IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2018, accepted November 22, 2018, date of publication December 25, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2883738

Machine Learning Based Optimized Pruning Approach for Decoding in Statistical Machine Translation

DEBAJYOTY BANIK, ASIF EKBAL, AND PUSHPAK BHATTACHARYYA
Department of Computer Science and Engineering, Indian Institute of Technology Patna, Bihta 801103, India
Corresponding author: Debajyoty Banik (debajyoty.banik@gmail.com)

ABSTRACT A conventional decoding algorithm is critical to the success of any statistical machine translation system. Providing an enormous amount of space leads to inappropriate slow decoding. There is a trade-off between the translation accuracy and the decoding speed. Pruning algorithms (like histogram pruning, threshold pruning) are trying to optimize this. The pruning algorithm has a pre-defined limit on the supplemental parameters (i.e. stack size, beam threshold) that helps to improve the translation quality and speed up the decoder. However, the same parameter value cannot provide the qualitative translation in optimum time. These stack size and beam threshold values should be changed based on texts' structures. In this paper, we identify the best stack size and beam threshold values runtime based on the text structure and characteristics using a machine learning-based approach. Then, the values of these parameters are applied into the beam search algorithm for decoding. Finally, our experiments on low-resourced Asian languages show significant performance improvements in terms of their translation accuracy and decoding time. The HindEnCorp and ILCI datasets are used as the benchmark datasets with English-Hindi, Hindi-Marathi, Hindi-Konkani, Bengali-Hindi language pair, for our various experiments. Moreover, we incorporate the proposed technique in cube pruning algorithm for faster decoding. We notice more improvement in this approach.

INDEX TERMS Machine learning, machine translation, decoding, evaluation metric.

I. INTRODUCTION

Statistical machine translation (SMT) has received enormous interest in the field of natural language processing community. Best performing statistical machine translation systems are based on phrase-based models that aim to reduce the restrictions of word-based translation by translating whole sequences of words, where the lengths may vary. The sequences of words are called phrases. Mainly three models are incorporated into the phrase base SMT system for decoding, the translation model, the distortion model, and the language model [32]. Mathematically, it can be modeled as:

$$e_{best} = argmax_e P(e|f) = argmax_e [P(f|e) P_{LM}(e)] \quad (1)$$

Where, f, e, have their usual meanings of output and input, respectively. The translation with the highest score is denoted as e_{best}. $P_{LM}(e)$ is language model, and $P(f|e)$ is the translation model. The translation model can be expressed as:

$$P({f_1}^I|{e_1}^I) = \prod_{i=1}^{I} \phi(f_i|e_i) d(start_i - end_{i-1} - 1) \quad (2)$$

where, $\phi(f_i|e_i)$ is the phrase translation probability, which is the probability of translating to phrase e_i from the phrase f_i. The phrase translation probability is learned from large parallel corpora. The distortion probability $d(start_i - end_{i-1} - 1)$ is settled with an exponential cost. The simple meaning of distortion cost is how many input words are skipped to generate the next output phrase. The language model measures how likely it is that a sequence of words would be in the target text. The decoding process is handled by segmenting input sentence f into the sequence of I phrases ${f_1}^I$, which are then distributed over all possible segments. The goal of the machine translation model is to find the better translation [30], [5] in a reasonable time. The decoding problem for statistical machine translation model is NP-complete [26].

The exhausting analysis of all possible translations, scoring them, and then finally selecting the best-translated output is very costly. So, the decoder performs a heuristic search on the search space and finds the translation closest to the best translation (e_{best}).

The decoder's task is to trace the translation that is most probable as per arrangement of already learned parameters. Traditional decoding of statistical machine translation model uses beam search [28] algorithm that tries to find the best translation. Generation of a huge number of hypotheses in the stacks is a normal phenomenon for the exponential nature of machine translation decoding. The main challenge in machine translation decoding is large numbers of hypotheses in the stacks, for its exponential nature of hypothesis generation. Comparatively poor hypotheses need to be pruned out from the hypothesis list to speed up the decoding. Two types of pruning algorithms are prevalent for this task; i.e. histogram pruning and threshold pruning [41]. Maximum number (n) of hypotheses in the stack is considered for histogram pruning. In threshold pruning, only those hypotheses which have the score higher than a fixed value are allowed in the stack. So, if the score of the next hypothesis is less than this fixed value, it is pruned out. In existing methods, these values are predefined and fixed. We propose a mechanism to find optimal values for these parameters dynamically. The primary goal of this work is to decode into the target sentence in optimal time with better accuracy.

To make decoder faster, cube pruning [23] is an alternate way of the search algorithm in addition to the traditional approach. We have found out that cube pruning [8], [18] is not only faster than beam search but also gives a much higher translation accuracy in terms of BLEU. Very few works have been done with cube pruning. We have also presented a comparison between traditional beam search based and cube pruning based decoding approaches to show their accuracy and decoding time over differ over different source texts. The performance improvement with machine learning based parameter selection has been also described in our paper. Cube pruning algorithm [8] has certain parameters that can be used to improve the decoding process. It is also impossible to manually predict the correct value of these parameters like the selection of beam search algorithm's parameters values for different inputs. We have incorporated the proposed method with cube pruning algorithm to have better translation accuracy in lesser time than any kind of existing decoding approaches. We have achieved the goal after classifying input text by using a machine learning approach and then applying the best set of parameters while decoding. Finally, we have incorporated our technique to Moses toolkit[1] which is a free open source statistical machine translation engine and is well approved by the community. We incorporate the proposed framework with Moses2[2] which is cube pruning decoder for optimal decoding.

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

[2]https://github.com/moses-smt/mosesdecoder/tree/master/moses2

The decoding is an NP-hard problem [27]. Providing an enormous amount of space leads to inappropriate slow decoding. Since the space of probable translation is too large, traditional decoding algorithms are just ready to explore its segment. Hence, there is a chance to miss the better solutions. Some pruning strategies have been integrated with statistical machine translation decoding to keep the better solutions for better translation in optimal time. The pruning algorithm has a pre-defined limit on supplemental parameters that help to improve translation quality and speed up the decoder. The main problem of the existing machine translation decoder is using predefined static parameters (i.e. stack size, beam threshold) for decoding. But every text may have a different structure. So, these parameter values should be changed proactively for optimal decoding. It is quite hard to predict the combination of these parameters' values which will be optimum for a specific piece of text. Higher decoding speed usually affected the cost of translation quality.

In this paper, our contribution is to identify the best stack size and the beam threshold values dynamically and apply the values of these parameters during decoding. First, we use beam search algorithm to decode. This paper aims to create a simple but effective model for optimizing the decoding algorithm, in terms of accuracy and speed. The paper shows, the technique to dynamically choose the optimal stack size and the beam threshold values in the beam search algorithm which provides promising accuracy improvement in terms of the BLEU score and decoding time. Moreover, we use the cube pruning algorithm after dynamically selecting the parameters' values to have more optimal translated output in term of accuracy in minimum time. It also validates the inheriting power of our proposed approach.

A. RELATED WORK

The open source toolkit for statistical machine translation (Moses) has been designed in [19]. Two new decoders were proposed for fast and optimal decoding [15]. These two decoders have opposite characteristics: a slow but optimal decoder that treats decoding as an integer-programming optimization problem and a fast but non-optimal greedy decoder. They compared output quality and the speed of traditional stack-based decoding approach with new decoders.

Researchers described language models which are very efficient and fast in [17] and [39]. The phrase table implementation which is loaded on demand for the SMT decoder was done in [40]. This method reduces the memory requirements and initial loading time. Extend version (by compressing the on-disk phrase table and lexicalized re-ordering model) of this method was noted down in [25]. In [8] the cube-pruning and the cube-growing algorithm were introduced that allows the trade-off between translation quality and speed to the adjusted with a single parameter. Hoang [19], Li [35] support cube-pruning for phrase-based statistical models. Hoang *et al.* [18] re-examine the major components of phrase-based decoding and decoder implementation with

specific accentuation on speed and versatility on multi-core machines. The outcome is a drop-in substitution for the Moses decoder which is up to fifteen times faster and scales monotonically with the number of cores. In [33], translation accuracy was increased by using sub-word level decoding but decoding time also increased to a great extent (about 70 times). So, they tried to reduce the decoding time by manipulating various factors that can affect the decoding process but still, the decrease was not much and the decoding time was 63 times more than the baseline case. Koehn *et al.* [30] presented the Moses decoder[1] for phrase-based machine translation. This tool was used to some extent in our experiments. Traditional SMT uses the beam search algorithm for finding the best translation. But in SMT, same decoder parameters are applied for decoding whatever the source text is unless specified otherwise. But it is not possible for the user to decide the set of parameter's values for different input. There is no such inbuilt mechanism to do so. We know that there is always a trade-off between better translation and decoding time in machine translation. There is still an effort to decrease the decoding time or increase the translation accuracy or both. But almost all the efforts were made till now that try to improve either decoding algorithm [12] or tune the parameters of the training model (MERT) [2]. But in every case, the process of decoding is applied the same way independent of the input. A fundamental fact has never been considered so far, a certain type of text file, like in stack decoding might require a different value stack size and beam threshold. This scenario is not an exception for other decoding algorithms, like cube pruning. In this paper, we have shown that parameters' values are a very crucial factor for decoding time and translation accuracy which were not explored before. We propose a machine learning based parameters selection technique for better decoding. As per the best of our knowledge, the concept of machine learning approach for machine translation has been introduced for the first time in the literature.

B. MOTIVATION

Though the neural machine translation (NMT) [38] is the recent trend but it is not a good option for low resource languages due to its data hungriness. Though there are several techniques for low resource languages using NMT approach but statistical machine translation (SMT) approach is better till now than NMT system [31].

In SMT approach, some predefined values for stack size and beam threshold may not provide the best translation in the optimal time for every case which is shown in Table 2. Here, the optimal stack size is different for different the ten texts. Texts which are going to be decode may have different structures. A long sentence with complex structure may require a large stack size than the comparatively sort sentence with the simple sentence. Not only that, the parameter values depend on the number of stop words, pause, etc. Moreover, a text may have the variety of sentences. Mix types of sentences are very common. So, the same values for these parameters is not a good choice for all cases. How the translation accuracy and decoding time depend on different values is shown in Table 1. Similar behaviour is noticeable for beam threshold value. An intelligence system is required to find the optimal parameter values. In this paper, we propose the machine learning based intelligence system to find the optimal parameter values on runtime and automatically apply these parameter values for decoding (using beam search algorithm and cube pruning algorithm) which help us to achieve better translation output in minimal time. Lots of files are handled during the experiments. We maintain a convention for future use. Our convention for the naming is <type>.<language>.<file number>. For an example, f.en.1 is the first source file in the English language.

TABLE 1. Detailed decoding performance analysis of single file (f.en.6) for various stack sizes and default beam threshold; The decoding has been done for the file f.en6 which consists of 419 sentences or 6116 tokens.

Different stack sizes (s)	Decoding Time (in sec)	BLEU
s = 10	12.131	15.77
s = 50	34.643	16.08
s = 100	65.167	16.23
s = 150	95.821	16.17
s = 200	125.326	16.18
s = 250	154.283	16.16
s = 300	188.145	16.18
s = 350	214.04	16.18
s = 400	241.513	16.18
s = 450	267.826	16.18
s = 500	305.024	16.77
s = 550	333.132	16.18
s = 600	366.645	16.18
s = 650	396.202	16.18
s = 700	423.579	16.18
s = 750	455.309	16.18
s = 800	492.565	16.18
s = 850	530.863	16.18
s = 900	560.738	16.18
s = 950	571.772	16.18
s = 1000	612.192	16.18

There are various strategy to use beam search algorithm in NMT approach [1], [13]. Here, the beam search is used to find a translation that approximately maximizes the conditional probability [16], [4] after a model is trained. This approach is used by [38] to generate translations from their neural machine translation model. We can also incorporate our technique here to identify and apply the optimal parameter values for beam search in NMT system, which is left for interested researchers in the recent future.

II. MACHINE LEARNING BASED DECODING APPROACH

The decoder has to find the best plausible translation that is in agreement with earlier trained data. Since the number of possible translations are huge, decoding algorithms are only able to examine a part of it, risking to overlook better translations. Since the decoding problem is NP-complete [26], the time required for optimal decoding increases exponentially with the length of the input.

We have used Moses [30], [5], [20] for phrase-based statistical machine translation model. Moses uses a beam search

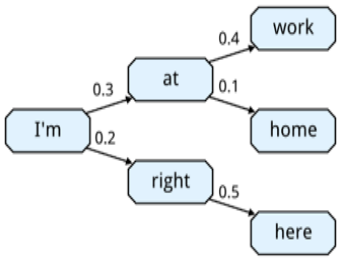

FIGURE 1. Example of the beam search tree.

algorithm that tries to find the best translation. A beam search is a heuristic search method that integrates features of breadth-first and best-first search methods. Only the most promising nodes known as the beam width (instead of all nodes) at each step of the search are retained for further branching. Thus, it reduces the time and memory consumption [29]. But still decoding is a computationally expensive process. Traditional statistical machine translation system uses stack decoding, where hypotheses are stored in stacks and then pruning of bad hypotheses is done.

Beam search uses a state-space search procedure, like best-first search (BFS) or depth-first search (DFS) [42]. What makes this search technique apart from others, is the use of heuristic rules to prune out bad search results even before exploring them.

Figure 1 shows the working of beam search in the machine translation system. During stack decoding hypotheses stacks are created and then beam search is carried out to find the translation with highest probability score (e_{best}). The probabilities are assigned and the translation with the best translation is chosen.

A. DECODER PARAMETERS IN TRADITIONAL BEAM SEARCH ALGORITHM (STACK DECODING)

In machine translation, the goal is to find the best possible translation, but even for a sentence of fair size, the number of possibilities generated is so large that it is impossible to go through all of the possible translation and find the best translation. Even after applying heuristic search algorithms which reduce the search space, the process still takes a lot of time, and there is always a trade-off between accuracy and time. The sophisticated phrase-based statistical machine translation algorithm [32] (implemented in Moses decoder) includes some vital decoder parameters that can be used to control the search space and affect decoding time without decreasing accuracy. In many cases choosing the right set of parameters not only reduced decoding time but also increased translated accuracy.

The different texts have different sentence structures. We propose that distinct text files if decoded with the right parameters, there would be a possibility for better translations with less decoding time. So, it is essential to select the specific decode parameters for optimal result. If we do not choose the right parameters for the source text, we might even get worse accuracy in spite of taking more decoding time. Our main aim is to reduce the decoding time and simultaneously to keep the translation as accurate as possible. So, to decide the right value for our parameters for any given text we have tried to integrate a machine learning classifier which can determine the best set of parameters for us.

If we could select the search space in such a way containing only the best translation options for a given file, then decoding time will be minimal, and we might even get better accuracy. One way to reduce the search space correctly is to set the optimum size of hypothesis stacks for a file. For each translated foreign phrase, the decoder keeps a stack of the best (partial) translations. So, we have to select the right stack size so that the better translation does not get pruned out. But the user doesn't know what values would be optimal for his data. Our primary goal in this paper is to solve this problem. By reducing this stack size, the search will be quicker, since fewer hypotheses are kept at each stage, and therefore fewer hypotheses are generated. Decoding leads to huge numbers of hypotheses in the stacks and the bad hypotheses have to be pruned out. In this proposed methodology, parameters for both of the pruning algorithms have considered for better performance:

- Histogram Pruning: The topmost of n hypotheses are kept in the stack, and the rest are pruned out. The value of n is directly proportional to the decoding time. Also, sometimes good hypotheses are pruned out, and bad hypotheses are kept.
- Threshold Pruning: Main aim of the threshold pruning is the hypothesis having α times worse score than the best hypothesis, that very hypothesis will be pruned out. α is threshold pruning parameter, where $0 \leq \alpha \leq 1$. Among all hypotheses for a source sentence in position j, $Q_0(J)$ is the maximum probability. The hypothesis is pruned out iff:

$$Q(n, e_1^i) < \alpha.Q_0(J) \tag{3}$$

  Where $Q(n, e_1^i)$ is the maximum probability of a phrase sequence that results in the word sequence e_1^i. We have shown in Section II-A, how quality and decoding time vary for the different values of n and α. Finally, we have identified dynamically optimal values of these parameters for decoding in optimal time with better accuracy. But according to best of our knowledge, there is no such method to select their optimal value for n and α.

In these days, decoders use both histogram pruning and threshold pruning to get the translated output at a reasonable time. stack size (s) and beam threshold (b) are controlling parameters for histogram pruning and threshold pruning respectively. Currently, phrase-based machine translation system (i.e., Moses) uses a default set of values during decoding for any input unless specified otherwise. But the issue with this approach is that different pieces of text have a different syntactic structure and different length. For example, some texts consist of more short sentences while

TABLE 2. Observation to find classes from various files with different size. The reported Class (stack size) is responsible for optimal result. Default beam threshold is used here.

File name	Sentence-count	Token-count	Selected optimal stack size (s)	Decoding Time (in sec)	BLEU
f.en.1	568	7915	s = 200	325.121	16.26
f.en.2	1170	16769	s = 400	749.125	17.24
f.en.3	1236	16585	s = 350	633.562	16.31
f.en.4	1130	15284	s = 300	450.694	16.96
f.en.5	765	9801	s = 150	129.925	17.19
f.en.6	419	6116	s = 100	65.167	16.23
f.en.7	1137	19938	s = 400	1365.824	14.04
f.en.8	537	8094	s = 300	607.686	12.59
f.en.9	954	12132	s = 150	138.502	15.12
f.en.10	833	11542	s = 350	468.068	16.45

TABLE 3. Detailed decoding performance analysis of files for various threshold and default stack size.

File name	Sentence-count	Token-count	Beam Threshold (b)	Decoding Time (sec)	BLEU
f.en.1	568	7915	0.001	322.479	16.26
			0.005	319.783	16.26
			0.01	321.476	16.26
			0.025	318.722	16.26
			0.05	320.311	16.26
			0.1	321.235	16.26
			0.2495	312.76	16.26
			0.5	198.962	16.26
f.en.2	1170	16769	0.001	344.984	17.2
			0.005	360.613	17.2
			0.01	363.721	17.2
			0.025	341.285	17.2
			0.05	360.438	17.2
			0.1	341.04	17.2
			0.2495	330.726	17.2
			0.5	229.711	17.16

some have long sentences or simple and complex sentences. We know that the decoding time for a sentence increases with sentence length. So, different texts may require different values of these parameters. But it is difficult to tell which value will give the best results for a particular set of sentences. So, in this paper, we have built a classification model using machine learning approach to solve this problem. [33].

Here, our primary goal is to decrease the decoding time but, without causing any harm to translation accuracy. If possible, we have also tried to increase the accuracy of translated text by selecting the proper values of stack size and beam threshold automatically. We have observed from our experiments, that stack size in range 10 to 1000 changes decoding time and accuracy of translated output. Outside that range, the accuracy is very low and/or the decoding time is extremely delayed. Different files from different domains were decoded using different values of stack size.

We found promising results, such as, for a particular file, till an absolute value of stack size, the accuracy of the translation increases with the stack size and for higher values, only the decoding time increased the accuracy either decreased or remained constant. One interesting finding would be that default stack size didn't perform very well with all types of source text rather in some cases the translation was more accurate for certain values of stack size (such as 100 or 150, etc.) and the decoding time also decreased considerably. Nevertheless, the time always increased with increasing stack size as the search also increased. Table 1 shows the behaviour for one such file. Here the stack size is increased from 10 to 1000 with a difference of fifty between two consecutive cases. The decoding time increases with the stack size. But the accuracy (in term of BLEU score [36]) has increased to stack size 250 and has remained constant since. After a detailed analysis (decoding) of various files, we have found eight classes (stack size) which are responsible for providing optimal results with respect to BLEU and time. Later these classes have been used for classification. Some of the files with their optimal classes are noted down in Table 2.

On the other hand, the results for threshold pruning, the parameter beam threshold showed entirely different behavior. The default value for beam threshold seems to be a tiny one, so the resultant translation is unaffected. We have noticed from our observations that the desirable beam threshold range is between 0.001 to 0.5.

After detailed analysis with lots of source files to select optimal classes for beam threshold (b), we have found an unusual behavior. Table 3 shows the behaviour of files concerning changes in beam threshold while decoding. Here, the stack size is kept as default. We had encouraging results in this situation. There was no direct relation between time and beam threshold but at beam threshold value 0.5 the decoding time for file f.en.1 is lowest with the highest accuracy.

For this particular source file, the difference is not big enough, but there is a noticeable decrease in decoding time. But after lots of observations with different data, it is noticed that this is not the case for every source file. Other showed a decrease in translation accuracy, but there was a decrease in decoding time in all cases. For the example of file f.en.2, at beam threshold value 0.5 the BLEU score decreases, but there is also a substantial decrease in decoding time.

B. MACHINE LEARNING BASED PARAMETERS SELECTION

To solve the problems as discussed in previous sections, some fast and automatic decision-making processes are required, which can analyze the input source text and decide parameters fit for its decoding. This decision-making process must choose such parameter values which are capable of better translating in minimal decoding time (the optimal translation). A pretty obvious solution to this problem is implementing a machine learning classification model, which is consolidated into our work. We have used the CN2 unordered algorithm as the classifier [9], [34], which is a learning algorithm for rule induction. Training data of this classifier is set of optimal translated outputs using SMT decoder and its corresponding stack size and beam threshold values (selected as classes). Thus, enough amount of data is the constraint here. The CN2 unordered algorithm is designed in such a way so that it can handle this problem. This algorithm can work with a small amount of data. This algorithm will work even if the training data is imperfect [10]. The size of our training dataset for the classifier is not enough, so we prefer this CN2 algorithm rather other standard classification approach used in the NLP community, i.e. decision trees, support vector machine, etc. The central concepts are based on Algorithm quasi-optimal learning (AQ algorithm) [7] and the Iterative Dichotomiser 3 (ID3) algorithm. As a result, it creates a rule set like that produced by AQ but is capable of handling noisy data like ID3. It is a type of sequential covering algorithms. A covering algorithm is an association rule algorithm that creates a cover for the set of positive instances, i.e., a set of hypotheses that account for all the positive instances but none of the negative instances. Association rule learning is a rule-based machine learning method for finding important relationships between variables. The CN2 algorithm is a classification mechanism created for the efficient induction of simple, easy to understand, rules of form ''condition-based class prediction'', even in domains where noise may be present. It modifies the standard sequential covering algorithm in some points:

- It learns rules that cover all classes of training examples.
- It accepts rules with certain accuracy; hence it can handle noises.
- Its Learn One Rule procedure is not depended on any particular monitoring example.
- It generates either an ordered or unordered set of rules.

CN2 has a choice to generate ordered or unordered set of rules as their final result. There are good and bad points on both options.

- Ordered: induce ordered rules (decision list). Rule conditions are found, and the majority class is assigned in the rule head.
- Unordered: induce unordered rules (rule set). Learning rules for each class individually, regarding the original learning data.

The primary challenge in building our classification model was how to classify text. We cannot use the conventional ways of text classification due to many reasons. Till now the text classification models are mostly built using the popular 'bag-of-words' model [21] or n-gram model [6] or syntactic and semantic classification. Here, we need to classify documents by structure and machine translation phenomena. So, we cannot use either bag-of-words model or n-gram model as the use word frequencies to classify text. Also, the concept of syntactic and semantic parsing was not viable as it took lots of time to parse a document. Since here our main aim is to decrease the decoding time, this concept could not be used. Now we had to come up with a new way to classify text which is discussed in the next section.

1) FEATURES USED

We discuss a classification mechanism using a machine learning approach to predict the right values of parameters for a particular piece of text. The output of the classifier will be such that the accuracy must not decrease, but in many cases, it may increase while lowering the decoding time simultaneously. The goal of our project requires us to classify text, not on the basis of the content of the text rather on its structure and type of sentences. The English word order is S-V-O, whereas for For Hindi, it will be changed to S-O-V order. Though SMT system does not consider any linguistic structure of the sentences, the distortion model is there to handle this challenge statistically. So, if the sentence structure is complex and long, it will be challenging to reorder and may require a larger search space to find the best translation. Similarly, smaller sentences will be decoded easily in a smaller search space. So, the lower stack size for larger sentences will lead to poor translation, whereas higher stack size for small sentence will lead to worse decoding time. Same kinds of things are applicable for beam threshold as discussed earlier. Since every file in our data has a different structures and a different number of lines, so, the following features are used to classify the text for our task:

- Percentage of the comma (,) in a text: The percentage of the comma against the total number of characters. The comma means a pause in a sentence.
- Percentage of long sentences in the text: The percentage of sentences with more than δ words is calculated. Texts containing more long phrases will belong to another class than texts with more short sentences.
- Average words per line in the text: The total number of words against the number of lines.
- Percentage of Stop words: Text files contain many stop words like 'the', 'is', 'are', 'and' etc. Different texts

TABLE 4. Snapshot of Training Data for stack size Selection. stack size 100 is represented by "class a", 150 is represented by "class b", 200 is represented by "class c" and so on, up to stack size 550 which is represented by "class j". Description of file name is like time.<text part name>.<b><beam size>.

Percentage of Comma	Percentage of Stop Words	Percentage of Long Sentence	Average Word Per Sentence	stack size	File Name
continuous	continuous	continuous	continuous	discrete	discrete
				class	meta
0.912	63.418	30.473	14.162	g	time.f11.b0.2495
0.903	64.53	25.869	14.705	f	time.f11.b0.05
0.93	63.744	27.408	14.266	b	time.f13.b0.5
0.865	63.921	31.743	14.632	e	time.f14.b0.5
0.915	63.85	25.81	13.51	a	time.15.b0.2495
0.907	63.138	24.212	13.961	b	time.f16.b0.2495
0.965	62.83	26.466	14.001	c	time.f17.b0.05
0.878	63.057	27.257	13.298	e	time.f19.b0.2495
0.897	62.955	25.905	13.463	a	time.f20.b0.5

TABLE 5. Snapshot of Training Data for Beam Threshold Selection. Beam threshold value 0.2495 is represented by "class x"; 0.05 by "class y" and 0.5 by "class z".

Percentage of Comma	Percentage of Stop Words	Percentage of Long Sentence	Average Word Per Sentence	Beam Threshold	FileName
continuous	continuous	continuous	continuous	discrete	discrete
				class	meta
0.912	63.418	30.473	14.162	x	time.f11.b0.2495
0.903	64.53	25.869	14.705	y	time.f11.b0.05
0.93	63.744	27.408	14.266	z	time.f13.b0.5
0.865539033	63.921	31.743	14.632	z	time.f14.b0.5
0.915	63.85	25.81	13.51	x	time.15.b0.2495
0.907	63.138	24.212	13.961	x	time.f16.b0.2495
0.965	62.83	26.466	14.001	y	time.f17.b0.05
0.878	63.057	27.257	13.298	x	time.f19.b0.2495
0.897	62.955	25.905	13.463	z	time.f20.b0.5

are written in different styles. So, they will belong to different categories.

Traditional features (i.e. bag of words, word-to-vec etc.) don't keep here as features because our system does not worried about the meaning or content of the text. It is more interested about its complexity and structure. We use Orange [11] as a classification tool for our task. Table 4 shows a sample of classifier (orange) training data along with the features for each file against the class (stack size). And Table 5 represents the sample of classifier (orange) training data along with the features of each file against the class (beam threshold).

We keep ten classes from stack size and three classes from beam threshold for our classification model, as discussed in Section II-C. These classes form the basis of our classification model: stack size 100 is represented by "class a", 150 is represented by "class b", 200 is represented by "class c" and so on, up to stack size 550 which is represented by "class j". Beam threshold value 0.2495 is represented by "class x"; 0.05 by "class y" and 0.5 by "class z". These values can be combined in any way to create a new class; such as "class ax", "class ay", "class az", "class bx", by, and so on, up-to "class jz". So, now we have a total of $10 \times 3 = 30$ classes of parameters. When the given text file is classified it can belong to any of the 30 classes. But since we are doing the classification separately we have to choose only from $10 + 3 = 13$ classes, so time complexity is decreased.

C. STEP BY STEP PROCEDURE

In this section, we describe empirical steps of our proposed method most simply. Also, we discuss the detailed experiment step by step here:

1) EMPIRICAL STEPS

Three steps have been encapsulated to accomplish the whole task. The first phase is devoted to make training data of the classifier. This training data is nothing but various features of the source text, and their corresponding stack size and beam threshold values (which is treated as a class). This helps to achieve best-translated output in minimal time. There may have various translated outputs for different stack size and

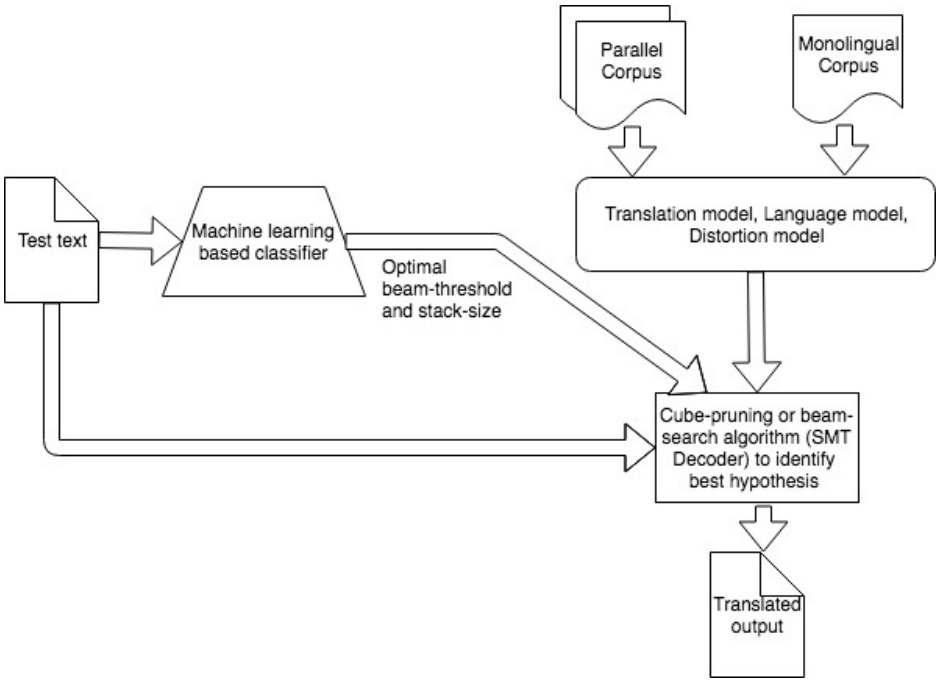

FIGURE 2. Overall architecture for decoding.

beam threshold. In training data, only best-translated output and their corresponding class will get a chance to be placed.

The second phase is the control and the responsibility go to the classifier. When a source text is submitted for translation, classifier chooses the best class (stack size and beam threshold values) for decoding.

In the final phase, the selected stack size and beam threshold values are forwarded to SMT decoder. SMT decoder has used these values for pruning out the hypotheses which are comparatively less sensitive to get better-translated output in optimal time. The compact procedure for the machine learning based SMT decoding is shown in Figure 2.

2) DETAILED EXPERIMENT

We decoded the first ten files to get a basic understanding on how exactly changes in stack size and beam threshold affect decoding time and accuracy. We used the following values of these parameters. stack size: 10,50,100.150,200,250,300,350,400,450,500, 550, 600, 650, 700, 750,800,850,900,950,1000 beam threshold: 0.001, 0.005, 0.01, 0.025, 0.05, 0.1, 0.2495, 0.5

We decided to use these particular values in our experiment after going through several threads of Moses-support mailing archives.[3] Also, as mentioned previously, these values are selected after numerous experiments on various test files. For the sake of simplicity, the range of values for stack size was very high. We took them at a distance of 50 each. Beam threshold values were selected experimentally.

[3]https://www.mail-archive.com/moses-support@mit.edu/msg02224.html

Table 6 (file f.en.6) shows the changes in decoding time and translation accuracy as both the parameters are varied. The accuracy of translation has increased to stack size 200, after which it remains constant. The stack sizes 600 to 1000 take a considerable amount of time to find the best translation, but there is no change in accuracy. This fact was consistent with all our files; sometimes the accuracy even decreased as the stack size was increased. As in Table 1 (for file f.en.6). On the other hand, in the case of beam threshold in Table 6, the changes in time is not much till beam threshold 0.1 but as it reaches beam threshold 0.5 a considerable decrease in decoding time is observed and a surprising bump in translation accuracy. In almost all our experiments on different files, we recorded the second lowest decoding time (combining all 31 observations; 21 observations from stack size and 10 from beam threshold) after stack size 10 where time is lowest. But this fact is not consistent with all the files, as in f.en.2 of Table 3 the decoding time has decreased noticeably so in the case for the translation accuracy. The substantial decrease in decoding time for beam threshold value 0.5 was consistent with all our files.

After running the decoder through the ten files chosen earlier for all values mentioned above of stack size and beam threshold we discovered that:

- stack size 10 and 50 had a less decoding time, but the translation accuracy was also quite low. So, we omit these values for further study. Now coming to stack sizes 600 to 1000, which are not helpful at all. For these stack sizes, not only the decoding time increased but the translation accuracy was the same as earlier, or it decreased.

TABLE 6. Changes in decoding time and translation accuracy as both the parameters (stack size, beam threshold) are varied.

Default Beam Threshold		
stack size	Decoding Time	BLEU
10	31.373	15.93
50	88.021	16.24
100	163.25	16.21
150	241.859	16.19
200	325.121	16.26
250	403.105	16.26
300	494.107	16.26
350	578.188	16.26
400	665.816	16.26
450	760.736	16.26
500	838.517	16.26
550	934.993	16.26
600	1020.356	16.26
650	1141.865	16.26
700	1215.647	16.26
750	1307.748	16.26
800	1404.757	16.26
850	1494.831	16.26
900	1600.286	16.26
950	1674.808	16.26
1000	1753.629	16.26

Default stack size		
Default stack size	Decoding Time	BLEU
0.001	322.479	16.26
0.005	319.783	16.26
0.01	321.476	16.26
0.025	318.722	16.26
0.05	320.311	16.26
0.1	321.235	16.26
0.2495	312.76	16.28
0.5	198.962	16.28

There was no case of an increase in the accuracy of the translated text. After various experiments, we observed that the stack size 100 to 550 showed some desirable changes in BLEU score and decoding time. For example, in table 6 the translation accuracy increases till value 50 and drops for 100 and 150 and again rises for stack size value 200. And 200 onwards it became constant.

- Beam threshold values 0.00000001f, b0.00001f, 0.001, 0.005, 0.01, 0.025,0.1 did not do any good either. There were slight changes in decoding time and no change in BLEU score. So, we took values 0.5, 0.05 and 0.2495 as it is proved by experiment to be quite helpful.

Now the decoding data for the above mentioned useful parameters (stack size 100 to 550 and beam threshold 0.5, 0.05 and 0.2495) are processed as follows. The data are sorted according to decreasing BLEU and increasing the decoding time for each of the fifty-nine files. After sorting the case with highest BLEU and minimum time (that is the leading case) of each file is used to create our training dataset for our classification problem. This means that we now know which file belongs to which class in training dataset. The classes for files f.en.11 to f.en.23 are mentioned in Table 7 along with their number of words and number of sentences. This data along with features as mentioned above has been used to create the training data for classification for all the 59 files. The classification task is trained using this limited amount of data due to low resource. Incorporating more data as training set may improve the performance of the classifier. It will improve the final translation accuracy.

Now the features namely ''Percentage of the comma'', ''Percentage of short sentences'', ''Percentage of stop words'' and ''average words per line'' in a file; as discussed in the previous section are added to the data set for training as Table 4 and Table 5. The procedure for the on three other language pairs: Hindi to Konkani, Hindi to Marathi, Bengali to Hindi was the same except for the classifier training and testing dataset were 20 files and 4 files each respectively, as mentioned in Section IV-A. Rest of the procedure was the same.

TABLE 7. Class for files along with file statistics.

Total number of sentences	Total number of words	Class for stack size	Class for beam threshold
338	4787	g	x
920	13529	f	y
1142	16291	b	z
734	10740	e	z
864	11673	a	x
698	9745	b	x
1130	18622	c	y
1218	16197	e	x
1297	17462	a	z
697	10058	b	x
542	7255	a	x
761	8829	d	x

III. MACHINE LEARNING BASED CUBE PRUNING

Efficient decoding is an important problem in machine translation, especially with an integrated language model, which is essential for achieving good translation quality. Cube pruning is a quick and better technique than traditional decoding. It is quite similar to A* search on a specific search space with specific heuristics. There can be different ways for the decoder to act upon the data. Earlier in beam search, the language model is also considered at the time of decoding and so it becomes computationally more expensive and also takes more time. But another way is that we first decode without traditional language model (henceforth -LM decoding) to produce a k-best list of candidate translations, and then re-rank the k-best list using the language model (LM). This method runs faster in practice but often produces a considerable number of

search errors since the true best translation (taking LM into account) is often outside of the k-best list. Here, cube pruning becomes useful. In cube pruning [8], [23], a compromise between re-scoring and full-integration is done; it re-scores k sub-translations at each node of the forest, rather than only at the root node. It reduces the search space considerably. When the above method is combined with beam search, only a small fraction of the possible +LM items at a node will escape being pruned. Moreover, we can select with reasonable accuracy those top-k items without computing all possible items first. In a nutshell, cube pruning works on the -LM forest, keeping at most k +LM items at each node, and uses the k-best [22] parsing algorithm to speed up the computation.

A. DECODER PARAMETERS FOR CUBE PRUNING TECHNIQUE

The task of the decoder is to search for the best possible translation in line with the earlier trained model. As the number of possible translation can be very large; it is not possible to go through the entire search space in a suitable amount of time. So, decoding algorithms try to prune out (or remove) poor hypotheses which are not fit to be included in the final translation. There are various ways to do this pruning. Here, we discuss cube pruning algorithm, what kind of parameters it has and how they do affect the decoding process.

Even after applying various search algorithms and pruning techniques, the process still takes a lot of time, and there is always a trade-off between accuracy and time. But our experiments have shown that choosing the right values for the parameters according to the input improves the current (default) conditions. This improvement occurs in terms of a decrease in decoding time and increase in translation accuracy as discussed earlier.

The different text has a different sentence structure. We propose a general architecture that different input, if decoded with the suitable values of parameters (according to input), then there would be better translations with less decoding time. It could be incorporated with any decoding technique to fix the parameter. So, it is very important what values of parameters should be selected for a particular input and which parameter's values need to change over various input text.

We decoded various text files for certain values of cube-pruning pop limit, beam threshold, stack size and along with help from moses support mail archives, and experimentally discovered the following:

a. The range of cube-pruning pop limit is 500 to 50,000.

b. The range of beam threshold is from an unknown but negligible value to 0.5.

c. The range of stack size is 10 to 1000.

d. Our experiments results showed that cube-pruning pop limit and beam threshold didn't have any suitable effect on the decoder, but changing stack size did improve the situation.

e. stack size 10 and 50 had really low decoding time, but the translation accuracy was also quite low. So, we omit these values for further study. Now coming to stack sizes 550 to 1000 were of no help either, here not only the decoding time increased but the translation accuracy was the same as earlier, or it decreased. There was no case of an increase of accuracy of the translated text. After various experiments, we observed that the stack size 100 to 500 showed some desirable changes in BLEU score and decoding time. So, we choose only stack size as the determining parameter in our phrase-based machine translation using cube pruning.

IV. DATASET AND EXPERIMENTAL SETUP

In this section, we describe the used corpora, the details of software and hardware setup here.

A. CORPORA AND DATASETS

We have used HindEnCorp [3] for training and testing a machine translation model. The parallel corpora had over 2.7 lakh sentences from various domains, this data was split as follows: first sentences 1 to 1001 tuning set; sentence number 1002 to 51002 split into 59 files containing multiple different number of sentences (anything between 200 to 1200) to make test set; and then 223880 sentences training set for Moses. These 59 files were decoded as described in Section II-C to form training and testing data for Orange classification. These files were named f.en.1, f.en.2 ..., f.en.59 serially. The first ten files were selected for observing the changes in time, and translation accuracy for choosing the important values of parameters stack size and beam threshold, described in Section II-C.

Apart from these, we have implemented the model on three other language pairs: Hindi to Konkani, Hindi to Marathi, Bengali to Hindi. For this, we used the multilingual ILCI corpus [24] containing sentences from tourism and health domains. This corpus was split as lines 1 to 1001 tuning set; 1002 to 11002 splits into 24 files containing 400 sentences each; then 37998 sentences training set for Moses. These files were named in the following format: "f.{language}.{serial no}". So, for Bengali to Hindi translations file names will be f.bn.1, f.bn.2 ... and likewise. The test sets are split in such a manner to record observations and understand the change in decoding time and translation accuracy for different values of the parameters on different files. These observations form the training data for the classification model.

B. HARDWARE AND SOFTWARE SETUP

Decoding such a large number of files on a single computer would take a lot of time. So, the load has been distributed to six machines with the same hardware and software configuration for completing the task in a short span of time. The six computers were desktops with Intel i7 Octacore, clocked @ 3.2 GHz, and Ubuntu 14.04 Operating system. We first decoded the same text file on all of the machines and noted the decoding times for in all cases. We found out that the difference in decoding time was negligible. So, we finally decoded all the files with these selected computers simultaneously.

TABLE 8. Comparison for performance analysis for miscellaneous domain En-Hi translation (HindEncorp) between the traditional stack decoding and machine learning-based stack decoding. ↑, and ↓ represent improvement and deterioration of proposed machine learning based decoding approach over traditional decoding, and ↕ represents a steady situation for both of the cases.

File name	Sentence count	token count	Stack decoding with default parameter		Machine learning based (with beam search) Stack decoding (proposed)			Improvement in	
			Decoding time	BLEU	Value of predicted parameters	Classification time + Decoding time	BLEU	Decoding time	Accuracy
f.en.1	568	7915	343	14.15	-s 100 ; -b 0.2495	171.473	14.12	↑	↓
f.en.2	1170	16769	675.293	16.19	-s 100 ; -b 0.2495	336.473	16.22	↑	↑
f.en.3	1236	16585	581.839	15.88	-s 300 ; -b 0.5	297.111	17.89	↑	↑
f.en.4	1130	15284	348.588	16.02	-s 100 ; -b 0.2495	178.652	16.02	↑	↕
f.en.5	765	9801	234.498	16.22	-s 150; -b 0.5	134.481	16.22	↑	↕
f.en.6	419	6116	1096.096	14.05	-s 100 ; -b 0.2495	556.575	14.05	↑	↕
f.en.7	1397	19938	403.091	12.62	-s 300 ; -b 0.2495	592.539	13.65	↑	↑

We used Moses for stack decoding and Moses2 for cube pruning with Indic NLP library[4] for tokenizing Hindi data. For tokenizing English sentences, we used tokenizer.perl[5] script and giza++[6] for alignment and irstlm[7] to build our 3-gram language model. The classification was done using Orange3 data mining library [11] and python3.

1) HYPER-PARAMETER INFORMATION

NMT system uses following hyper-parameters during the training. Hindi and English vocabulary has been limited to 25K. 128-word embedding and 500 hidden units have been used for this task. Stochastic Gradient Descent was used with learning rate 0.001. We trained our models for a total of 25 epochs with Batch-size 100. We use GPU - NVIDIA 1080Ti, Memory Size - 11 GB, Boost: 1582 MHz / Base: 1480 MHz, CUDA Cores - 3584, GPU Architecture - Pascal, Core Clock - Reference Card, Memory Bus - 352 bit, Card Bus - PCI-E 3.0 x 16, Memory Type - GDDR5X, GPU Architecture - Pascal to train the NMT model with the same training data used for training of SMT system.

V. RESULTS AND ANALYSIS

We have employed BLEU [36] to assess the translation accuracy. Also, the sum of classification time and decoding time (informed by Moses) was used to calculate the time taken for decoding the test files. The results were very satisfactory and consistent with our study for every case. The test files showed a great improvement when the classification model was used. We decoded the test files in two ways:

- Decode using the Moses default values for the parameters.
- And allowing our approach to analyze the text and decide what parameters to take and decode it using those parameters.

[4]https://bitbucket.org/anoopk/indic_nlp_library

[5]https://github.com/moses-smt/mosesdecoder/blob/RELEASE-3.0/scripts/tokenizer/tokenizer.perl

[6]https://github.com/moses-smt/giza-pp

[7]https://hlt-mt.fbk.eu/technologies/irstlm

A. MACHINE LEARNING-BASED STACK DECODING

For stack decoding in phrase-based SMT system, stack size and beam threshold are important parameters. A higher beam threshold makes sure that only hypotheses with good probability scores are included in the best translation. But sometimes some bad hypotheses might be the correct translation of the input file. So here the classifier model helps to decide the right values. According to the observations for English to Hindi translation decoding, six out of first seven files showed a decrease in decoding time by more than 50% and four out of seven files showed an increase in translation accuracy against when they were decoded with Moses default parameters. Also, three files showed a significant decrease in decoding time, about 50% and the translation accuracy remained the same. But for one file the translation accuracy decreased from the default counterparts. This was because our classifier was unable to predict the right class for the input (test set) file. But even with the misclassification, there was a significant decrease in decoding time. Also, the decrease was of only 0.03% which we considered negligible.

Table 8 shows a snapshot of our results as discussed previously. Here, in this table we have presented total decoding time including classification time for proposed machine learning based decoding approach, against decoding time using traditional default parameters which are pointed as baseline decoding with a default parameter. The first column shows the file names for each case. The file names provide us transparency of whole paper. Column with the name "classification time + decoding time" shows the sum of decoding time (informed by Moses) and classification time. Classification time is the time required to read the input file by analyzing its features and classifying it accordingly to a previously trained model. After that, the next column includes the translation accuracy obtained while applying the classification model. As the name suggests "Value of predicted parameters" column shows details of the predicted values for each file. The next column is a separator. The "baseline decoding with default parameters" shows the Decoding time and translation accuracy when default parameters are used in decoding. Finally, ↑, and ↓, and ↕ are used to show performance analysis. ↑ and ↓ represent improvement and

TABLE 9. Performance evaluation of machine learning-based stack decoding with various language pairs.

	NMT system		Stack decoding with default parameter		Machine learning based (with beam search) Stack decoding (proposed)			Performance Improvement over SMT system		Performance Improvement over NMT system	
	Decoding time	BLEU	Decoding time	BLEU	Values of predicted parameter	Classification time+ Decoding time	BLEU	Decoding time	Accuracy	Decoding time	Accuracy
hin-mar	127.124	13.769	119.044	18.8	-s 100 -b 249	40.031	18.8	↑	↕	↑	↑
hin-kon	221.674	16.987	120.751	21.09	-s 100 -b 0.5	11.71	21.09	↑	↕	↑	↑
ben-hin	195.769	13.684	93.723	17.19	-s 300 -b 0.249	120.788	17.29	↓	↑	↑	↑
eng-hin	254.987	13.43	133.073	39.71	-s 250 -b 0.2495	124.874	39.73	↑	↑	↑	↑

deterioration of proposed machine learning based decoding approach over traditional decoding, and ↕ represents a steady situation for both of the cases; training and development set consist of 223880, and 1001 sentence pairs respectively from HindEncorp corpus, as described in Section IV-A. In the case of Bengali to Hindi translation classifier chooses more stack size than default to have better accuracy. So, decoding time has been increased with accuracy. In general all of the cases except the Bengali to Hindi translation of machine learning-based stack decoding, required time decreased for optimally select the parameters. And it is an observation there are three cases of accuracy improvement in this approach.

A. The accuracy increases in most cases which is desirable for our approach due to the selection of effective values of decoder parameters. So, we have a benefit concerning accuracy and time.

B. The steady situation in some cases because the parameters are selected in such a way so that it can provide the decoded output in less time without affecting on translation accuracy. So, we have a benefit with respect to time but no need to compromise with translation accuracy in this case.

C. Deteriorate accuracy in the rare case (one file in whole stack decoding for our experiments) because the parameters are selected in such a way so that it can decode the input text on a very minimal amount of time if it is very hard to get better hypothesis with observed parameters. So, we have a benefit with respect to time and the accuracy deterioration is very negligible. This type of situation is reported at file f.en.1 in Table 8. The decreased accuracy in term of BLEU is 0.03 which is really negligible. All of this scenario was done because our model has been learned such a way to better values of parameters selection. First, this model tried to improve accuracy and after that the time. If unable to improve both then it is devoted to select those parameters values which are responsible for decoding with similar accuracy in less amount of time. If the model is unable to predict the parameters values which are responsible for similar translation accuracy with the default option. Then, at least it will improve the decoding time. Though we do not have any experience in machine learning-based stack decoding approach where translation accuracy and decoding time decrease. But in rare of the rare cases it could be possible due to the wrong classification of parameters because our classification model is not 100% accurate.

To verify our results we decode a given test set (f.en.3 of Table 8) for a range of pruning parameters and compute the corresponding BLEU scores in figure 3. It shows the BLEU score range can vary from 15.04 to 17.89 for this test set. The sets of pruning parameters are ({100, 0.2495}, {100, 0.05}, {100, 0.5}, {150, 0.2495}, {150, 0.05}, {150, 0.5}, {200, 0.2495}, {200, 0.05}, {200, 0.5}, {300, 0.2495}, {300, 0.05}, {300, 0.5}, {350, 0.2495}, {350, 0.05}, {350, 0.5}, {400, 0.2495}, {400, 0.05}, {400, 0.5}, {450, 0.2495}, {450, 0.05}, {450, 0.5}, {500, 0.2495}, {500, 0.05}, {500, 0.5}, {550, 0.2495}, {550, 0.05}, {550, 0.5}). Where, the

TABLE 10. Statistics of various language pairs.

Language pair	Training set	Development set	Test set		Corpora name	Domain name
	Sentence count	Sentence count	Sentence count	Token count		
hin-mar	38998	1000	400	7450	ILCI	Miscellaneous
hin-kon	38998	1000	400	7450	ILCI	Miscellaneous
ben-hin	38998	1000	400	5998	ILCI	Miscellaneous
eng-hin	64724	1001	1002	7229	Launchpad	Technology

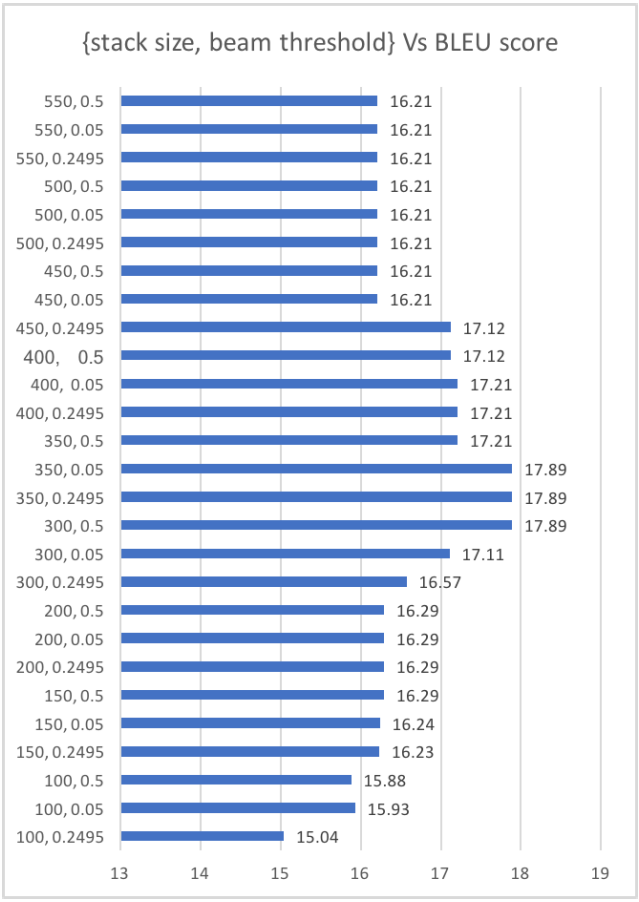

FIGURE 3. The set 300, 0.5 is responsible for better BLEU score in optimal time. The required decoding time for the test set (f.en.3 of Table 8) is 297.111 ms which is exact same as our proposed system.

maximum BLEU score (17.89) is found using three parameter sets only 300, 0.5, 350, 0.2495, 350, 0.05. We achieve the BLEU score 17.89 using our automated ML-based approach which is shown in Table 8. Similar experiments is reported in Figure 4 to validate our system for optimal time. The required decoding times are for these three parameter sets 300, 0.5, 350, 0.2495, 350, 0.05 are 297.111 ms, 335.678 ms and 409.538 ms, respectively. The required decoding time for the test set (f.en.3 of Table 8) is 297.111 ms which is minimum among these three sets.

The proposed model is not restricted for English to Hindi translation. The proposed technique able to produce the more accurate result in less time. Performance analysis for various language pair such as Hindi-Marathi, Hindi-Konkani,

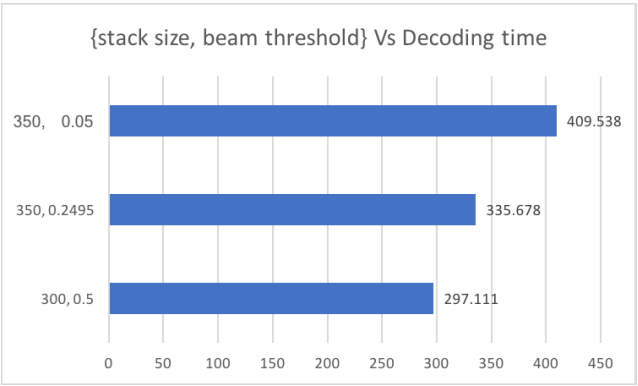

FIGURE 4. Decoding the test set (f.en.3 of Table 8) for a range of pruning parameters and compute the corresponding decoding time which validates our experiments.

Bengali-Hindi is reported in Table 9. It is shown everywhere that the proposed technique does not look harmful for translation accuracy but it improves the decoding time lots. The statistics for this dataset are shown in Table 10. Our demonstrations are mainly for miscellaneous domain data. To show that it works on domain-specific data, technical domain data for eng-hin language pair has used in Table 9. Finally, it shows that the proposed machine learning based decoding provides a significant result.

B. CUBE PRUNING BASED DECODING INCORPORATED WITH MACHINE LEARNING

In this section, we have disclosed detailed results of the cube pruning based decoding with various files which are used for stack decoding and machine learning-based stack decoding in Section V-A for better comparison.

In Table 11, most of the files show an increase in the translation accuracy when our classification model involves for decoding over cube pruning approach, which is a very desirable scenario. But in most cases, this increase in translation accuracy was achieved at a higher stack size than the default resulting in a minimal increase in decoding time. But, compared to the decoding time in beam search, even this increased decoding time can be considered negligible. Unfortunately, due to misclassification for the file f.en.5 there is a decrease in translation accuracy. This happens because the classifier that has predicted some wrong value of the stack size for the files is due to our proposed classifier (CN2), which does not have the same percent accuracy.

TABLE 11. Comparison for performance analysis for miscellaneous domain En-Hi translation (HindEncorp) between cube punning based decoding and cube punning with machine learning-based stack decoding

File name	Sentence count	token count	Cube pruning based decoding with default parameter		Cube pruning with machine learning-based decoding			Improvement in	
			Decoding time	BLEU	Value of predicted parameters	Classification time + Decoding time	BLEU	Decoding time	Accuracy
f.en.1	568	7915	18.0252	19.69	-s 450	27.7457	19.95	↓	↑
f.en.2	1170	16769	27.7229	20.43	-s 450	47.8278	21.42	↓	↑
f.en.3	1236	16585	27.0383	19.94	-s 450	47.1202	20.98	↓	↑
f.en.4	1130	15284	25.9437	21.44	-s 200	25.7279	21.44	↑	↕
f.en.5	765	9801	18.1915	19.96	-s 450	30.2119	19.91	↓	↓
f.en.6	419	6116	13.9378	19.96	-s 100	10.9042	20	↑	↑
f.en.7	1397	19938	31.9013	17.57	-s 300	42.3312	18.94	↓	↑

Source	The network configuration doesn't comply to the ONC standard.
Ref	नेटवर्क कॉन्फ़िगरेशन ONC मानक का पालन नहीं करता।
Ref_T	netavarka koYnPZigareSana ONC mAnaka kA pAlana nahIM karawA.
NMT	नेटवर्क कॉन्फ़िगरेशन नहीं करता है।
NMT_T	netavarka koYnPZigareSana nahIM karawA hE.
NMT+BPE	नेटवर्क कॉन्फ़िगरेशन पालन नहीं करता है।
$\{\text{NMT+BPE}\}_T$	netavarka koYnPZigareSana pAlana nahIM karawA hE.
PBMT	नेटवर्क कॉन्फ़िगरेशन कम्पली नहीं करता है को की गई ONC मानक।
PBMT_T	netavarka koYnPZigareSana kampalI nahIM karawA hE ko kI gaI ONC mAnaka.
ML-based-Beam-PBMT	नेटवर्क कॉन्फ़िगरेशन ओएनसी मानक का की पालन नहीं करता है।
ML-based-Beam-PBMT_T	netavarka koYnPZigareSana oenasI mAnaka kA kI pAlana nahIM karawA hE.
ML-based-Cube-PBMT	नेटवर्क कॉन्फ़िगरेशन ओएनसी मानक का पालन नहीं करता है।
ML-based-Cube-PBMT_T	netavarka koYnPZigareSana oenasI mAnaka kA pAlana nahIM karawA hE.

FIGURE 5. Comparison analysis (using lauchpad dataset) of different English-Hindi MT systems' translated outputs. Where, Source and Ref refer to source sentence and its reference translation. NMT, NMT+BPE, PBMT, ML-based-Beam-PBMT, ML-based-cube-PBMT are neural machine translation system, neural machine translation system with byte pair encoding, phrase-based statistical machine translation system, machine learning based beam search with phrase-based statistical machine translation system, and machine learning based cube pruning approach with phrase-based statistical machine translation system's translated output, respectively. Ref_T, NMT_T, $\{\text{NMT+BPE}\}_T$, PBMT_T, ML-based-Beam-PBMT_T, ML-based-cube-PBMT_T refer to transliteration (in WX-notation) of Ref, NMT, NMT+BPE, PBMT, ML-based-Beam-PBMT, ML-based-cube-PBMT, respectively.

Case-study for English-Hindi (En-Hi) translation is shown in Figure 5. After qualitative analysis, we find that the NMT system is capable to generate extremely good qualitative translated output in terms of fluency but adequacy is worse here for low resource language. To improve its translation quality, byte pair encoding (BPE) [14] is used for rare words with subword units [37] as this is a solution of low resource language for NMT approach. Though the translation quality is improved using BPE but could not transfer exact meaning after translation. We find better adequacy in the translated output generated by phrase-based statistical machine translation (PBMT) system. The proposed ML-based-Beam-PBMT system helps to improve its translation quality in terms of its adequacy and fluency. This is reflected in our case-study. This translated sentence is almost fluent and adequate. Only an extra word (kI) is there. After replacing the beam search algorithm with cube pruning approach in our approach, this extra unwanted word (kI) is omitted and make the translation exactly adequate and fluent for this sentence. The case study for Bengali-Hindi translation (using ILCI dataset) is shown in Figure 6. Though the translated outputs are almost perfect for these cases but there are some morphological, case marking problems and little bit adequacy problem at the translated output of our proposed systems, which may resolve with the described future scope.

TABLE 12. Compact performance analysis among various MT systems in terms of BLEU score and the decoding time for En-Hi translation.

MT Systems	BLEU	Decoding Time
NMT [38]	12.11	658.367
NMT+BPE [37]	13.67	857.578
PBMT [32]	15.88	581.839
ML-based-Beam-PBMT	17.89	297.111
Cube pruning based-PBMT	19.94	27.0383
ML-based-Cube-PBMT	20.98	47.1202

Thus, it is proved that after using our machine learning based decoding model, the performance of the phrase-based decoder improves to a great extent in terms of decrease in decoding time or increase in translation accuracy or both. The compact performance comparisons

Source	চোখের দৃষ্টিতে ভালো - খারাপ দুই ধরনের তত্ত্বই সম্মিলিত থাকে যার প্রভাব নিজের ইচ্ছা মতো যেকোনো জিনিসের উপর ফেলা যেতে পার
Source_T	coKer xqRtiwe BAlo - KArAp xui Xaraner wawwbai sammiliwa WAke yAr praBAb nijer icCA mawo yekono jiniser upar PelA yewe pAra
Ref	नेत्रों की बेधक दृष्टि में भले - बुरे दोनों ही प्रकार के तत्त्व समाहित रहते हैं जिनका प्रभाव मनचाही दिशा में किया जासकता हैं।
Ref_T	newroM kI beXaka xqRti meM Bale - bure xonoM hI prakAra ke wawwva samAhiwa rahawe hEM jinakA praBAva manacAhI xiSA meM kiyA jA sakawA hE
NMT	नेत्रों की दो प्रकार के होते हैं जो किसी भी प्रभावित हो सकते हैं।
NMT_T	newroM kI xo prakAra ke howe hEM jo kisI BI praBAviwa ho sakawe hEM.
{NMT+BPE}	नेत्रों की दो प्रकार के सिद्धांत होते हैं जो किसी भी इच्छा से प्रभावित हो सकते हैं।
$\{\text{NMT+BPE}\}_T$	newroM kI xo prakAra ke sixXAMwa howe hEM jo kisI BI icCA se praBAviwa ho sakawe hEM
PBMT	नेत्रों में अच्छी - बुरे दोनों ही प्रकार के वैसे - वैसे जिसके पू की तरह किताब पहाड़ीनुमा मिले है, किसी पर फेरा जा सकता।
PBMT_T	newroM meM acCI - bure xonoM hI prakAra ke vEse - vEse,jisake pU kI waraha kiwAba pahAdZInumA mile hE , kisI para PerA jA sakawA.
ML-based-Beam-PBMT	नेत्रों दो प्रकार की के अच्छे बुरे सिद्धांत होते हैं जो जो किसी भी तरह इच्छा प्रभावित हो सकते।
ML-based-Beam-PBMT_T	newroM xo prakAra kI ke acCe bure sixXAMwa howe hEM jo jo kisI BI waraha icCA praBAviwa ho sakawe.
ML-based-Cube-PBMT	नेत्रों में दो प्रकार अच्छे बुरे सिद्धांत होते हैं जो किसी भी तरह की इच्छा प्रभावित हो सकते।
ML-based-Cube-PBMT_T	newroM meM xo prakAra acCe bure sixXAMwa howe hEM jo kisI BI waraha kI icCA praBAviwa ho sakawe.

FIGURE 6. Comparison analysis (using ILCI dataset) of different Bengali-Hindi MT systems' translated outputs. Where, Source and Ref refer to source sentence and its reference translation. NMT, NMT+BPE, PBMT, ML-based-Beam-PBMT, ML-based-cube-PBMT are neural machine translation system, neural machine translation system with byte pair encoding, phrase-based statistical machine translation system, machine learning based beam search with phrase-based statistical machine translation system, and machine learning based cube pruning approach with phrase-based statistical machine translation system's translated output, respectively. Source_T,Ref_T, NMT_T, $\{\text{NMT+BPE}\}_T$, PBMT_T, ML-based-Beam-PBMT_T, ML-based-cube-PBMT_T refer to transliteration (in WX-notation) of Source, Ref, NMT, NMT+BPE, PBMT, ML-based-Beam-PBMT, ML-based-cube-PBMT, respectively.

for En-Hi translation using the test file f.en.3 of Table 8 are shown in Table 12.

VI. CONCLUSIONS

We have presented a simple and effective extension to the phrase-based machine translation decoder in Moses by combining the knowledge of machine learning and machine translation to speed up decoding and making the translation as accurate as possible. As per our knowledge, such an approach has never been attempted in the past. We have tried to prove by experiments that the proposed framework has been very effective in decreasing the decoding time or the increasing translation accuracy for various corpora and domains of input. In almost all our observations there has been at least a 50% to about 90% decrease in decoding time with either no decrease or an increase in translation accuracy for stack decoding. As a matter of fact, in most cases, there has been an increase in translation accuracy which shows that validity of our work. Sometimes due to misclassification, wrong results have been predicted but still the decrease in translation accuracy in utmost 0.03% which can be considered negligible. Also in such cases of misclassification, there has been a significant decrease in decoding time. In rare cases (single time in whole stack decoding based experiments) there has been the favorable increase in translation accuracy along with an unwanted increase in decoding time. This is because our machine learning model is trained not to compromise with translation accuracy. So, the cases where the decoding time has increased, the translation accuracy has surely increased. For cube pruning based approach, though in some cases it is required more decoding time (negligible amount), translated accuracy improves significantly. Finally, we can conclude that the machine learning based decoding approach has a significant improvement in term of its accuracy improvement or decoding time improvement or both. We will try to optimize classification methods in the recent future. Here, the classifier could also be based on some features specific to the phrase-based approach like the maximum length of a phrase match, a total number of grammar rules that fire etc. For this, we are planning to identify the optimal features for this task. Due to the limitation of data, the neural network could not be involved here for feature selection. We are also planning to incorporate the neural network for automated features identification in this framework.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. (2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473
[2] N. Bertoldi, B. Haddow, and J.-B. Fouet, "Improved minimum error rate training in moses," in *The Prague Bulletin of Mathematical Linguistics*, vol. 91. Berlin, Germany: Versita, 2009, pp. 7–16.
[3] O. Bojar *et al.*, "Hindencorp-hindi-english and hindi-only corpus for machine translation," in *Proc. LREC*, 2014, pp. 3550–3555.
[4] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks," in *Proc. ISMIR*, 2013, pp. 335–340.

[5] P. F. Brown *et al.*, "A statistical approach to machine translation," *Comput. Linguistics*, vol. 16, no. 2, pp. 79–85, 1990.
[6] W. B. Cavnar and J. M. Trenkle, *N-Gram-Based Text Categorization*, vol. 48113. Ann Arbor, MI, USA: Citeseer, no. 2, 1994, pp. 161–175.
[7] G. Cervone, P. Franzese, and A. P. K. Keesee, "Algorithm quasi-optimal (AQ) learning," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 2, pp. 218–236, 2010.
[8] D. Chiang, "Hierarchical phrase-based translation," *Comput. Linguistics*, vol. 33, no. 2, pp. 201–228, 2007.
[9] P. Clark and R. Boswell, "Rule induction with CN_2: Some recent improvements," in *Proc. Eur. Work. Session Learn.* Berlin, Germany: Springer, 1991, pp. 151–163.
[10] P. Clark and T. Niblett, "The CN_2 induction algorithm," *Mach. Learn.*, vol. 3, no. 4, pp. 261–283, 1989.
[11] J. Demšar *et al.*, "Orange: Data mining toolbox in python," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 2349–2353, 2013.
[12] Y. Feng, H. Mi, Y. Liu, and Q. Liu, "An efficient shift-reduce decoding algorithm for phrased-based machine translation," in *Proc. 23rd Int. Conf. Comput. Linguistics Posters*, 2010, pp. 285–293.
[13] M. Freitag and Y. Al-Onaizan. (2017). "Beam search strategies for neural machine translation." [Online]. Available: https://arxiv.org/abs/1702.01806
[14] P. Gage, "A new algorithm for data compression," *C Users J.*, vol. 12, no. 2, pp. 23–38, 1994.
[15] U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada, "Fast and optimal decoding for machine translation," *Artif. Intell.*, vol. 154, nos. 1–2, pp. 127–143, 2004.
[16] A. Graves. (2012). "Sequence transduction with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1211.3711
[17] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proc. 6th Workshop Stat. Mach. Transl.*, 2011, pp. 187–197.
[18] H. Hoang, N. Bogoychev, L. Schwartz, and M. Junczys-Dowmunt. (2016). "Fast, scalable phrase-based SMT decoding." [Online]. Available: https://arxiv.org/abs/1610.04265
[19] H. Hoang and P. Koehn, "Design of the moses decoder for statistical machine translation," in *Proc. Softw. Eng., Testing, Qual. Assurance Natural Lang. Process.*, 2008, pp. 58–65.
[20] H. Hoang and P. Koehn, "Design of the moses decoder for statistical machine translation," in *Proc. Softw. Eng., Testing, Qual. Assurance Natural Lang. Process.*, 2008, pp. 58–65.
[21] C. R. Huang and L. H. Lee, "Contrastive approach towards text source classification based on top-bag-of-word similarity," in *Proc. PACLIC*, 2008, pp. 404–410.
[22] L. Huang and D. Chiang, "Better k-best parsing," in *Proc. 9th Int. Workshop Parsing Technol.*, 2005, pp. 53–64.
[23] L. Huang and D. Chiang, "Forest rescoring: Faster decoding with integrated language models," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*, 2007, pp. 144–151.
[24] G. N. Jha, "The TDIL program and the Indian langauge corpora intitiative (ILCI)," in *Proc. LREC*, 2010, pp. 982–985.
[25] M. Junczys-Dowmunt, "A space-efficient phrase table implementation using minimal perfect hash functions," in *Proc. Int. Conf. Text, Speech Dialogue*. Berlin, Germany: Springer, 2012, pp. 320–327.
[26] K. Knight, "Decoding complexity in word-replacement translation models," *Comput. Linguistics*, vol. 25, no. 4, pp. 607–615, 1999.
[27] K. Knight, "Decoding complexity in word-replacement translation models," *Comput. Linguistics*, vol. 25, no. 4, pp. 607–615, 1999.
[28] P. Koehn, "Pharaoh: A beam search decoder for phrase-based statistical machine translation models," in *Proc. Conf. Assoc. Mach. Transl. Amer.* Berlin, Germany: Springer, 2004, pp. 115–124.
[29] P. Koehn, *Statistical Machine Translation*. Cambridge, U.K.: Cambridge Univ. Presss, 2009.
[30] P. Koehn *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proc. 45th Annu. Meeting ACL Interact. Poster Demonstration Sessions*, 2007, pp. 177–180.
[31] P. Koehn and R. Knowles. (2017). "Six challenges for neural machine translation." [Online]. Available: https://arxiv.org/abs/1706.03872
[32] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, vol. 1, 2003, pp. 48–54.
[33] A. Kunchukuttan and P. Bhattacharyya. (2016). "Faster decoding for subword level phrase-based SMT between related languages." [Online]. Available: https://arxiv.org/abs/1611.00354
[34] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski, "Subgroup discovery with CN2-SD," *J. Mach. Learn. Res.*, vol. 5, pp. 153–188, Feb. 2004.
[35] Z. Li *et al.*, "Joshua: An open source toolkit for parsing-based machine translation," in *Proc. 4th Workshop Stat. Mach. Transl.*, 2009, pp. 135–139.
[36] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.
[37] R. Sennrich, B. Haddow, and A. Birch. (2015). "Neural machine translation of rare words with subword units." [Online]. Available: https://arxiv.org/abs/1508.07909
[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[39] M. Yasuhara, T. Tanaka, J. Y. Norimatsu, and M. Yamamoto, "An efficient language model using double-array structures," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 222–232.
[40] R. Zens and H. Ney, "Efficient phrase-table representation for machine translation with applications to online MT and speech translation," in *Proc. Hum. Lang. Technol. Conf. North Amer. Assoc. Comput. Linguistics Main Conf.*, 2007, pp. 492–499.
[41] R. Zens, F. J. Och, and H. Ney, "Phrase-based statistical machine translation," in *Proc. Annu. Conf. Artif. Intell.* Berlin, Germany: Springer, 2002, pp. 18–32.
[42] W. Zhang, "Complete anytime beam search," in *Proc. AAAI/IAAI*, 1998, pp. 425–430.

DEBAJYOTY BANIK received the B.Eng. degree from the Bengal Engineering and Science University, Shibpur (IIEST Shibpur), and the M.Tech. degree from the National Institute of Technology Durgapur. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, IIT Patna. His research interests include machine translation, reversible computing, quantum dot cellular automata, and VLSI design and testing.

ASIF EKBAL was a Post-Doctoral Research Fellow at the University of Trento, Italy, and Heidelberg University, Germany. He is currently the Dean of the R&D, IIT Patna, India, where he is an Associate Professor with the Department of Computer Science and Engineering. His current research interests include natural language processing, information extraction, machine learning applications, opinion mining, and text mining. In these areas, he has authored or co-authored around 100 papers in the journals like ACM TALIP, knowledge-based systems, and knowledge engineering, and conferences like ACL, COLING, EACL, IJCNLP, and ECAI. Google scholar citation, which is the benchmark of computer science, shows his citation count of 1593 with h5-index of 22. He is a recipient of the Best Innovative Project Award from the Indian National Academy of Engineering, the JSPS Invitation Fellowship from the Government of Japan, and the Visesvaraya Young Faculty Research Fellowship Award from the Government of India. His citation count is over 1.5K.

PUSHPAK BHATTACHARYYA was the Past President of the Association for the Computational Linguistics from 2016 to 2017, and the Ex-Vijay and Sita Vashee Chair Professor. He is a Computer Scientist and a Professor with the Computer Science and Engineering Department, IIT Bombay. He is the Director of IIT Patna. He currently heads the Natural language Processing Research Group Center, Indian Language Technology Lab, IIT Bombay. He is a well-known author in the field of the machine translation, natural language processing, machine learning, and artificial intelligence. His citation count is over 4.5K.