://github.com/Neural-Space/indic-transformers`

[16] for the Bengali language. While [15] only used static data for evaluation, [16] collected data from Wikipedia and annotated it for QA. As research significantly progressed in recent years, more advanced algorithms for different components were proposed in these systems. [17] used Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for the classification of a question, the software Lucene[4] for the retrieval of the document, and similarity features such as term-coverage and proximity-score for extracting the answers. Similarly, [18] used a language-independent graph-based algorithm for document retrieval and bidirectional Gated Recurrent Units (GRU) [19] for answer extraction in a multilingual setting for English and Hindi. More recently, Transformer-based models have been applied to the task of QA, but this has been restricted to fine-tuning of mBERT [4] over translated datasets [20]. A major issue for all the approaches discussed above is that they use different datasets or translate existing datasets, and hence lack a common benchmark. QA in English and other high-resource languages has become more end-to-end largely because of the SQuAD dataset [21] and a very clear definition of the task. Lately there have been efforts to make NLP more multilingual and many SQuAD-like datasets have been curated for low-resource languages [22, 23, 24].

2.2 POS Tagging

POS tagging for Indian languages has been a challenging task not only due to the lack of annotated data but also because of their rich morphology. Most of the earlier attempts [25] made to tag such languages relied heavily on morphological rules and linguistic features of a language and did not generalize to other languages. [26] tried to counter this by proposing a simple POS tagger based on a Hidden Markov Model (HMM) which achieved reasonable results. Amongst Indian languages, a lot of work has been done on tagging Hindi when compared to other low-resource languages. [27] proposed a method to transfer features from a comparatively high-resource Indian language like Hindi using parallel translation corpora which were more easily available. POS tagging for code-mixed[5] [28] Indian languages was another effort in this direction. It involved tagging code-mixed data in multiple language pairs including English-Bengali, English-Telugu, and English-Hindi, collected from various social media platforms [29, 30]. More recently, a lot of surveys have focused on evaluating contextualized embeddings for POS tagging across many languages using the Universal Dependencies [31] treebanks. For example, [32] present a comparative analysis of 54 languages on tasks like POS tagging, dependency parsing and lemmatization using Embeddings from Language Models (ELMo) [3], flair [33] and BERT embeddings. Furthermore, [34] analyse how mBERT performs by evaluating it on tasks of POS tagging and Named-Entity Recognition (NER).

2.3 Text Classification

Advancements in NLP for Indian languages mirrors the progress made for English albeit it has been slower due to the lack of annotated datasets, pre-trained word embeddings, and, more recently, pre-trained language models. Nonetheless, inltk [11] and indic-NLP [35] have curated datasets, trained embeddings and created benchmarks for classification. Indic-NLP provides pre-trained FastText [36] word embeddings in multiple Indian languages. They have also curated classification datasets for nine languages providing a common benchmark for future research. Apart from these contributions towards the classification task, we briefly discuss some other notable work. Similar to tagging code-mixed text, sentiment analysis of code-mixed social media text has also drawn attention [37], involving English-Hindi and English-Bengali datasets that saw various machine learning and deep learning approaches being applied to classification. [38] presented a comparative study of various classification techniques for Hindi. They train CNN and LSTM architectures built upon FastText embeddings. They also evaluate mBERT sentence embeddings on their translated datasets.

[4] `https://github.com/apache/lucene-solr`

[5] Code-mixing is the mixing of two or more languages or language varieties in speech.

3 Transformer Language Models

The Transformer model by [1] uses stacked encoder and decoder layers comprised of multi-headed self-attention layers without using any recurrent or convolutional modules. Despite being a departure from the then state-of-the-art RNN-based approaches to translation, the Transformer model showed significant performance improvements on the WMT 2014 translation task[6]. Inspired by this use of attention mechanism, the language models described below make use of the encoder layers from this model architecture.

3.1 BERT

BERT [4], a bidirectional Transformer model, showed significant improvements in natural language understanding tasks. A version of the model was trained on English and another on 104 languages using a masked language modeling approach. The latter version, called mBERT, was pre-trained on Wikipedia articles. The pre-trained version could be used to further fine-tune the performance on downstream tasks such as QA and token classification.

[39] observed significant improvements in performance over mBERT by training a language model from scratch on data in German from multiple sources. We expect to see similar performance improvements in low-resource languages by fine-tuning on monolingual data (Setup 2) as well as training from scratch (Setup 3).

3.2 DistilBERT

Although the BERT language model achieves state-of-the-art performance, it comes at the cost of a large model size having more than 340 million parameters [4]. This makes it harder to deploy and use it on resource-constrained devices. [40] proposed a method to reduce BERT's model size by 40% using distillation. The model, called DistilBERT and consisting of only 66 million parameters runs 60% faster and retains 97% language understanding capabilities of the original model. Since we aim to show the effectiveness of Transformer-based language models in a production setting as well, we have included DistilBERT (DistilBERT) in our experiments.

3.3 RoBERTa

[12] performed an analysis of the training procedure of BERT and showed that BERT's performance can be improved by training BERT on a larger dataset for a longer duration. This model, called RoBERTa, shows an improvement of 4-5% over BERT on natural language inference and QA tasks. Another interesting modification that the authors make is the use of a byte-level BPE (Byte Pair Encoding) tokenizer, instead of a character-level BPE tokenizer used in BERT, to have the advantage of a universal encoding scheme at the cost of a minor degradation in performance. CamemBERT [41], a RoBERTa language model trained from scratch on French language data, achieves state-of-the-art performance on downstream tasks using a relatively small web crawled dataset. We aim to verify this in our experiments by including a similar monolingual RoBERTa model trained on a web crawled Indian language dataset.

3.4 XLMRoBERTa

Using over 2 terabytes of web-crawled data, the XLM-RoBERTa model [42] achieves state-of-the-art performance on natural language inference and several downstream tasks. More interestingly, their work shows that lower resource languages such as Urdu benefit significantly through cross-lingual training at a large scale. Unlike the language-specific tokenization employed by mBERT, XLM-RoBERTa uses Sentencepiece [43] tokenization on raw text without any degradation in performance.

[6] A detailed task description and the dataset are available at `(http://www.statmt.org/wmt14/)`

4 Experimental Setup

All our experiments are run on BERT-based Transformer language models. We use the Huggingface Transformers library [44] for setting up our experiments and for downloading pre-trained models. For each language model, we use three setups:

1. **Setup A**: We directly use the pre-trained multilingual version of the models released along with research papers for BERT, DistilBERT and XLM-RoBERTa. RoBERTa was only trained on English language data and has no multilingual variant. These models form the baseline for all our experiments. The pre-trained models are downloaded from the Huggingface model hub[7].
2. **Setup B**: We use the pre-trained models from above and train them further, or fine-tune them, keeping the vocabulary unchanged. We fine-tune separately for Hindi, Bengali, and Telugu using monolingual corpora of each language. With this setup, we want to see the effect that increasing the amount of language data has on the original multilingual model. We train these models on the monolingual data for 5 epochs using approximately 300MB of data for each language.
3. **Setup C**: As observed in [39], training the German language BERT model from scratch results in a better performance than mBERT. For each of the architectures, including RoBERTa, we train Hindi, Bengali, and Telugu models on monolingual data. We do not transfer any parameters from either the pre-trained (Setup 1) or the fine-tuned models (Setup 2). The models are trained for 5 epochs on monolingual data for each of the languages. We compare this setup with previous setups to see what impact, if any, does cross-lingual training have on performance.

4.1 Language Selection

We choose to evaluate Transformer models on languages from three different regions — North, East, and South — of India. These languages also show a great variation in their linguistic features as well as their scripts. This makes them well suited for a comprehensive evaluation of language models.

1. **Hindi**: The Hindi language belongs to the Indo-Aryan family [45] and is spoken by around 322 million people in India [10] making it the most widely spoken language in the country. Most speakers of Hindi live in the Northern part of India. Hindi is a verb-final language with its verbs and nouns being inflected for gender and number. It is written in the phonetic Devanagari script with spaces being used to separate words.
2. **Bengali**: Spoken by more than 96 million people in the Eastern part of India, Bengali is the next most popularly spoken Indian language after Hindi. It also belongs to the Indo-Aryan family, but unlike Hindi it has no grammatical gender [46]. Bengali is also written in a phonetic script but is distinct from Devanagari.
3. **Telugu**: While Bengali and Hindi belong to the same family of languages, Telugu belongs to a separate family of Dravidian languages. It is spoken by 81 million people, largely in the Indian states of Telangana and Andhra Pradesh in the South. Telugu is an agglutinative language [47], meaning morphemes are combined without a change in form to produce complex words. Its nouns and verbs are inflected for both number and gender. Telugu, like Hindi and Bengali, is written using a phonetic script but with a separate orthography.

4.2 Datasets

We evaluate the performance of different model architectures on QA, POS tagging, and Text Classification tasks. While training our language models from scratch (Setup 3), we use monolingual unlabelled datasets. As described in Setup 1 and Setup 3, we train all the language models from scratch to compare the effect of pre-training. The Open Super-large

[7] `https://huggingface.co/models`

Crawled ALMAnaCH coRpus (OSCAR) dataset [48] is a filtered version of the Common-Crawl dataset and has monolingual corpora for 166 languages. Prior to training, we normalize the OSCAR dataset for Hindi, Bengali, and Telugu using the inltk library.

The Stanford Question Answering Dataset (SQuAD) [21], which is commonly used in the evaluation of Transformer language models, has all its examples in English. Inspired by the original SQuAD dataset, there are now a few multilingual datasets available which fulfil the same purpose. The TyDi QA dataset by [23] covers 11 languages with diverse linguistic features in order to evaluate the performance of language models on languages beyond English. The dataset contains 204K question-answer pairs and is divided into two sets of tasks. The primary tasks include selecting the relevant passage and the minimal answer span from the entire context based on the question. However, we use the dataset for the secondary 'Gold Passage' task where the relevant passage is directly provided. The secondary task is similar to SQuAD and allows us to readily test existing language models.

Hindi is more widely spoken than Bengali and Telugu, yet, interestingly, does not have a comprehensive QA dataset available for training. Both the MLQA [49] and the XQuAD [24] datasets contain validation and test sets for Hindi but not for training. The XQuAD dataset contains a machine-translated training dataset that has not been human-verified. We instead combine the training and the evaluation sets from XQuAD and MLQA datasets and split the training and test sets in the ratio 90:10. We refer to this dataset in our paper as *mergedQuAD*. We release the training and test split for *mergedQuAD* dataset for future development.

POS tagging occurs at the word level and focuses on the relationship between words within a sentence. We use open-source treebanks annotated by the Universal Dependencies [31] framework for Hindi and Telugu. We use the UPOS[8] tags to evaluate our models on POS Tagging. The treebank for Hindi comprises of 16 tags and the Telugu treebank is tagged using 14 such tags. Finally, for Bengali, we use the tagged sentences which are a part of the Indian corpus in nltk. There are 887 examples overall which we split manually into a training and a validation set. 29 unique tags are used in this dataset.

While QA and POS tagging look at parts of documents and sentences, text classification considers the entire document before placing it in a category. We have, therefore, included this task for completeness in our evaluation of Transformer-based language models. For Hindi, we use the BBC Hindi News Articles[9] which contains annotated news articles classified into 14 different categories. Since the dataset only has train and validation splits, we evaluate on the validation set only. We report our results on Bengali on a dataset released by Indic-NLP [35]. The examples are categorized into 2 different classes. For Telugu[10] too, we use the classification dataset provided by Indic-NLP. Each example in the dataset can be classified into any one of 3 different categories. We summarize the exact dataset splits for all the languages and tasks in Table 1.

Dataset	Classification			POS Tagging			Question Answering		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
NAME	BBC	Indic-NLP	Indic-NLP	UD	NLTK	UD	mergedQuAD	TyDi-QA	TyDi-QA
Train	3,468	11,200	19,199	13,304	665	1051	2072	2390	5563
Validation	867	1,400	2,399	1659	222	131	231	113	669
Test	-	1,400	2,399	1684	-	146	-	-	-

Table 1: Detailed break-down of all dataset-splits for the various tasks over which we evaluate our models. Each cell represents the number of examples in each split for every language. For cases where test split is missing, we used validation split for testing and validation was performed on a subset of training data.

4.3 Evaluation settings

For each setup explained above, we perform experiments under two settings:

[8] https://universaldependencies.org/u/pos/

[9] https://github.com/NirantK/hindi2vec/releases/tag/bbc-hindi-v0.1

[10] https://github.com/AI4Bharat/indicnlp_corpus

1. **Freezing**. In this setting, we use Transformer models as feature extractors. Specifically, we freeze the weights of the language model and use its output as feature-based embeddings to train different layers on top. For simplicity we refer to the language model as the base model and the trainable layers that we add on top as the head model. We experiment with multiple head types which include:
   (a) **Linear**. This simply involves passing the base model outputs to a linear layer.
   (b) **Multilinear**. Two linear layers are added on top of the base model.
   (c) **lstm**. We keep the configuration of LSTM layers added on top as flexible as possible by varying all its parameters along with other hyperparameters. As such, the head models can have multiple LSTM layers, can be bidirectional, or both.
   (d) **Transformer**. A Transformer encoder layer was added above the base model.
2. **Fine-tuning**. In this setting, the entire language model is trained along with a single linear layer on top of the model.

- We report our results for our text-classification experiments in terms of the accuracy obtained on the validation/test set.
- For POS tagging, we report F1 scores for all our experiments. This is calculated using the seqeval package[11]. Before feeding the labels and our model's prediction to the metric function, we ensure that we only consider the non-padded elements of the inputs.
- For QA, we report the F1 score on the validation set. We use the functions from the SQuAD v2.0 Evalutaion Script[12] to calculate this score.

5 Results and Analysis

5.1 Setup A

Model	Classification (Acc)			POS Tagging (F1)			Question Answering (F1)		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
mDistilBERT (frozen)	74.86	98.28	99.01	95.83	78.45	89.98	17.5	23.76	47.40
mDistilBERT (fine-tuned)	76.89	**98.58**	99.13	97.02	85.78	95.67	42.16	54.93	76.08
mBERT (frozen)	78.60	98.29	99.30	96.60	83.38	95.30	33.32	27.10	47.05
mBERT (fine-tuned)	78.97	97.79	99.38	97.51	86.45	96.20	56.96	72.79	**82.90**
XLM-RoBERTa (frozen)	**80.63**	98.36	**99.59**	97.25	87.55	95.34	19.65	33.14	53.40
XLM-RoBERTa (fine-tuned)	79.20	98.01	99.58	**97.98**	**92.60**	**97.12**	**59.70**	**74.31**	81.12

Table 2: Summary of the results from experiments based on Setup A, where we evaluate multilingual Transformer models on downstream tasks under two training settings, which are specified for each model variant. The training settings are (a) frozen: we use Transformer models as feature extractors and (b) fine-tuned: we fine-tune the entire model. The **highlighted** metrics show the highest performance across the models that we compare for every language. A similar tabular structure is used for subsequent tables in this section.

A general trend (Table 2) across all three languages is that the results improve as we move from mDistilBERT to mBERT and then from mBERT to XLM-RoBERTa. Although the Multilingual DistilBERT (mDistilBERT) for Bengali performs better than XLM-RoBERTa, the improvement is marginal. For Hindi, this trend is most evident. XLM-RoBERTa improves the validation accuracy on mDistilBERT approximately by 7%. Such an increase is less evident for Bengali and Telugu where all model variants achieve high metrics rather easily. For each task and each model variant, we further report results in two different training settings: frozen and fine-tuned. In the former setting, we find that the best performing head model out of all the different head models we experimented with, for all experiments (across all tasks and languages) is generally a multilayer

[11] https://github.com/chakki-works/seqeval

[12] https://worksheets.codalab.org/rest/bundles/0x6b567e1cf2e041ec80d7098f031c5c9e/contents/blob/

LSTM or Bi-directional Long Short-Term Memory (BiLSTM) which is shown specifically for POS tagging in [4]. Hence in the table above and the subsequent result tables, we only report results from a multilayer LSTM or BiLSTM as the head model. While the current norm in NLP literature is to fine-tune the entire language model, the classification results show that freezing gives comparable to better results than fine-tuning.

The trend of XLM-RoBERTa outperforming other variants across all languages is apparent when we compare POS tagging results. The performance gain with XLM-RoBERTa against mDistilBERT is more evident for Bengali (+8%) and Telugu (+6%). For POS tagging, the best results are achieved consistently with fine-tuning. Moreover, fine-tuned XLM-RoBERTa performs better by approximately 6% for Bengali when compared to the frozen setting.

XLM-RoBERTa gives the highest accuracy, except for Telugu(-1.78%), in the QA task as well. The pre-trained variant of XLM-RoBERTa shows a performance gain in Hindi (+2.74%) and Bengali (+1.52%) compared to the next best performing model.

5.2 Setup B

Model	Classification (Acc)			POS Tagging (F1)			Question Answering (F1)		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
mBERT (frozen)	**78.18**	**98.08**	99.30	97.40	87.97	**96.34**	15.89	28.10	43.08
mBERT (fine-tuned)	77.94	98.07	**99.63**	**97.68**	**91.36**	95.85	**30.86**	**68.42**	**58.36**

Table 3: Summary of results from the experiments based on setup B, where we augment multilingual Transformer models with monolingual data for each language. More specifically, we fine-tune mBERT with 300 MB of monolingual data.

In this setup, we augment the multilingual Transformer models by fine-tuning the language model on 300 MB of monolingual data for each language. We did not test this hypothesis on all model variants because the initial results (Table 3) of doing so with mBERT were not encouraging. Augmenting the mBERT with data from a single language (Hindi, Bengali, and Telugu in our case) does not lead to massive improvements in performance on downstream tasks, and the results are similar or at best comparable to mBERT. We believe that one of the possible reasons for this is that the dataset size with which we fine-tune the model is small and that the performances might improve with larger dataset size. However, for QA, there is a considerable drop in performance with this setup when compared to mBERT. While this drop could be task-dependent, there might be other reasons which need to be studied and analyzed in much more detail. We plan to conduct a more comprehensive analysis pertaining to this experimental setup alone in the future, where we would also include other multilingual models and vary the dataset sizes to understand these behaviours more concretely.

5.3 Setup C

Model	Classification (Acc)			POS Tagging (F1)			Question Answering (F1)		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
DistilBERT (frozen)	79.39	98.14	99.37	97.63	**92.09**	95.67	13.01	24.59	32.07
DistilBERT (fine-tuned)	**81.83**	98.19	99.5	97.93	91.05	96.14	22.03	42.73	55.74
BERT (frozen)	79.77	**98.58**	99.42	97.72	89.96	95.43	12.68	19.44	34.61
BERT (fine-tuned)	81.01	98.45	99.62	**98.08**	91.14	96.14	18.61	41.12	53.86
XLM-RoBERTa (frozen)	74.08	98.43	99.47	96.56	88.80	**96.31**	11.85	27.48	49.16
XLM-RoBERTa (fine-tuned)	78.75	98.36	**99.50**	96.73	90.61	95.60	15.93	**53.03**	**70.80**
RoBERTa (frozen)	78.22	97.08	97.86	96.13	84.28	87.15	8.82	20.95	34.61
RoBERTa (fine-tuned)	76.82	97.72	98.88	96.74	86.85	90.92	**24.74**	39.13	58.02

Table 4: Summary of results from experiments based on setup C, where we train four variants of contextual monolingual models for three languages.

In setup C, we train monolingual language models from scratch and evaluate their performance on downstream tasks. We observe (Table 4) that our model variants for BERT and DistilBERT perform better than their multilingual counterparts (setup A) across all tasks and languages. While our XLM-RoBERTa variants post competitive results for Bengali and Telugu, they still fall short of the multilingual model by a small margin. XLM-RoBERTa-Hindi performs poorly than expected on classification, POS tagging and QA under both the training settings. We believe this behaviour can be attributed to the fact that we put some constraints on our training setup. We trained all

our model variants uniformly for the same number of epochs. Essentially, we conjecture that our XLM-RoBERTa model for Hindi is under-trained and can be trained more to furnish better results on downstream tasks.

Another observation that warrants our attention is the relatively poor performance of RoBERTa models across all tasks and languages. RoBERTa uses a Byte-Level BPE tokenizer, which is known to have poorer morphological alignment with the original text compared to a simple unigram tokenizer, and that this results in a higher performance gap on SQuAD and MNLI tasks [50]. The authors of RoBERTa [12] choose the Byte-Level BPE tokenizer for higher coverage and notice no drop in performance when it comes to English. However, dealing with morphologically rich languages, as in our case, clearly seems to impact the performance.

Fine-tuning the model for the QA tasks results in improvements in Setup C as well (more than 11% improvement for Bengali and Telugu). Across all setups, there is a marked performance gain when a model is fine-tuned against the dataset compared to when only the classifier head is used. This gain is more marked in the case of QA task than it is for POS tagging or Text classification. Especially apparent from the results for the QA task is that the pre-trained models always perform significantly better than the models trained from scratch.

Model	Classification (Acc)			POS Tagging (F1)			Question Answering (F1)		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
Setup A (frozen)	80.63	98.36	99.59	97.25	87.55	95.34	33.32	33.14	53.40
Setup A (fine-tuned)	79.20	98.58	99.58	97.98	92.60	97.12	59.70	74.31	82.9
Setup C (frozen)	79.39	**98.58**	99.47	97.63	92.09	96.31	13.01	27.48	49.16
Setup C (fine-tuned)	**81.83**	98.45	99.50	98.08	91.14	96.14	24.74	53.03	70.80
IndicBERT	-	97.14	**99.67**	-	-	-	-	-	-
inltk	78.75	-	-	-	-	-	-	-	-
TyDiQA	-	-	-	-	-	-	-	**75.4**	**83.3**

Table 5: Direct comparison between the best results from setup A and setup C and related work. The empty cells denote models were either not evaluated on that particular task/language (e.g., IndicBERT on POS Tagging) or that the model was only evaluated on one specific task (e.g., TyDiQA).

Finally, we compare our best results from setup A and setup C with other relevant work in the field. The most recent and relevant work related to ours is by [51], who released the IndicBERT model, a multilingual ALBERT model [52] pre-trained on 11 Indian languages. They, however, do not report results for POS tagging and SQuAD-based QA tasks. For Hindi classification, we compare our results with inltk [11] that uses a fine-tuned ULMFiT language model. We compare our results with the baseline results for the TyDi QA Gold Passage task for Bengali and Telugu. We do not have a comparable dataset and baseline for Hindi QA tasks and we publish our results with our merged dataset. We report competitive results for Hindi and Bengali classification, both attained by our language models in setup C. Even for Telugu, the difference between our model and IndicBERT is marginal. As for the QA task, our results do not match up to the baselines of TyDi QA Gold Passage task. The baseline scores for TyDi QA were calculated by fine-tuning over the entire dataset, and not on individual languages. This strongly suggests that the QA tasks benefits a lot from cross-lingual transfer learning.

6 Conclusion

In this work, we have investigated the efficacy of state-of-the-art Transformer language models on languages other than English. We trained 4 variants of monolingual contextual language models with different sizes and capabilities for 3 Indian languages that cover more than 60% of the country's population. We evaluated our models on three downstream tasks (POS Tagging, text classification, and QA) under two distinct training settings and across three carefully planned experimental setups. By doing so, we present a highly exhaustive analysis of the performance of Transformer-based models on Indian languages, something which has been lacking in this research space. Our best models reach or improve the state-of-the-art results across multiple tasks and languages.

One of the most important aspects of our analysis is that we directly compare our trained language models with their existing multilingual counterparts. Upon comparison, our results demonstrate that while monolingual models perform better for some tasks/languages, the improvement attained, if at all, is marginal at best. Our analysis also shows that some variants of Transformer models might be better suited to your needs depending on the available resources, training-dataset size, and downstream task. Furthermore, our experiments also show that competitive results can be

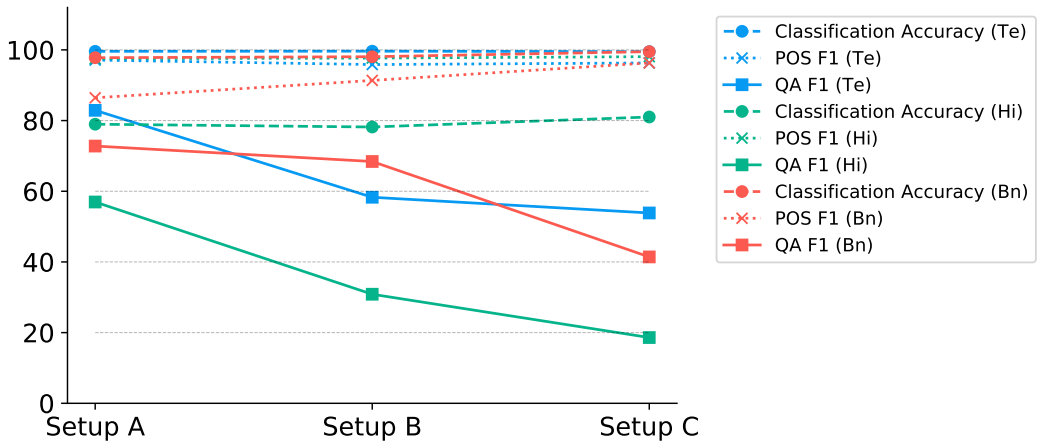


Figure 1: Relationship between the dataset size that the language model was trained on and its performance on various downstream task. The Y-axis here denotes the generic metric which is F1 score in case of POS tagging and QA and accuracy in case of text classification.

achieved by using Transformer models as feature extractors and training different layers on top (LSTM for best results).

We observed that a Byte-Level BPE (in ROBERTa) affects model performance especially in QA tasks. In our follow up work, we aim to explore the impact of tokenizers choice on Indian languages. We notice that the Telugu language model tends to perform well in the QA task despite a lower monolingual dataset size (Table 9). This encourages us to explore more on what combination of monolingual dataset size and task level dataset size is sufficient to train models with high accuracy on language inference tasks (an area where there is a lot of room for improvement).

7 Future Scope

While there is still a lot of work to do in this area, we outline a few ideas that we hope to pursue in the future. Firstly, we need to create larger corpora for under-resourced languages in order to train more powerful and efficient language models. This work shows that monolingual models trained from the data currently available perform only marginally better than multilingual models. Secondly, we also establish the need of curating annotated datasets for various downstream tasks like POS Tagging and QA in Indian languages and more importantly, make them easily accessible to the research community. At the same time, we need to create uniformly annotated datasets and benchmarks for Indian languages so that new approaches can be compared easily. [51] very recently released such a benchmark for Indian languages called the IndicGLUE, which we believe is a step in the right direction.

References

[1] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017.

[2] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *arXiv:1801.06146 [cs, stat]* (May 2018). arXiv: 1801.06146. URL: http://arxiv.org/abs/1801.06146.

[3] Matthew E. Peters et al. "Deep contextualized word representations". In: *arXiv:1802.05365 [cs]* (Mar. 2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (May 2019). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[5] Kevin Clark et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[6] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv:1906.08237 [cs]* (Jan. 2020). arXiv: 1906.08237. URL: http://arxiv.org/abs/1906.08237.

[7] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692 [cs]* (July 2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[8] Shijie Wu and Mark Dredze. "Are All Languages Created Equal in Multilingual BERT?" In: *arXiv:2005.09093 [cs]* (May 2020). arXiv: 2005.09093. URL: http://arxiv.org/abs/2005.09093.

[9] Sebastian Ruder. *Why You Should Do NLP Beyond English*. http://ruder.io/nlp-beyond-english. 2020.

[10] Office of the Registrar General & Census Commissioner India. *C-17 POPULATION BY BILINGUALISM AND TRILINGUALISM*. URL: https://www.censusindia.gov.in/2011census/C-17.html.

[11] Gaurav Arora. *iNLTK: Natural Language Toolkit for Indic Languages*. 2020. arXiv: 2009.12534 [cs.CL].

[12] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692 [cs]* (July 2019).

[13] William Bright. "A matter of typology: Alphasyllabaries and abugidas". In: *Written Language & Literacy* 2.1 (1999), pp. 45–55.

[14] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*. Mar. 2006. DOI: 10.1162/neco.1997.9.8.1735. URL: https://www.mitpressjournals.org/doix/abs/10.1162/neco.1997.9.8.1735.

[15] Shriya Sahu. "Prashnottar: A Hindi Question Answering System". In: *International Journal of Computer Science and Information Technology* 4.2 (Apr. 2012), pp. 149–158. ISSN: 09754660. DOI: 10.5121/ijcsit.2012.4213.

[16] Somnath Banerjee, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. "BFQA: A Bengali Factoid Question Answering System". In: *Text, Speech and Dialogue*. Ed. by Petr Sojka et al. Vol. 8655. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 217–224. ISBN: 978-3-319-10815-5. DOI: 10.1007/978-3-319-10816-2_27. URL: http://link.springer.com/10.1007/978-3-319-10816-2_27.

[17] Deepak Gupta et al. "MMQA: A Multi-domain Multi-lingual Question-Answering Framework for English and Hindi". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. URL: https://www.aclweb.org/anthology/L18-1440.

[18] Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. "A Deep Neural Network Framework for English Hindi Question Answering". In: *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 19.2 (Nov. 2019). ISSN: 2375-4699. DOI: 10.1145/3359988. URL: https://doi.org/10.1145/3359988.

[19] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[20] Somil Gupta and Nilesh Khade. "BERT Based Multilingual Machine Comprehension in English and Hindi". In: (June 2020).

[21] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *arXiv:1606.05250 [cs]* (Oct. 2016).

[22] Patrick Lewis et al. "MLQA: Evaluating Cross-lingual Extractive Question Answering". In: *arXiv:1910.07475 [cs]* (May 2020). arXiv: 1910.07475. URL: http://arxiv.org/abs/1910.07475.

[23] Jonathan H. Clark et al. "TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages". In: *arXiv:2003.05002 [cs]* (Mar. 2020).

[24] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. "On the Cross-lingual Transferability of Monolingual Representations". In: *arXiv:1910.11856 [cs]* (May 2020). arXiv: 1910.11856. URL: http://arxiv.org/abs/1910.11856.

[25] Jan Haji, Barbora Hladka, and Petr Pajas. "The Prague Dependency Treebank: Annotation Structure and Support." In: Jan. 2001, pp. 105–114.

[26] Manish Shrivastava and P. Bhattacharyya. "Hindi POS Tagger Using Naive Stemming : Harnessing Morphological Information Without Extensive Linguistic Knowledge". In: 2008.

[27] Pruthwik Mishra, Vandan Mujadia, and Dipti Sharma. "POS Tagging For Resource Poor Indian Languages Through Feature Projection". In: Feb. 2018.

[28] Amitava Das. URL: https://amitavadas.com/Code-Mixing.html.

[29] Prakash B. Pimpale and Raj Nath Patel. "Experiments with POS Tagging Code-mixed Indian Social Media Text". In: *arXiv:1610.09799 [cs]* (Oct. 2016). arXiv: 1610.09799. URL: http://arxiv.org/abs/1610.09799.

[30] Sree Harsha Ramesh and Raveena R. Kumar. "A POS Tagger for Code Mixed Indian Social Media Text - ICON-2016 NLP Tools Contest Entry from Surukam". In: *arXiv:1701.00066 [cs]* (Dec. 2016). arXiv: 1701.00066. URL: http://arxiv.org/abs/1701.00066.

[31] Joakim Nivre et al. *Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection*. 2020. arXiv: 2004.10643 [cs.CL].

[32] Milan Straka, Jana Straková, and Jan Haji. "Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing". In: *arXiv:1908.07448 [cs]* (Aug. 2019). arXiv: 1908.07448. URL: http://arxiv.org/abs/1908.07448.

[33] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139.

[34] Telmo Pires, Eva Schlinger, and Dan Garrette. "How Multilingual is Multilingual BERT?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019, pp. 4996–5001. DOI: 10.18653/v1/P19-1493. URL: https://www.aclweb.org/anthology/P19-1493.

[35] Anoop Kunchukuttan et al. "AI4Bharat-IndicNLP Corpus: Monolingual Corpora and Word Embeddings for Indic Languages". In: *arXiv preprint arXiv:2005.00085* (2020).

[36] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[37] Braja Gopal Patra, Dipankar Das, and Amitava Das. "Sentiment Analysis of Code-Mixed Indian Languages: An Overview of SAIL_CodeMixed Shared Task @ICON-2017". In: *arXiv:1803.06745 [cs]* (Mar. 2018). arXiv: 1803.06745. URL: http://arxiv.org/abs/1803.06745.

[38] Ramchandra Joshi, Purvi Goel, and Raviraj Joshi. "Deep Learning for Hindi Text Classification: A Comparison". In: *arXiv:2001.10340 [cs, stat]* 11886 (2020). arXiv: 2001.10340, pp. 94–101. DOI: 10.1007/978-3-030-44689-5_9.

[39] Chan, Branden et al. *Deepset - Open Sourcing German BERT*. https://deepset.ai/german-bert.

[40] Victor Sanh et al. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter". In: *arXiv:1910.01108 [cs]* (Feb. 2020).

[41] Louis Martin et al. "CamemBERT: A Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020).

[42] Alexis Conneau et al. "Unsupervised Cross-Lingual Representation Learning at Scale". In: *arXiv:1911.02116 [cs]* (Apr. 2020).

[43] Taku Kudo and John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL].

[44] Thomas Wolf et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* (2019), arXiv–1910.

[45] Omkar N Koul. *Modern Hindi Grammar*. Dunwoody Press Springfield, USA, 2008.

[46] Hanne-Ruth Thompson. *Bengali*. Vol. 18. John Benjamins Publishing, 2012.

[47] Bhadriraju Krishnamurti. *The dravidian languages*. Cambridge University Press, 2003.

[48] Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. "A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 1703–1714. URL: https://www.aclweb.org/anthology/2020.acl-main.156.

[49] Patrick Lewis et al. "MLQA: Evaluating Cross-lingual Extractive Question Answering". In: *arXiv preprint arXiv:1910.07475* (2019).

[50] Kaj Bostrom and Greg Durrett. *Byte Pair Encoding is Suboptimal for Language Model Pretraining*. 2020. arXiv: 2004.03720 [cs.CL].

[51] Divyanshu Kakwani et al. "IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages". In: *Findings of EMNLP*. 2020.

[52] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[53] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

8 Appendix

A Training details and metrics for Language Modeling

The techniques and setups that we have used for our different language models vary depending on the setup. In our future work we plan to cover in the missing combination of dataset sizes and model architectures for completeness.

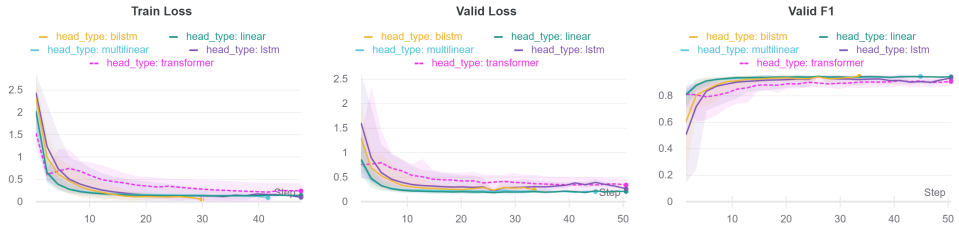
Model	Setup B			Setup C		
	Hindi	Bengali	Telugu	Hindi	Bengali	Telugu
DistilBERT	300 MB	300 MB	300 M	10 GB	6 GB	2 GB
BERT	300 MB	300 MB	300 M	3 GB	3.1 GB	1.6 GB
RoBERTa	-	-	-	10 GB	6 GB	2 GB
XLM-RoBERTa	300 MB	300 MB	300 MB	3 GB	3.1 GB	1.6 GB

Table 6: A comparison of the dataset sizes used for training different language models

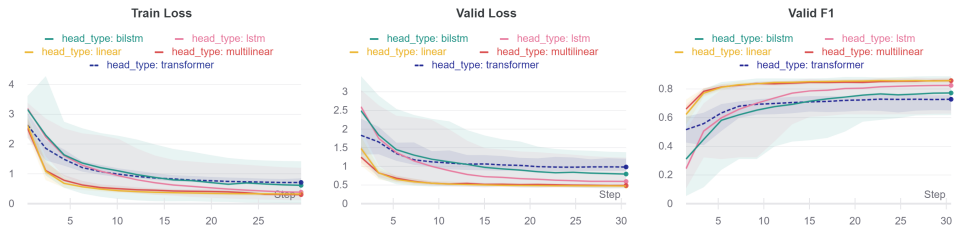
B Training metrics for pos Tagging

In this section we show some of the plots for our experiments for each language. These plots clearly show the different head layers that we train as part of our training setting described in the paper. We used Weights and Biases [53] to run extensive hyperparameter search for our models. The plots in this section are for DistilBERT in setup C.

B.1 Telugu



B.2 Bengali

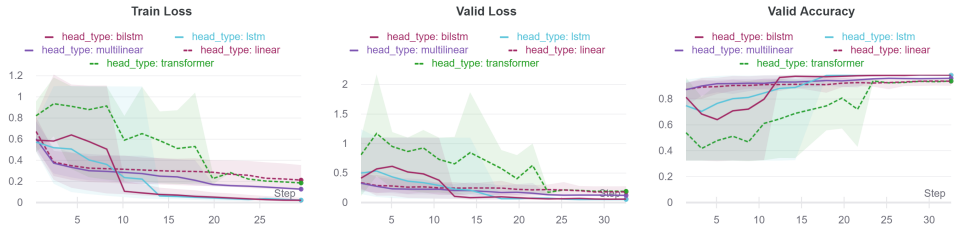


B.3 Hindi

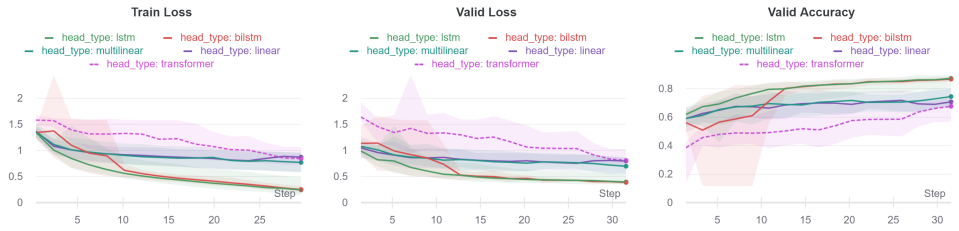
C Training metrics for Text Classification

This section shows similar graphs for text classification. Here we have taken plots for mDistilBERT from setup A to demonstrate our experimental setups.

C.1 Telugu



C.2 Bengali



C.3 Hindi

D Results for Mask Filling

We show the results of mask filling for each of our language models. We used human translators to translate "It is important to be able to communicate with them". The word corresponding to "communicate" was masked for each language.

Model	Mask Fill Results
Human Translations	उनके साथ संवाद करने में सक्षम होना महत्वपूर्ण है
BERT Base Cased (Setup A)	उनक साथ न करन म सकषम होना महतवपरण ह
BERT Base Uncased (Setup A)	उनक साथ काम करन म सकषम होना महतवपरण ह
BERT Scratch (Setup C)	उनक साथ काम करन म सकषम होना महतवपरण ह
BERT Finetuned (Setup B)	उनक साथ काम करन म सकषम होना महतवपरण ह
DistilBERT Scratch (Setup C)	उनक साथ काम करन म सकषम होना महतवपरण ह
RoBERTa Scratch (Setup C)	उनके साथ मदद करने में सक्षम होना महत्वपूर्ण है
XLM-RoBERTa Base (Setup A)	उनके साथ काम करने में सक्षम होना महत्वपूर्ण है
XLM-RoBERTa Scratch (Setup C)	उनके साथ काम करने में सक्षम होना महत्वपूर्ण है

Table 7: Mask filling results for Hindi

Model	Mask Fill Results
Human Translations	এগুলোর মাধ্যমে যোগাযোগ করতে সক্ষম হওয়া গুরুত্বপূর্ণ
BERT Base Cased (Setup A)	এগলোর মাধযমেদ করতে সকষম হওযা গরতবপরণ
BERT Base Uncased (Setup A)	এগলোর মাধযমে কাজ করতে সকষম হওযা গরতবপরণ
BERT Scratch (Setup C)	এগলোর মাধযমে কাজ করতে সকষম হওযা গরতবপরণ
BERT Finetuned (Setup B)	এগলোর মাধযমে কাজ করতে সকষম হওযা গরতবপরণ
DistilBERT Scratch (Setup C)	এগলোর মাধযমে অরজন করতে সকষম হওযা গরতবপরণ
RoBERTa Scratch (Setup C)	এগুলোর মাধ্যমে সহজ করতে সক্ষম হওয়া গুরুত্বপূর্ণ
XLM-RoBERTa Base (Setup A)	এগুলোর মাধ্যমে কাজ করতে সক্ষম হওয়া গুরুত্বপূর্ণ
XLM-RoBERTa Scratch (Setup C)	এগুলোর মাধ্যমে কাজ করতে সক্ষম হওয়া গুরুত্বপূর্ণ

Table 8: Mask filling results for Bengali

Model	Mask Fill Results
Human Translations	వారితో కమ్యూనికేట్ చేయడం అనేది ఎంతో ముఖ్యం
BERT Base Cased (Setup A)	వరతల చయడం అనద ఎంత ముఖయం
BERT Base Uncased (Setup A)	వరతపటు చయడం అనద ఎంత ముఖయం
BERT Scratch (Setup C)	వరత సనహం చయడం అనద ఎంత ముఖయం
BERT Finetuned (Setup B)	వరత పసట చయడం అనద ఎంత ముఖయం
DistilBERT Scratch (Setup C)	వరత సనహం చయడం అనద ఎంత ముఖయం
RoBERTa Scratch (Setup C)	వారితోరక చేయడం అనేది ఎంతో ముఖ్యం
XLM-RoBERTa Base (Setup A)	వారితో మంచి చేయడం అనేది ఎంతో ముఖ్యం
XLM-RoBERTa Scratch (Setup C)	వారితో సెక్స్ చేయడం అనేది ఎంతో ముఖ్యం

Table 9: Mask filling results for Telugu