Improving Access to Medical Information for Multilingual Patients using Pipelined Ensemble Average based Machine Translation

DEBAJYOTY BANIK, SR University, India
RAHUL PAUL, Kalinga Institute of Industrial Technology, India
RAJKUMAR SINGH RATHORE, Cardiff School of Technologies, Cardiff Metropolitan University, United Kingdom
RUTVIJ H. JHAVERI*, Pandit Deendayal Energy University, India

Machine translation has shown potential in improving access to medical information and healthcare services for multilingual patients. This research aims to enhance machine translation accuracy in the medical field, specifically for translating from Hindi to English. The study introduces a new approach that dynamically allocates decoding parameters using regression models, overcoming the limitations of fixed parameters in the decoder. A comprehensive dataset is created to address limited data availability, enabling regression models to predict optimal pruning parameters. The main motivation for the study is the introduction of a regression method for optimizing pruning parameters, which is a novel approach in this context. The proposed approach outperforms existing methods, achieving improved translation accuracy. Standard metrics such as the BLEU score are used to evaluate translations. Ensemble average and pipeline approaches further enhance performance. The improved performance of the proposed models can be attributed to the ensemble of diverse models (Extra Trees, LightGBM, XGBoost, and Random Forest) that employ various techniques to reduce overfitting, enhance prediction accuracy, and improve translation by correcting prediction errors. The study contributes to facilitating the translation and sharing of medical literature, promoting collaboration and knowledge exchange across languages. The research demonstrates the effectiveness of the regression method for optimizing pruning parameters in machine translation, leading to improved translation accuracy in the medical field. The proposed models offer promising results, paving the way for enhanced machine translation systems and promoting collaboration and knowledge exchange in the medical domain. The source code is available at https://huggingface.co/debajyoty/statistical-regression-Based-MT/tree/main/Statistical-Regression-SMT.

CCS Concepts: • **Natural Language Processing → Machine Translation**;

Additional Key Words and Phrases: Statistical Machine Translation, Decoder, Beam Search Algorithm, Histogram Pruning, Threshold Pruning, Machine Learning, RMSE, Regression;

ACM Reference Format:
Debajyoty Banik, Rahul Paul, Rajkumar Singh Rathore, and Rutvij H. Jhaveri*. 2023. Improving Access to Medical Information for Multilingual Patients using Pipelined Ensemble Average based Machine Translation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 1, 1 (August 2023), 18 pages. https://doi.org/0000001.0000001

Authors' addresses: Debajyoty Banik, School Computer Science and Artificial Intelligence, SR University, Warangal, India, debajyoty.banik@gmail.com; Rahul Paul, School of Computer Engineering, Kalinga Institute of Industrial Technology, Bhubaneswar, India, 2164002@kiit.ac.in; Rajkumar Singh Rathore, Department of Computer Science, Cardiff School of Technologies, Cardiff Metropolitan University, Llandaff Campus, Western Avenue Cardiff, United Kingdom, rsrathore@cardiffmet.ac.uk; Rutvij H. Jhaveri*, Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Energy University, India, correspondence:rutvij.jhaveri@sot.pdpu.ac.in.

2375-4699/2023/8-ART $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION AND PROBLEM DESCRIPTION

Over the past few years, the medical field has seen a lot of potential in using machine translation, especially in enhancing access to medical information and healthcare services for patients who do not speak the same language [36]. By using machine translation, medical information like patient data, medical reports, and guidance on medications and procedures can be converted into the preferred language of the patient [15]. This can significantly improve communication between healthcare providers and patients, leading to reduced chances of miscommunication and an improvement in the quality of care. Machine translation can be employed to translate medical research articles into different languages, promoting global cooperation and knowledge exchange. The possible benefits of using machine translation in improving access to medical information and services for multilingual patients are substantial and hold the potential to minimize disparities in healthcare and enhance patient outcomes. The structure of medical texts may differ based on each author's distinctive writing style. Recent studies on machine translation have produced encouraging results when translating between human languages. The phrase-based model used by statistical machine translation (SMT) outperforms word-based models [25], which has caught the interest of the modern NLP community. Phrase-based SMT operates by diving a source string into a sequence of phrases (S), that are converted into a series of phrases in the target language (T). It uses the Noisy channel model, Equation 1, where the output is generated as hypotheses or partial translations.

$$t_{best} \approx arg_e max[P(S|T).P(T)] \tag{1}$$

In Equation 1, s and t are the phrase strings for the source and target languages, respectively. $P(s|t)$ is the bilingual translation model or translation model (TM) and $P(t)$ is the target-side language model (LM).

From Equation 1, the translation model $P(s|t)$ is further expressed as:

$$P\left(s^I|t^I\right) = \prod_{i=1}^{I} \phi(s^i|t^i).d(start_i - end_{i-1} - 1) \tag{2}$$

$\phi(s^i|t^i)$ is the probability of source phrase s^i being translated to target phrase t^i, called phrase translation probability. $d(start_i - end_{i-1} - 1)$ is the distortion probability. It is the distance between words in texts in source and target languages. The language model calculates the likelihood that a given phrase would appear in the target text.

In SMT, decoding is a scoring process that searches for the most accurate translation among all the possible translations or hypotheses. To find accurate translations within a reasonable time frame, the beam search technique [23] is employed to reduce the search space, as scoring all hypotheses would require more time. Since the decoding problem for SMT is NP-complete [22], a heuristic search method like the beam search algorithm is utilized to identify the best translation. This algorithm prunes out weak hypotheses, thereby limiting the search space that contains numerous hypotheses and optimizing the decoder. Threshold pruning and histogram pruning methods are used in the conventional decoding implementation. The hypotheses are organized into stacks. The process of threshold pruning involves eliminating hypotheses from the stack if their score is less than a specific value α (known as the beam threshold) multiplied by the score of the best hypotheses. On the other hand, histogram pruning involves maintaining a fixed number of hypotheses (n) in the stack, and discarding the rest.

Accuracy of translation and decoding time is significantly influenced by beam threshold and stack size [5]. The stack size and beam threshold parameters are constant for a standard implementation. The issue with this method is that different texts' lengths and grammatical structures vary. For example, some texts contain longer and simpler sentences, while others have short and complicated ones. Therefore, the decoding parameters' values need to vary depending on the input text. Machine learning has been applied as regression to predict the optimum values for n and α. For the first time, machine learning-based decoding was introduced in [5]. CN2 unordered algorithm (classification) was used to allocate the decoding parameters dynamically.

The drawback of the classification approach is that the decoding parameters are discrete in nature and correlate to a specific stack size or beam threshold value. It leads to bad resource utilization. Using continuous values for the decoding parameters would let the decoder choose the best translation from more hypotheses space. The classification model used was not advanced and the training was done on a small dataset. Implementing regression methods could address this issue. Two approaches are made to create a large dataset to train the regression model. Using the created dataset, 11 different regression models were trained, out of which four models were selected based on RMSE. The chosen models are Light gradient boosting, XGBoost, Random Forest, and Extra Trees. XGBoost and Random Forest resulted in better accuracy. For further improvement in accuracy, these 4 models are implemented into ensemble average and pipeline approaches.

1.1 Bridging Language Barriers in Healthcare through Machine Translation

Machine translation empowers patients to overcome language barriers and gain access to medical documents, providing accurate translations in their desired language. By utilizing online platforms or applications that support machine translation, patients can effortlessly upload various documents, including test results and prescriptions. The incorporation of machine learning in translation processes further enhances the accuracy and quality of translations. Through the analysis of multilingual data, and continuous optimization of output, machine learning continually refines the translation capabilities. As a result, patients can rely on improved machine learning-powered translation to bridge linguistic gaps effectively and comprehend crucial medical information, empowering them in their healthcare journey.

1.2 Related Work

The studies in [36] explored the effectiveness of neural-based machine translation in comparison to statistical machine translation for medical data, specifically for Polish-English translation. It also states that machine translation systems, including neural-based ones, are not perfect and can still make errors in translation. SMT has improved greatly since IBM made it in the late 1980s and early 1990s ([10, 12]), especially with the addition of phrase-based translation. ([30], [29], [26]).P. Koehn in [18] designed Mosesdecoder[1], which is a publicly available set of tools for machine translation. It has improved the translation accuracy and speed compared to the prior stack-based decoding method using new decoders [18], [16]. Additional enhancements were made to language models, which are fast and effective [17], [37]. A significant decrease in memory use and initial loading time was seen after the SMT decoder included a phrase table that may be loaded whenever necessary [38]. The introduction of weights within the phrase table in [2] has shown favorable outcomes regarding translation quality. The on-disk phrase database was later compressed, and the lexical reordering option was added [20]. In a recent study, hardware modules have been used to transliterate words using rules [3]. Researchers have done studies on improving translation quality. It has been seen that the source-side syntactic reordering approach produces better translation

[1]https://github.com/moses-smt/mosesdecoder

for hierarchical MT [32] and RBMT [9]. Despite being a recent trend, neural machine translation (NMT) is not an effective solution for low-resource languages because a significant quantity of information is needed to work effectively [34]. P. Koehn in [24] investigated a few issues regarding NMT, namely, domain mismatch, amount of training data, rare words, long sentences, word alignment, and beam search. The research in [19] focuses on solving the domain shift issue in neural machine translation by introducing an unsupervised adaptation method that adjusts pre-trained out-of-domain NMT models with a pseudo-in-domain corpus to increase their responsiveness to domain-specific words and phrases. The proposed method has limitations, including assumptions about in-domain lexicons, reliance on pre-trained out-of-domain NMT models, and limited experiments. Researchers have developed hybrid models combining neural-based models and statistical-based models to increase the quality of translation [7, 8]. So, improving the SMT model would improve the overall translation quality for hybrid models. For evaluating the translation, the most popular metric in BLEU scores [31]. BLEU is a method based on n-grams, which evaluates machine translation output by comparing exact matches of n-grams with reference translations. However, better evaluation can be done using the Wuplebleu metric [6]. Neural Network models were also considered for NLP research like back-propagation neural networks for offline signature authentication [4]. The studies in [36] explored the effectiveness of neural-based machine translation in comparison to statistical machine translation for medical data, specifically for Polish-English translation. It also states that machine translation systems, including neural-based ones, are not perfect and can still make errors in translation. SMT has improved greatly since IBM made it in the late 1980s and early 1990s ([10, 12]), especially with the addition of phrase-based translation. ([30], [29], [26]).P. Koehn in [18] designed Mosesdecoder[2], which is a publicly available set of tools for machine translation. It has improved the translation accuracy and speed compared to the prior stack-based decoding method using new decoders [18], [16]. Additional enhancements were made to language models, which are fast and effective [17], [37]. A significant decrease in memory use and initial loading time was seen after the SMT decoder included a phrase table that may be loaded whenever necessary [38]. The introduction of weights within the phrase table in [2] has shown favorable outcomes regarding translation quality. The on-disk phrase database was later compressed, and the lexical reordering option was added [20]. In a recent study, hardware modules have been used to transliterate words using rules [3]. Researchers have done studies on improving translation quality. It has been seen that the source-side syntactic reordering approach produces better translation for hierarchical MT [32] and RBMT [9]. Despite being a recent trend, neural machine translation (NMT) is not an effective solution for low-resource languages because a significant quantity of information is needed to work effectively []. P. Koehn in [24] investigated a few issues regarding NMT, namely, domain mismatch, amount of training data, rare words, long sentences, word alignment, and beam search. The research in [19] focuses on solving the domain shift issue in neural machine translation by introducing an unsupervised adaptation method that adjusts pre-trained out-of-domain NMT models with a pseudo-in-domain corpus to increase their responsiveness to domain-specific words and phrases. The proposed method has limitations, including assumptions about in-domain lexicons, reliance on pre-trained out-of-domain NMT models, and limited experiments. Researchers have developed hybrid models combining neural-based models and statistical-based models to increase the quality of translation [7], [8]. So, improving the SMT model would improve the overall translation quality for hybrid models [33]. For evaluating the translation, the most popular metric in BLEU scores [31]. BLEU is a method based on n-grams, which evaluates machine translation output by comparing exact matches of n-grams with reference translations. However, better evaluation can be done using the Wuplebleu metric [6]. Neural

[2]https://github.com/moses-smt/mosesdecoder

Network models were also considered for NLP research like back-propagation neural networks for offline signature authentication [4].

1.3 Research Gap

The research gap in machine translation in the medical field is the need for developing models that can accurately translate specialized medical terminology and jargon, as well as understand the context-specific meanings of medical terms and phrases. The primary disadvantage of the moses-decoder for machine translation is that the decoding parameters are fixed and constant regardless of the input. In [5], machine learning was used to tune the decoding parameters for machine translation. They used a classification model that takes different structural features of an input text and predicts the classes containing the values for n and α. But the classification model is not advanced. The CN2 unordered algorithm was used due to a lack of data. Since they had a very small amount of training data, it was not sufficient for generalization. The output of the classification models is discrete in nature which corresponds to a particular value or class for n or α. It led to the loss of some important values for accuracy.

1.4 Novelty and Contribution

This section presents the study's novel method for improving machine translation in the medical field. Firstly, we have proposed a new method for determining the optimal pruning parameters using regression models. The main motivation behind this research is the pioneering exploration of utilizing regression in machine translation, which has never been done before. Secondly, the ensemble average and pipeline approaches have been considered to improve the proposed regression methods. To support our regression models, we have also prepared a comprehensive dataset for predicting the decoding parameters, addressing the issue of data scarcity. Developing machine translation systems that utilize regression models can improve the precision and dependability of medical translations, resulting in improved access to healthcare services and medical information for patients who speak diverse languages.

1.4.1 Regression Models. By training regression models, the decoding parameters can be dynamically allocated considering the training features in the dataset. The output of regression models is continuous in nature, which provides the decoder with the flexibility to select the optimal translation from a wider range of hypotheses, leading to improved translation accuracy. To train our regression models, we utilized the structural features of the input text file as the training features and set the beam threshold and stack size as the target features. This approach allows us to accurately predict the optimal decoding parameters for each input text file, leading to better translation results.

1.4.2 Improvement in Regression Models. To further improve the accuracy of our approach, we employed two proposed models: the Ensemble Average (proposed Model 1) and the Pipeline approach (proposed Model 2). The Ensemble Average model calculates the average of the predicted values from the selected best-performing regression models. This approach leverages the strengths of multiple models to provide a more accurate prediction. The Pipeline approach applies the regression models in sequential form, with each model stacked one after another. This approach allows the overall error from the pipeline to be adjusted with respect to the individual model, leading to a more accurate prediction. The output predicted from the pipelined model performs better than the individual models alone.

1.4.3 Dataset. To train our machine learning models, we first created a dataset using two strategies: the odd-even strategy and the 3-file-combination strategy. This resulted in the creation of

34,220 new text files, each of which was decoded using 36 different combinations of decoding parameters. The parameters that produced the maximum BLEU score [31] in the shortest amount of time were selected for each file, and each file contributed one row to the dataset. The structural features of the text files, such as word count, number of lines, percentage of commas, percentage of stopwords, average word per line, and percentage of long lines, were used as training features for the regression models. The target features for the models were the stack size and beam threshold.

2 PROPOSED METHODOLOGY

The state-of-the-art technique for machine learning-based statistical machine translation, as described in [5], used a classification method to predict and dynamically allocate the decoding parameters. For Hindi-to-English machine translation, the input to the decoder is a Hindi text file. The pruning parameters, namely the stack size and beam threshold, are predicted by the regression models and then used as input to the decoder for the purpose of pruning the translation hypotheses. After implementing the pruning parameters, the decoder produces translations for the input text file, and the quality of the translation is assessed using standard metrics like the BLEU score. As the proposed regression model approach for finding optimal decoding parameters for pruning in machine translation is novel, there is currently no dataset available for this task. To overcome this issue, we created a dataset by decoding each Hindi file using various n and α values. The labels for the dataset were assigned based on the highest BLEU score achieved in the least amount of decoding time.

2.1 Regression Method for pruning

The advantage of using a regression model is that it produces continuous outputs, unlike the classification models that generate discrete values. The continuous output from the regression model offers a broader spectrum of values for the decoding parameters, which enables the decoder to select the best translations from a larger set of hypotheses. This, in turn, increases the chances of making better predictions. Eleven different regression models were trained in this study, including Linear Regression, Lasso Regression, Ridge Regression, Elastic Net, Support Vector Regressor, Random Forest Regressor, Decision Tree Regressor, Extra Trees, Gradient Boosting Regressor, Extreme Gradient Boosting (XGBoost), and Light Gradient Boosting (LightGBM). The performance of these models was evaluated based on their root mean square error (RMSE), as shown in Equation 3. Out of the eleven models, the four best-performing models were selected: Light Gradient Boosting [21], Extreme Gradient Boosting [13], Random Forest [28], and Extra Trees Regressor. The predictions generated by these models were then passed to the decoder for Hindi-to-English machine translation.

To further improve the regression models, two machine-learning models have been proposed: ensemble average and pipeline. These models were built by combining the four best-performing regression models, which were selected based on RMSE.

$$\sqrt{\frac{\sum_{n=1}^{N}(y_{pred}-y_{true})^2}{N}}; N = Number\ of\ instances \tag{3}$$

Equation 3 is the formula for root mean square error (RMSE), where, p_{true} is the actual output and y_{pred} is the predicted output from a machine learning model.

2.2 Improving Regression Models

To enhance the effectiveness of the regression models even further, two additional machine learning models were proposed: the ensemble average and pipeline models. The ensemble average

model takes the average of the predictions from the four best-performing regression models to generate the final output, as shown in Equation 4. The pseudocode for the ensemble average is shown in 1. The algorithm takes as input the training features and target features. It then uses four different machine learning models (*model_1 to model_4*) to generate predictions for the target features. The output of each model is stored in a variable *pred_n*. Finally, it calculates the ensemble average of the predictions by taking the sum of all predictions and dividing it by the total count of models used (N=4 in this case). This results in a single prediction output.

$$Ensemble\ Average = \frac{\sum_{i=1}^{N} Prediction_i}{N}; N = Number\ of\ models \tag{4}$$

In Equation 4, $Prediction_i$ denotes the prediction from i^{th} model, where $i = 1, 2...N$; $N = 4$.

ALGORITHM 1: Ensemble Average

Input ← (Training Features, Target Features)
pred_1 ← Model_1(Input)
pred_2 ← Model_2(Input)
pred_3 ← Model_3(Input)
pred_4 ← Model_4(Input)
output ← $\frac{\sum_{n=1}^{N} \text{pred\_}n}{N}$

$$Pipeline\ Model = Model1 \rightarrow Model2 \rightarrow Model3 \rightarrow Model4 \tag{5}$$

In the pipeline approach (proposed model 2), the four regression models are implemented in a sequential manner, i.e. one after another, shown in Equation 5. Figure 1 shows the diagrammatic representation of a pipeline strategy. In this approach, the pipeline consists of a base model and additional models, where, the knowledge gained by one model is propagated throughout the pipeline. The output of each model acts as a feature vector and a new model is trained to make the final prediction. The errors in the base model are revised further by training the succeeding models, resulting in a more accurate output than that of individual models. The benefit of utilizing this method is that the final model can learn to combine the predictions of the individual models in a way that maximizes performance. The following model in the pipeline employs the output from the previous model as its input, and the result from the final model is taken as the ultimate output of the pipeline. For example, in Pipeline Model 1, $Model_1 = Light\ Gradient\ Boosting$, $Model_2 = XGBoost$, $Model_3 = Random\ Forest$ and $Model_4 = Extra\ Trees$ are in sequence (Equation 5). The prediction of the pipeline's first model is scaled by a factor X after training, and the scaled prediction is then fed into the pipeline's second model along with the training features, and this process is repeated two more times. The scaled prediction helps in integrating the regression models by passing the information from one model to the next one as a weighted feature. The prediction of the fourth model is taken as the final output of the pipeline. After some experiments, the value of X was determined to be 1000 since it resulted in the least RMSE scores for the pipeline approach. There are 24 pipeline models, considering all the combinations with the four selected regression models. The pseudocode for the Pipeline Model is shown in Algorithm 2.

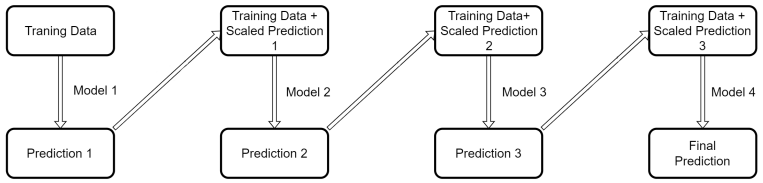

Fig. 1. Flowchart for the pipeline approach

ALGORITHM 2: Pipeline Model

```
Input ← (Training Features, Target Features)
pred_1 ← Model_1(Input)
pred_2 ← Model_2(Input, scaled(pred_1))
pred_3 ← Model_3(Input, scaled(pred_2))
pred_4 ← Model_4(Input, scaled(pred_3))
output ← pred_4
```

3 DATASET AUGMENTATION

3.1 Dataset

The study conducted in [5] utilized HindEnCorp [11] as the dataset for training and testing their machine translation system. HindEnCorp is a parallel corpus that comprises more than 274k sentences in Hindi and English languages. The corpus was gathered from diverse internet sources and has been extensively employed in natural language processing for tasks related to machine translation.

3.2 Dataset Augmentation Procedure

In [5], the HindEnCorp [11] corpus was split into three parts for training and testing the machine translation system. The first 1001 sentences were used as the tuning set, and the next 50001 sentences were split into 59 files of varying sizes from 200 to 2000 sentences to create a sizeable testing dataset. The Mosesdecoder toolkit was trained using the remaining section of the corpus. To further increase the testing dataset size, two strategies were employed for creating more files from the 59 files, for both Hindi and English files. The strategies are:

(1) Even Odd Combinations: From the initial 59 files, 2 files are picked for each language, and the sentences with an even index from one file and the sentences with an odd index from the other file are combined. Using this method, 1711 files are generated for each language.
(2) Three File Combinations: From the initial 59 files, 3 files are picked for each language, and the sentences with the n^{th}, $(n+1)^{th}$, $(n+2)^{th}$ index from first, second and third files are combined. Using this method, 32509 files are generated for each language.

Details of the files created using the two strategies are provided in Table 1. It presents the specifics of the files created using the Even Odd Combinations and Three File Combinations strategies. The table summarizes the details of the files produced as well as, the overall number of files, the count of tokens, and the count of sentences in each category

Table 1. Details of the generated dataset with respect to the various strategies.

Method	Number of files	Number of tokens	Number of sentences
files from corpus	59	727867	50121
odd even	1711	21075940	1449944
3 file combination	32509	599956197	41398908

Table 2. Snapshot of the dataset created using the odd-even and three-file-combination strategies with their best-performing parameters that resulted in the highest BLEU with the lowest decoding time during decoding for Hindi to English Machine translation. (Files with the names like $f.oddeven.hi. < file\_num > . < file\_num >$ are the odd-even files and files with the names like $f.3combi.hi. < file\_num > . < file\_num > . < file\_num >$ are 3-file combination files, AWL: Average Word per Line)

Name	Number of lines	Comma %	Word Count	Stopword %	AWL
f.oddeven.hi.1.2	869	0.509	13464	35.806	15.493
f.oddeven.hi.1.3	902	0.514	13091	36.053	14.513
f.oddeven.hi.1.4	849	0.522	12255	36.294	14.434
f.oddeven.hi.3.52	925	0.559	13495	36.519	14.589
f.oddeven.hi.3.53	1099	0.568	16206	35.432	14.746
f.oddeven.hi.3.54	1033	0.522	14542	36.034	14.077
f.3combi.hi.1.2.10	1285	0.488	19892	36.514	15.480
f.3combi.hi.1.2.11	1037	0.489	15920	35.922	15.351
f.3combi.hi.1.2.12	1328	0.468	20074	35.899	15.115
f.3combi.hi.14.49.53	1357	0.457	19770	36.437	14.568
f.3combi.hi.15.21.36	1143	0.451	16386	36.485	14.335
Name	**Long Line %**	**Stack Size**	**Beam Threshold**	**BLEU**	**Time (secs.)**
f.oddeven.hi.1.2	31.875	300	0.500	9.40	212
f.oddeven.hi.1.3	29.490	250	0.249	9.01	170
f.oddeven.hi.1.4	28.739	400	0.249	9.54	247
f.oddeven.hi.3.52	30.270	400	0.249	9.70	206
f.oddeven.hi.3.53	29.299	500	0.050	9.78	308
f.oddeven.hi.3.54	26.621	500	0.500	9.81	275
f.3combi.hi.1.2.10	31.439	500	0.500	9.54	502
f.3combi.hi.1.2.11	31.147	200	0.050	9.62	148
f.3combi.hi.1.2.12	30.045	400	0.050	10.38	363
f.3combi.hi.14.49.53	30.140	450	0.500	9.70	462
f.3combi.hi.15.21.36	26.946	400	0.050	9.20	283

3.3 Prepared Dataset

Rather than relying on traditional features like bag-of-words or word-to-vector, we focused on the structure and complexity of the translations when training our regression models. In particular, we employed various elements of the Hindi text files' structure as training features. These included the count of lines, the comma percentage, the number of words, the stop-words percentage, the

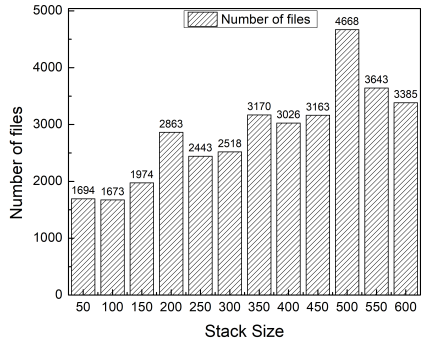

Fig. 2. Number of files with different Stack Size.

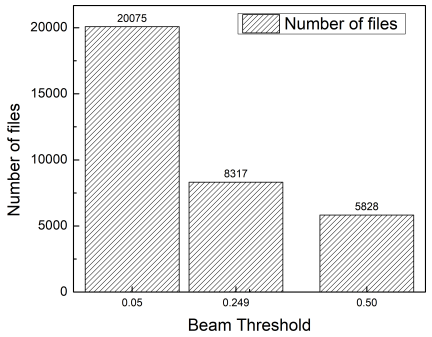

Fig. 3. Number of files with different Beam Threshold.

large sentence percentage, and the mean word count of every sentence. The target features were the stack size and beam threshold. To extract these structural features from the Hindi text files, we used the IndicNLP library[3] [27]. The training features are discussed below:

(1) Number of Lines: Line count in the input text.
(2) Percentage of Commas: The percentage of commas against the total cpunt of characters in the input text.
(3) Word Count: Word Cound in the input text.
(4) Percentage of Stop Words: The percentage of Hindi stopwords, e.g.: तक, ना, against the total count of words in the input text.
(5) Average Words per Line: The average number of words present in each line.
(6) Percentage of Long Lines: The percentage of long lines(greater than 17 words) against the total count of lines in the input text.

The dataset is created by decoding each file 36 times with different stack sizes and beam threshold values. The stack size values used are 50, 100, 150, 200, 250, 250, 300, 350, 400, 450, 500, 550, and 600, and the beam threshold values used are 0.5, 0.05, and 0.2495. These values were determined as the best-performing ones in [5]. Among these combinations, the best-performing n and α values are selected for each file based on the evaluation criteria of decoding time and BLUE score. The chosen parameter pair for a Hindi text file is the one that produces the maximum BLEU score with the lowest decoding time. Each file corresponds to one row in the resulting dataset.

The dataset that was prepared has been divided into two separate CSV files, one for training and the other for testing the regression models. The training dataset comprises of odd-even files and 3-file combinations, whereas the testing dataset consists of 59 files extracted from the corpus. A snapshot of the CSV created for training the regression model has been shown in Table 2. Figure 2 and Figure 3 display the count of files in the training set with varying Stack Sizes and Beam Thresholds. Table 3 provides additional information regarding the training and testing files used for regression.

[3]https://github.com/anoopkunchukuttan/indic_nlp_library

Table 3. Details of training dataset and testing dataset for regression.

Dataset	Description	Rows	Feature Columns	Target Columns
Training	odd_even + 3-file-combination	34220	Number of lines, Comma %, Word Count, Stopword %, Average Words per Line, Long Line %	Stack Size, Beam Threshold
Testing	59 files from corpus	59	Number of lines, Comma %, Word Count, Stopword %, Average Words per Line, Long Line % (Testing Features)	Stack Size, Beam Threshold (For Prediction)

4 EXPERIMENTAL SETUP

To create the dataset, 34,220 Hindi text files were decoded on 98 systems with Ubuntu 14.04 operating system, Intel i3-3.3 GHz processor, and 8 GB of memory. A total of 36 combinations of n and α were used. The decoding process was done in Mosesdecoder, and the smallest file had 330 lines, while the largest file had 2,070 lines. It took approximately 1.5 months (1,065 hours) to decode all the files, with an average time of 2.48 hours per file. The decoding time ranged from 10.78 minutes to 2.98 hours. Two CSV files were created, one for the training dataset and one for the testing dataset. Odd-even files and 3-file combination files were used for training, while 59 extracted files from the corpus were used for testing the regression models. The target features for regression were stack size and beam threshold. Out of the eleven different regression methods evaluated using the RMSE method, four of the best-performing methods are LightGBM, XGBoost, Random Forest Regressor, and Extra Trees Regressor (Figure 4). To further optimize the performance of these models, hyperparameter tuning is performed using the FLAML toolkit[4] [35]. FLAML is a Python library that is fast, affordable, and easy to use, and it quickly and efficiently finds reliable machine-learning models. By using FLAML, we are able to create a self-tuning model that adapts to the training dataset without the need for manual hyperparameter tuning. The model training and hyperparameter tuning is carried out on the Google Colab platform.

To perform better predictive analysis, we have proposed two machine learning models. The first model, named Ensemble Average [1], combines the predicted values from the four best-performing models by averaging them. The second model, named Pipeline, sequentially stacks the four selected models to create a more accurate prediction.

5 RESULT AND ANALYSIS

Initially, the testing files are decoded using the predefined parameters, which include the stack size (n) and beam threshold (α). The average BLEU score obtained for Hindi to English translation on 59 testing files is 9.199, while the average decoding time is 95.288 seconds. The stack values and threshold values for the testing files are predicted using the CN2 model, which was proposed in [5], and the four best-performing models, which are Light Gradient Boosting, XGBoost, Extra Trees, and Random Forest. These models were selected based on RMSE out of eleven regression models. Once the proposed machine learning models predict n and α, the Hindi files are re-decoded in

[4]https://github.com/microsoft/FLAML

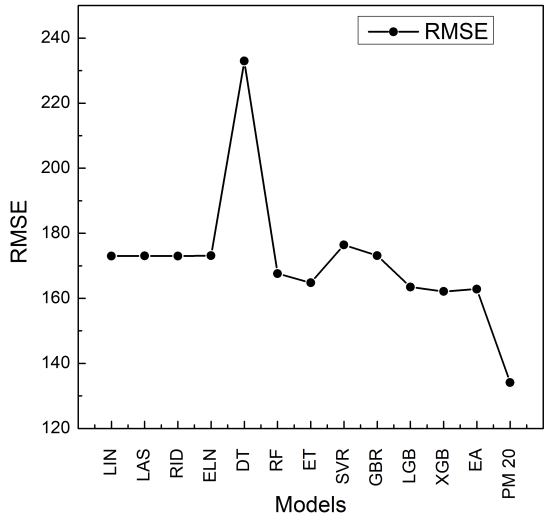

Fig. 4. RMSE scores for different models during training. (LIN: Linear Regression, LAS: Lasso Regression, RID: Ridge Regression, ELN: ElasticNet, DT: Decision Tree Regressor, RF: Random Forest Regressor, ET: Extra Trees Regressor, SVR: Support Vector Regressor, GBR: Gradient Boost Regressor, LGB: Light Gradient Boosting Regressor, XGB: XGBoost Regressor, EA: Ensemble Average, PM 20: Pipeline model 20)

Table 4. Comparison of results using predicted parameters using Individual Models with the number of files out of 59 testing files which got better BLEU score with respect to the pre-defined parameters.

Model	Predicted (Mean)				Number of files
	Stack Size	Beam Threshold	BLEU	Time (secs.)	
CN2	494.910	0.049	9.212	202.000	15
Light Gradient Boosting	373.982	0.158	9.211	157.406	19
XGBoost	**368.840**	**0.154**	**9.215**	**155.406**	**19**
Extra-Trees	371.230	0.162	9.213	155.745	19
Random Forest	371.943	0.157	9.215	156.338	20

Mosesdecoder. Table 4 presents the average outcomes from the individual models, along with the count of testing files that have better BLEU scores using these models.

The CN2 model improved the accuracy of 15 out of 59 testing files compared to the default parameters, resulting in a mean BLEU score of 9.212 and a decoding time of 202.000 seconds. Among the regression models, XGBoost and Random Forest performed the best in terms of mean BLEU score, with Random Forest yielding the highest number of files (20 out of 59) with a mean BLEU score of 9.215. XGBoost had the shortest mean decoding time of 155.406 seconds. The biggest improvement in BLEU scores using the regression models was 1.70. On average, implementing the regression models increased the mean BLEU score by 0.016 compared to the default values.

Table 5. Description for 24 pipeline models created by combining 4 best-performing models in a sequential manner with their RMSE score (LGBM: Light Gradient Boosting, XGB: XGBoost, RF: Random Forest, ET: Extra-Trees)

Models	Model 1	Model 2	Model 3	Model 4	RMSE
Pipeline Model 1	LGBM	XGB	RF	ET	152.587
Pipeline Model 2	LGBM	XGB	ET	RF	140.915
Pipeline Model 3	LGBM	ET	XGB	RF	136.199
Pipeline Model 4	LGBM	ET	RF	XGB	135.020
Pipeline Model 5	LGBM	RF	XGB	ET	146.449
Pipeline Model 6	LGBM	RF	ET	XGB	142.688
Pipeline Model 7	XGB	LGB	RF	ET	150.545
Pipeline Model 8	XGB	LGB	ET	RF	139.789
Pipeline Model 9	XGB	ET	LGB	RF	139.122
Pipeline Model 10	XGB	ET	RF	LGB	135.420
Pipeline Model 11	XGB	RF	LGB	ET	149.455
Pipeline Model 12	XGB	RF	ET	LGB	144.512
Pipeline Model 13	RF	ET	LGB	XGB	141.009
Pipeline Model 14	RF	ET	XGB	LGB	140.505
Pipeline Model 15	RF	LGB	XGB	ET	147.062
Pipeline Model 16	RF	LGB	ET	XGB	146.085
Pipeline Model 17	RF	XGB	LGB	ET	147.258
Pipeline Model 18	RF	XGB	ET	LGB	143.861
Pipeline Model 19	ET	LGB	RF	XGB	137.032
Pipeline Model 20	**ET**	**LGB**	**XGB**	**RF**	**134.147**
Pipeline Model 21	ET	RF	LGB	XGB	135.116
Pipeline Model 22	ET	RF	XGB	LGB	134.720
Pipeline Model 23	ET	XGB	LGB	RF	134.925
Pipeline Model 24	ET	XGB	RF	LGB	134.914

Two proposed models are evaluated for their effectiveness in predictive analysis. For the first model, named Ensemble Average, the four best-performing models are combined by averaging their predicted values. For the second model, named Pipeline, the four selected models are stacked sequentially. In the case of Pipeline, there are 24 different models to choose from, with Pipeline Model 20 having the best RMSE score, as shown in Table 5. To evaluate the effectiveness of these proposed models, they were implemented on two datasets: the Kaggle House Price Regression Dataset [14] and a prepared dataset. The mean RMSE score for the Kaggle dataset improved from 38876.07 using regression models to 27380.03 using pipeline models, resulting in a 29.59% improvement. Similarly, for the prepared dataset, the mean RMSE score improved from 164.503 to 134.147, a 18.45% improvement. The mean BLEU score, mean decoding time, and the count of files with mean BLEU scores using proposed model 1 and proposed model 2 are depicted in Figure 5, Figure 6, and Figure 7. Results from the proposed model 1 and proposed model 2 are shown in Table 6. Proposed Model 1, the Ensemble Average approach, showed improvement in 21 out of 59 files with respect to the BLEU scores with pre-defined parameters. The biggest improvement in the BLEU score using Ensemble Average is 1.71. For the Pipeline approach, Pipeline Model 6 resulted in 26 files with better BLEU scores, followed by Pipeline Model 7 with 24 files having better BLEU scores. Pipeline Model 8 resulted in the least mean decoding time of 150.729 seconds, while Pipeline

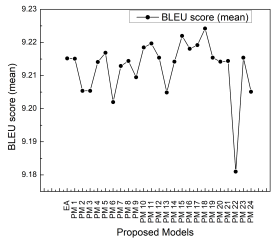

Fig. 5. Mean BLEU Scores for proposed models. (EA: Ensemble Average, PM 1, PM 2 ...PM 24: Pipeline Model 1, Pipeline Model 2...Pipeline Model 24)

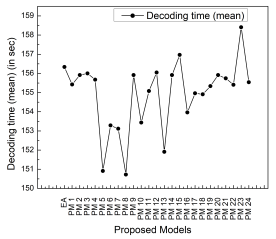

Fig. 6. Mean Decoding Time for different proposed models. (EA: Ensemble Average, PM 1, PM 2 ...PM 24: Pipeline Model 1, Pipeline Model 2...Pipeline Model 24)

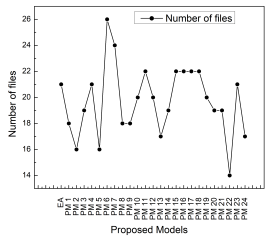

Fig. 7. Number of files having better BLEU scores w.r.t. scores by default values for different proposed models. (EA: Ensemble Average, PM 1, PM 2 ...PM 24: Pipeline Model 1, Pipeline Model 2...Pipeline Model 24)

Model 18 achieved the highest mean BLEU score of 9.224. The biggest improvement in the BLEU score was seen in Pipeline Model 23, where the BLEU score increased by 1.72. Pipeline Model 18 is constructed by sequentially stacking four selected models: Extra Trees, LightGBM, XGBoost, and Random Forest. Each model in the pipeline builds upon the predictions of the preceding model, refining and enhancing the translation output. The considerable improvement can be attributed to several models in the pipeline correcting the overall prediction error, bringing the final result closer to the ideal values for improved translation. Extra Trees builds the trees by selecting random feature thresholds, making them more diverse and less prone to overfitting. Bootstrapping is also used to sample the training data for each tree, enabling the algorithm to use more data per tree and achieve better performance. LightGBM, a gradient-boosting method, reduces memory usage and speeds up training by using histogram-based bucketing. Additionally, its leaf-wise growth strategy selects the highest-gain leaf node at each split, resulting in more accurate and deeper trees. XGBoost, on the other hand, builds shallow trees with few nodes to reduce overfitting. It also uses pruning to remove redundant components from the trees, thus decreasing overfitting and improving training time. Finally, the Random Forest ensemble method reduces variance and enhances prediction accuracy. By leveraging the strengths of multiple models and combining them in a sequential manner, Pipeline Model 18 effectively corrects prediction errors and refines the translation output.

5.1 Limitation

While the proposed regression models improve the BLEU score, they also result in significantly longer decoding times. This is due to the decoder having to compute a larger number of hypotheses since the regression models predict the decoding parameters as continuous values. Among the best-performing regression models, Light Gradient Boosting had the highest mean decoding time. In comparison, pipeline model 25 had the highest mean decoding time among the pipeline models. The mean decoding time for parameters from regression models was 157.228 seconds, while the mean decoding time for parameters from proposed models was 154.563 seconds.

5.2 Case Study

A comparison of different MT systems for Hindi to English translation has been shown in Table 7, where a source text in Hindi is translated to English using the available online tools, Google

Table 6. Comparison of results using predicted parameters using Ensemble Average (Proposed Model 1) and Pipeline (Proposed Model 2) Models with the number of files that got better BLEU scores with respect to the pre-defined parameters.

Model	Predicted (Mean)				Number of files
	Stack Size	Beam Threshold	BLEU	Time(secs.)	
Ensemble Average	371.612	0.158	9.2152	156.338	21
Pipeline Model 1	379.975	0.158	9.2151	155.424	18
Pipeline Model 2	372.211	0.157	9.2054	155.915	16
Pipeline Model 3	373.636	0.146	9.2141	156.000	19
Pipeline Model 4	360.077	0.052	9.2169	155.678	21
Pipeline Model 5	364.703	0.156	9.2020	150.915	16
Pipeline Model 6	**353.863**	**0.067**	**9.2129**	**153.288**	**26**
Pipeline Model 7	367.989	0.155	9.2080	153.119	24
Pipeline Model 8	**353.782**	**0.166**	**9.2144**	**150.729**	**18**
Pipeline Model 9	364.687	0.218	9.2095	155.915	18
Pipeline Model 10	351.514	0.225	9.2185	153.441	20
Pipeline Model 11	366.546	0.280	9.2197	155.085	22
Pipeline Model 12	360.048	0.224	9.2154	156.051	20
Pipeline Model 13	367.194	0.191	9.2049	151.915	17
Pipeline Model 14	386.165	0.213	9.2142	155.915	19
Pipeline Model 15	371.648	0.221	9.2220	156.966	22
Pipeline Model 16	373.993	0.124	9.2181	153.966	22
Pipeline Model 17	378.663	0.273	9.2192	154.966	22
Pipeline Model 18	**369.839**	**0.263**	**9.2242**	**154.915**	**22**
Pipeline Model 19	379.749	0.266	9.2154	155.339	20
Pipeline Model 20	373.666	0.282	9.2142	155.915	19
Pipeline Model 21	378.288	0.332	9.2144	155.746	19
Pipeline Model 22	362.948	0.321	9.1810	155.407	14
Pipeline Model 23	351.065	0.248	9.2154	158.407	21
Pipeline Model 24	363.513	0.335	9.2051	155.542	17

Translate and Microsoft Bing Translate. The translations generated by Mosesdecoder using pre-defined values and the proposed method are evaluated with the translations produced by Google Translate and Microsoft Bing Translate.

6 CONCLUSION AND FUTURE SCOPE

This research focuses on enhancing translation accuracy in the medical field, specifically for translating from Hindi to English, by introducing a novel approach that dynamically allocates decoding parameters through regression models. The study addresses the limitations of fixed parameters in the decoder by creating a comprehensive dataset and utilizing standard metrics like the BLEU score for evaluation. The pipeline approach further enhances translation accuracy by sequentially stacking the selected models. Pipeline Model 23 showcases remarkable improvements in the BLEU score, achieving higher translation quality compared to other models in the pipeline. Similarly, Pipeline Model 18 demonstrates the highest mean BLEU score among the pipeline models, indicating its ability to generate accurate and high-quality translations. The proposed models employ diverse techniques and ensemble methods to reduce overfitting, enhance prediction accuracy, and correct

Table 7. Comparative analysis of Hindi to English MT system using Google translate (as of 12^{th} December 2022), Microsoft Bing Translate (as of 12^{th} December 2022), and the proposed method, where Source text and Reference text are the source sentence and its reference translation.

Source Text	मुझे विश्वास है कि सभी चार सरकारों की सक्रिय भागीदारी और से हम ऐसी चुनौतियों हमारे अंतर्राष्ट्रीय साझेदारों की सहायता से निपट सकते हैं।
	mujhe vishvaas hai ki sabhee chaar sarakaaron kee sakriy bhaageedaaree aur hamaare antarraashtreey saajhedaaron kee sahaayata se ham aisee chunautiyon se nipat sakate hain.
Reference Text	I am confident that with the active engagement of all four governments, and the support of our international partners, we can overcome such challenges.
Google Translate (as on 12-12-2022)	I am confident that with the active participation of all four governments and the support of our international partners, we can overcome such challenges.
Bing Translate (as on 12-12-2022)	I am confident that with the active participation of all four governments and the help of our international partners, we can meet such challenges.
Moses decoder with default parameters	I believe that all the four सरकारों active भागीदारी and our अंतर्राष्ट्रीय साझेदारों with the help of a चुनौतियों we can निपट हैं।
Predicted Parameters (Proposed Model 1)	I believe that all the four सरकारों active भागीदारी and our अंतर्राष्ट्रीय साझेदारों with the help of a चुनौतियों we can from निपट हैं।
Predicted Parameters (Proposed Model 2)	I believe that all four सरकारों is the active and our अंतर्राष्ट्रीय साझेदारों भागीदारी with the help of a चुनौतियों निपट from can we हैं।

errors. These models demonstrate their effectiveness in improving translation accuracy, outperforming existing methods in terms of BLEU scores. By facilitating the translation and sharing of medical literature, this research promotes collaboration and knowledge exchange across languages, thereby improving access to medical information and healthcare services for multilingual patients. The regression method for optimizing pruning parameters, introduced for the first time in this study, contributes to the field by enhancing translation accuracy in the medical domain. The proposed models offer promising results and pave the way for enhanced machine translation systems. The research highlights the effectiveness of the regression method and the pipeline approach in optimizing pruning parameters, leading to improved translation accuracy and fostering collaboration in the medical field. Future studies can build upon these findings to further advance machine translation capabilities and facilitate seamless communication in healthcare field.

FUNDING ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the grant GUJCOST/STI/2021-2022/3922 from the Government of Gujarat, India, and support from Pandit Deendayal Energy University (PDEU) to perform this work.

REFERENCES

[1] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. 2021. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *arXiv preprint arXiv:2110.10832* (2021).

[2] Debajyoty Banik. 2021. Phrase table re-adjustment for statistical machine translation. *International Journal of Speech Technology* 24, 4 (2021), 903–911.

[3] Debajyoty Banik, Pushpak Bhattacharyya, and Asif Ekbal. 2016. Rule based hardware approach for machine transliteration: A first thought. In *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 192–195.

[4] Debajyoty Banik and Riya Roy Chowdhury. 2016. Offline signature authentication: A back propagation-neural network approach. In *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 334–339.

[5] Debajyoty Banik, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Machine learning based optimized pruning approach for decoding in statistical machine translation. *IEEE Access* 7 (2018), 1736–1751.

[6] Debajyoty Banik, Asif Ekbal, and Pushpak Bhattacharyya. 2018. Wuplebleu: The wordnet-based evaluation metric for machine translation. In *15th International Conference on Natural Language Processing*. 104.

[7] Debajyoty Banik, Asif Ekbal, Pushpak Bhattacharyya, and Siddhartha Bhattacharyya. 2019. Assembling translations from multi-engine machine translation outputs. *Applied Soft Computing* 78 (2019), 230–239.

[8] Debajyoty Banik, Asif Ekbal, Pushpak Bhattacharyya, Siddhartha Bhattacharyya, and Jan Platos. 2019. Statistical-based system combination approach to gain advantages over different machine translation systems. *Heliyon* 5, 9 (2019), e02504.

[9] Debajyoty Banik, Sukanta Sen, Asif Ekbal, and Pushpak Bhattacharyya. 2016. Can smt and rbmt improve each other's performance?-an experiment with english-hindi translation. In *Proceedings of the 13th international conference on natural language processing*. 10–19.

[10] Adam Berger, Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, John R Gillett, John Lafferty, Robert L Mercer, Harry Printz, and Lubos Ures. 1994. The Candide system for machine translation. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

[11] Ondřej Bojar, Vojtěch Diatka, Pavel Rychlỳ, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. Hindencorp-hindi-english and hindi-only corpus for machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. 3550–3555.

[12] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics* 16, 2 (1990), 79–85.

[13] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[14] Yihao Chen, Runtian Xue, and Yu Zhang. 2021. House price prediction based on machine learning and deep learning methods. In *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*. IEEE, 699–702.

[15] Ondřej Dušek, Jan Hajic, Jaroslava Hlaváčová, Michal Novák, Pavel Pecina, Rudolf Rosa, Aleš Tamchyna, Zdenka Uresova, and Daniel Zeman. 2014. Machine translation of medical texts in the khresmoi project. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 221–228.

[16] Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2004. Fast and optimal decoding for machine translation. *Artificial Intelligence* 154, 1-2 (2004), 127–143.

[17] Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*. 187–197.

[18] Hieu Hoang and Philipp Koehn. 2008. Design of the moses decoder for statistical machine translation. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. 58–65.

[19] Junjie Hu, Mengzhou Xia, Graham Neubig, and Jaime Carbonell. 2019. Domain adaptation of neural machine translation by lexicon induction. *arXiv preprint arXiv:1906.00376* (2019).

[20] Marcin Junczys-Dowmunt. 2012. A space-efficient phrase table implementation using minimal perfect hash functions. In *International Conference on Text, Speech, and Dialogue*. Springer, 320–327.

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[22] Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational linguistics* 25, 4 (1999), 607–615.

[23] Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas*. Springer, 115–124.

[24] Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872* (2017).
[25] Philipp Koehn, Franz J Och, and Daniel Marcu. 2003. *Statistical phrase-based translation.* Technical Report. University of Southern California Marina Del Rey Information Sciences Inst.
[26] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 48–54.
[27] Anoop Kunchukuttan. 2020. The indicnlp library. *Indian language NLP Library* (2020).
[28] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.
[29] Daniel Marcu and Daniel Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. 133–139.
[30] Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics* 30, 4 (2004), 417–449.
[31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.
[32] Sukanta Sen, Debajyoty Banik, Asif Ekbal, and Pushpak Bhattacharyya. 2016. Iitp English-Hindi machine translation system at wat 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*. 216–222.
[33] Nandini Sethi, Amita Dev, Poonam Bansal, Deepak Kumar Sharma, and Deepak Gupta. 2022. Hybridization Based Machine Translations for Low-Resource Language with Language Divergence. *ACM Transactions on Asian and Low-Resource Language Information Processing* (2022).
[34] Shumin Shi, Xing Wu, Rihai Su, and Heyan Huang. 2022. Low-Resource Neural Machine Translation: Methods and Trends. *ACM Transactions on Asian and Low-Resource Language Information Processing* 21, 5 (2022), 1–22.
[35] Chi Wang and Qingyun Wu. 2019. Flo: Fast and lightweight hyperparameter optimization for automl. (2019).
[36] Krzysztof Wołk and Krzysztof Marasek. 2015. Neural-based machine translation for medical text domain. based on european medicines agency leaflet texts. *Procedia Computer Science* 64 (2015), 2–9.
[37] Makoto Yasuhara, Toru Tanaka, Jun-ya Norimatsu, and Mikio Yamamoto. 2013. An efficient language model using double-array structures. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 222–232.
[38] Richard Zens and Hermann Ney. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 492–499.