Towards efficient Neural Machine Translation for Indian Languages

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering
By Research

by

Ruchit Agrawal
201002013
`ruchit.agrawal@research.iiit.ac.in`

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
October 2017

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Neural Machine Translation for Indian languages - Techniques and Challenges" by Ruchit Agrawal, has been carried out under my supervision and is not submitted elsewhere for a degree.

27/10/2017
Date

[illegible]
Adviser: Dr. Dipti Misra Sharma

To

Our Library

Acknowledgments

This thesis is a confluence of many hands and minds. I would like to express my gratitude to my thesis advisor Dr. Dipti Misra Sharma. She contributed not only towards my academic growth, but towards an all round development across all areas of life. She was there to support me at all stages during my tenure as a research student at IIIT Hyderabad.

I would like to thank all other faculty members : Dr. Radhika Mamidi, Dr. Manish Shrivastava, Dr. Suryakant V. Gangashetty, Dr. R Govindrajulu and Dr. Kannan Srinathan to name a few, who contributed to my growth - not necessarily by guiding me in my research, but just being there as a person to look up to - be it academically or otherwise.

I express my gratitude to the Dean and the Director for promoting a research environment and working tirelessly towards ensuring developing an impeccable infrastructure for the same.

A special mention goes to Mihir Shekhar, for collaborating with me on a cross-domain topic. This thesis would not have been possible without the interaction with my colleagues at LTRC and my friends from other research centers : Vignesh, Nikhil, Ratish, Irshad, Pruthwik, Saumitra, Nikhilesh, Silpa, Pramod, Bhargav, Yashwaswi to name a few.

Lastly, I would like to thank the staff at IIIT for making my stay here a valuable experience. Dinesh Bhaiya, Mr. Ramana, Mr. Y Kishore, mess staff and other staff members also played a significant role in ensuring a peaceful and healthy atmosphere in the campus.

Abstract

Machine Translation among Indian languages is a challenging problem, owing to multiple factors like their morphological complexity and diversity, in addition to lack of sufficient parallel data for most language pairs. Recent advances in the past have employed rule-based and statistical techniques to approach the problem of Indian language MT. Neural Machine Translation is an emerging technique depicting impressive performance, better than traditional MT methods in multiple aspects. This thesis demonstrates the application of Neural Machine Translation (NMT) techniques for Indian languages, with an emphasis in two important directions:-

1. Usage of specific linguistic features belonging to Indian languages to improve translation quality.
2. Building a robust NMT model which delivers efficient performance across different domains with a limited parallel corpus.

We create NMT systems for 110 Indian language pairs utilizing various morphological and syntactic features to improve translation quality. We observe that although NMT models have a strong efficacy to learn language constructs, the usage of specific features further help in improving the performance. We also propose a three-phase integrated approach which helps in improving robustness across domains as well as translation quality in the absence of large parallel corpora. The three-phase training shows a significant improvement in accuracy as well as coverage over a baseline NMT model.

This is the first effort towards developing Neural Machine Translation for Indian languages to the best of our knowledge.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter *1*

Introduction and Related Work

This chapter lays out a brief introduction to the thesis, touching upon four main aspects - the significance of Indian Language MT and its challenging nature, related work done so far in the field, our motivation behind employing Neural Machine Translation techniques for this task and finally our contributions to the field of Indian Language MT.

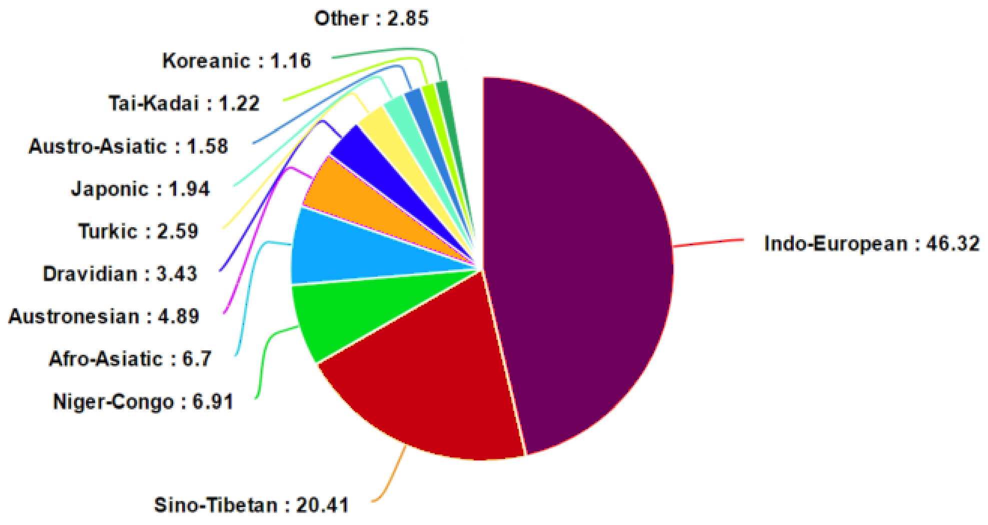
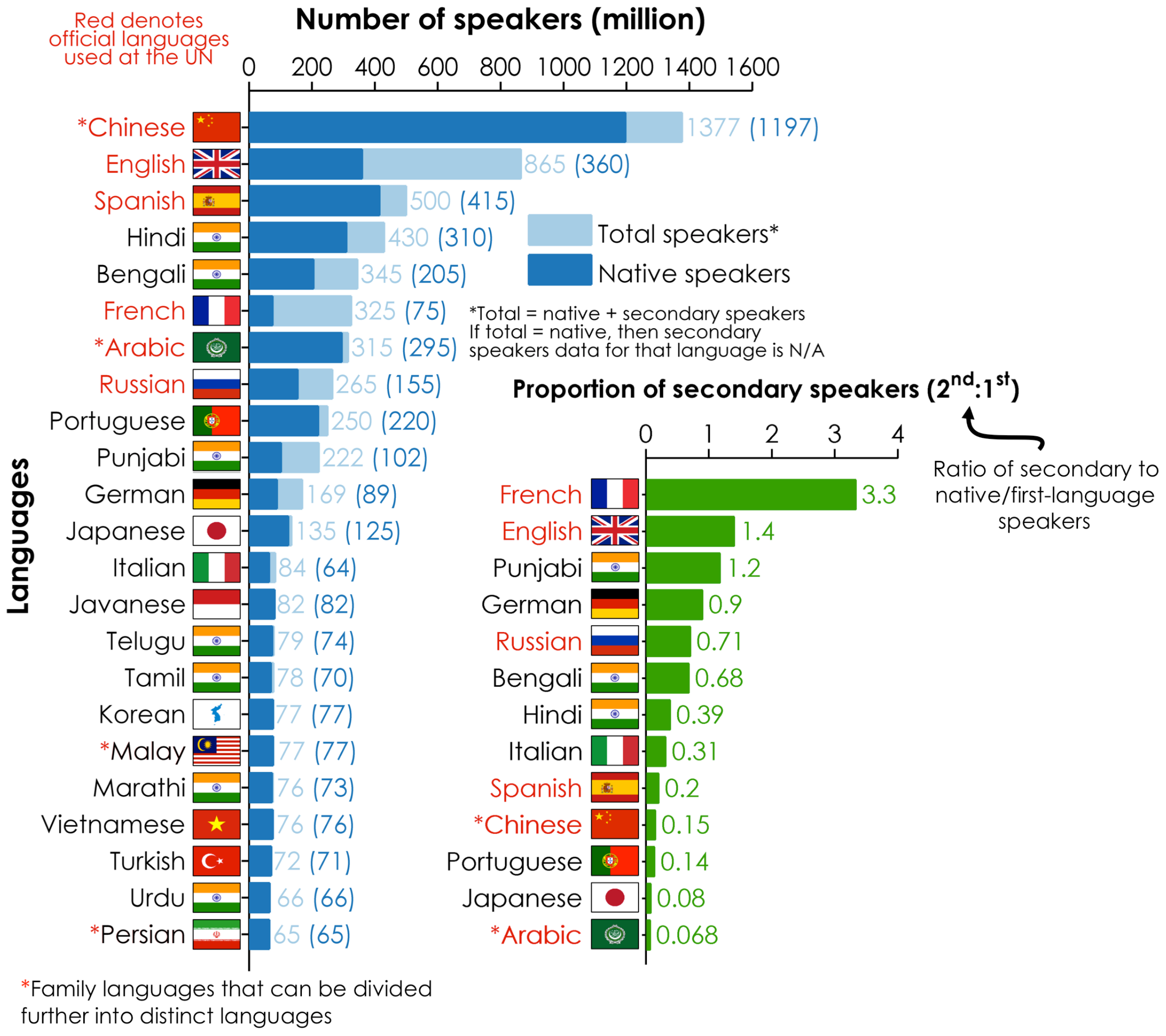


Figure 1.1 *Indo-European languages occupy a major portion of world language families. Statistics from https://www.ethnologue.com/statistics/family*

1.1 Machine Translation among Indian Languages - Significance and Challenges

1.1.1 Significance of Indian Language MT

Indian languages have a prominent presence in the languages spoken across the world, both in terms of the linguistic characteristics as well as the socio-cultural aspects. Figure 1.1 shows the language families of the world according to the percentage of speakers.



Language	Total speakers (Native speakers), million
*Chinese	1377 (1197)
English	865 (360)
Spanish	500 (415)
Hindi	430 (310)
Bengali	345 (205)
French	325 (75)
*Arabic	315 (295)
Russian	265 (155)
Portuguese	250 (220)
Punjabi	222 (102)
German	169 (89)
Japanese	135 (125)
Italian	84 (64)
Javanese	82 (82)
Telugu	79 (74)
Tamil	78 (70)
Korean	77 (77)
*Malay	77 (77)
Marathi	76 (73)
Vietnamese	76 (76)
Turkish	72 (71)
Urdu	66 (66)
*Persian	65 (65)

Language	Proportion of secondary speakers (2nd:1st)
French	3.3
English	1.4
Punjabi	1.2
German	0.9
Russian	0.71
Bengali	0.68
Hindi	0.39
Italian	0.31
Spanish	0.2
*Chinese	0.15
Portuguese	0.14
Japanese	0.08
*Arabic	0.068

Figure 1.2 *Most spoken languages of the world. Image Source : https://goo.gl/vt0ahm*

It can be noted that the Indo-European languages make up the largest percentage among the world language families. Indian languages further form a significant portion of the Indo-European languages. Figure 1.2 lays out a chart depicting the most common languages in the world. It can be observed that three among the top ten most spoken languages and seven among the top 25 most spoken languages are Indian languages.

Apart from the major Indian languages, there are a multitude of dialects spoken across the country, employing different scripts and belonging to different typologies. With the presence of such a large number of languages with diverse characteristics, communication across different linguistic groups can be facilitated to a great extent with the help of the technology of Machine Translation. Owing to the above factors, there is a lot of scope for work which can be done in the direction of Indian language MT - a field with many potential applications in domains like education, business, government, tourism, communication and so on.

1.1.2 Characteristics of Indian languages

As stated above, Indian languages are extremely diverse, belonging to various language families, employing various scripts and spanning across a multitude of dialects. They are broadly classified into the following language families : Indo-Aryan languages, Dravidian languages, Austroasiatic languages and Munda languages. For the purpose of this study, we choose some of the Indian languages, which are described in Table 1.1 along with their ISO codes. The ISO codes are used to denote the languages in the tables and graphs in further chapters to maintain brevity.

Table 1.1 ISO-639-2 codes and language families for Indian languages. **IA** : Indo-Aryan, **DR** : Dravidian, **IE** : Indo-European

Hindi	**hin**	**IA**	Gujarati	**guj**	**IA**
Urdu	**urd**	**IA**	Marathi	**mar**	**IA**
Punjabi	**pun**	**IA**	Konkani	**kon**	**IA**
Bengali	**ben**	**IA**	Tamil	**tam**	**DR**
Telugu	**tel**	**DR**	Malayalam	**mal**	**DR**
English	**eng**	**IE**			

The majority of Indian languages are morphologically rich and depict unique characteristics, which are significantly different from languages such as English. Some of these characteristics are the relatively free word-order with a tendency towards the Subject-Object-Verb (SOV) construction, a high degree of inflection, usage of reduplication, converbs, relative participial forms and correlative clause constructions. These unique characteristics coupled with the caveats of evaluation metrics described in Section 1.1.3 pose interesting challenges to the field of Indian Language MT - both in terms of development of efficient systems as well as their evaluation.

Table 1.2 Different Hindi translations corresponding to the English sentence - "Shyam has given the book to Manish." (Due to word order)

	Hindi	Transliteration
Sent : 1	मनीष को श्याम ने किताब दे दी ।	maneesh ko shyaam ne kitaab de dee
Sent : 2	श्याम ने मनीष को किताब दे दी ।	shyaam ne maneesh ko kitaab de dee

For example, in Hindi, a sentence s containing the words w_1,w_2,..,w_n can be formulated with multiple variants of word ordering. This behavior is depicted in Table 1.2, which shows two Hindi translations of the following English sentence :
'Shyam has given the book to Manish.' Although they use different word-order, both of them are semantically equivalent and correct translations of the source sentence.
Similarly, for the sentence 'The sun has set', there can be multiple valid translations, as shown in Table 1.3. *(We use the WX notation* [1] *for transliteration. This is used further throughout the paper).* It can be noted that 'सूर्य ' and 'सूरज ' are synonyms of 'Sun' in Hindi.

Table 1.3 Two different translations corresponding to the English sentence - "The sun has set." (Due to many-to-many mapping between vocabulary)

	Hindi	Transliteration
Sent : 1	सूर्य डूब चुका है ।	soorya doob chuka hai
Sent : 2	सूरज डूब चुका है ।	sooraj doob chuka hai

In addition to these, there are many subtle differences in the ways different Indian languages encode information. For example, Hindi has two genders for nouns whereas Gujarati has three. There are also many ambiguities introduced in language (both at lexical as well as sentence levels) due to the socio-cultural reasons and partial encoding of information in discourse scenario. In addition to this, the majority of Indian languages encode a significant amount of linguistic information in their rich morphological structures, and often lexemes can have multiple senses. All these factors like linguistic conventions, socio-cultural knowledge, context and highly inflectional morphology combined together with the lack of resources make Indian languages a challenging terrain for Machine Translation.

1.1.3 Caveats of evaluation metrics

A key aspect in developing efficient MT systems is addressing the issue of effective metrics for automatic evalution of translations, since manual evaluation is expensive and time-consuming. There has been significant interest in this area, both in terms of development as well as evaluation of MT metrics. The Workshop on Statistical Machine Translation (Callison-Burch et

[1] *http://bit.ly/2pDgALn*

al., 2007; Callison-Burch et al., 2008; Callison- Burch et al., 2009) and the NIST Metrics for Machine Translation 2008 Evaluation 1 have both collected human judgement data to evaluate a wide spectrum of metrics. However, the problem of reordering has not been addressed much so far. The primary evalutaion metrics which exist currently for scoring translations are BLEU, METEOR, RIBES and NIST.
BLEU [44] measures the number of overlapping n-grams in a given translation when compared to a reference translation, giving higher scores to sequential words. METEOR [35] scores translations using alignments based on exact, stem, synonym, and paraphrase matches between words and phrases[2]. RIBES [25] is based on rank correlation coefficients modified with precision. NIST [18] is a variation of BLEU; where instead of treating all n-grams equally, weightage is given on how informative a particular n-gram is. We report the BLEU score as a measure to test accuracy for the 110 NMT systems to maintain brevity. However, for the language-pair English -> Hindi; we report all of the above scores. We also describe the challenges in evaluating MT accuracy keeping this language pair in consideration, however it should be noted that the same or similar challenges are faced when dealing with other language pairs as well. We use the MT-Eval Toolkit[3] to calculate all these metrics.
It can be noted that most of the above-mentioned metrics employ some concept of word-order as well as word similarity using n-grams to score translations, which makes evaluating Hindi translations a tedious task. In addition to this, there exists a many-to-many mapping of vocabulary between English and Hindi which makes all of these scoring mechanisms less effective. For example, both translations shown in Table 1.3 are valid. However; since the current MT metrics rely heavily on lexical choice, there is no mechanism which takes into account the phenomena described above, which is which is quite common in Indic languages like Hindi. Hence, in addition to the metric scores, we also show sample examples with their descriptions in the following section, in order to demonstrate translation quality in a more comprehensive manner. These are described in detail in Chapter 5.

1.2 Related Work

Indian language MT is a challenging problem, owing to multiple reasons including morphological complexity and diversity, in addition to a lack of resources for many languages. Advances in the recent past mainly employ statistical and rule based methods for MT. Some of these systems include [2], [34], [1], [20] and [7].[34] uses statistical phrase based machine translation for Indian Languages using Moses [32] for phrase extraction as well as lexicalized reordering. Sampark [2] is a transfer based system for translation between 18 Indian language pairs, which uses a common lexical transfer engine, whereas minimum structural transfer is

[2]We have tuned the Meteor to score English - Hindi Translation using parallel corpus.
[3]http://bit.ly/2p5C2FB

required between Indian languages. [33] use orthographic features along with SMT to reach state of the art results in SMT for related languages.
This thesis is novel in the application of NMT techniques for translation among Indian languages. Since the majority of Indian languages lack large parallel corpora, we we have worked on building effective NMT models inspite of low training corpora. To achieve this, we propose the use of monolingual corpus to create synthetic data. The use of monolingual data to improve translation accuracy in NMT was first proposed by [22]. Monolingual models were trained independently and then were integrated to decoder module either through rescoring of the beam (shallow fusion), or by adding the recurrent hidden state of the language model to the decoder state of the encoder-decoder network, with an additional controller mechanism that controls the magnitude of the LM signal (deep fusion).
[24] shows use of SMT features improved NMT. [51] proposed use of synthetic data, a parallel data corpus generated using back-translation along with parallel corpus to increase the translation accuracy.

Our method differs from them since it is three-phased. In the first phase, we train our model over a synthetic corpus generated using a suboptimal MT technique, and then fine tune it further on gold data. This allows better control over training during various stages - leading to better translation quality for Indian languages. Our second phase is inspired from [62]. They use transfer learning to increase translation quality between resource scarce language pairs by incorporating the weights learnt during training for high resource language pairs. It was also found that languages having similar structure, like Fr $\longleftrightarrow$ En (French - English) showed better improvement in performance as compared to other languages having little similarity, like Uz $\longleftrightarrow$ En (Uzbek - English). Our approach is based on the intuition that transfer learning between the same language pair should perform better than its multilingual counterpart. The experimental results described in Chapter 5 demonstrate that the above intuition stands correct. During fine-tuning, the change in weights in each epoch learnt through transfer learning allows the model to align more towards the correct model.
[42] proposed using self-training for the task of parsing. We have experimented with its use in Neural machine translation.

1.3 Motivation

As mentioned in Section 1.2, MT for Indian languages has been explored using either rule-based or statistical approaches in the past. In this thesis, we present the application of Neural Machine Translation techniques to deal with the challenging problem of MT among Indian languages. Neural Machine Translation (NMT) [55, 13, 4] has shown promising results for various language pairs and is an emerging alternative to phrase-based Statistical Machine Translation

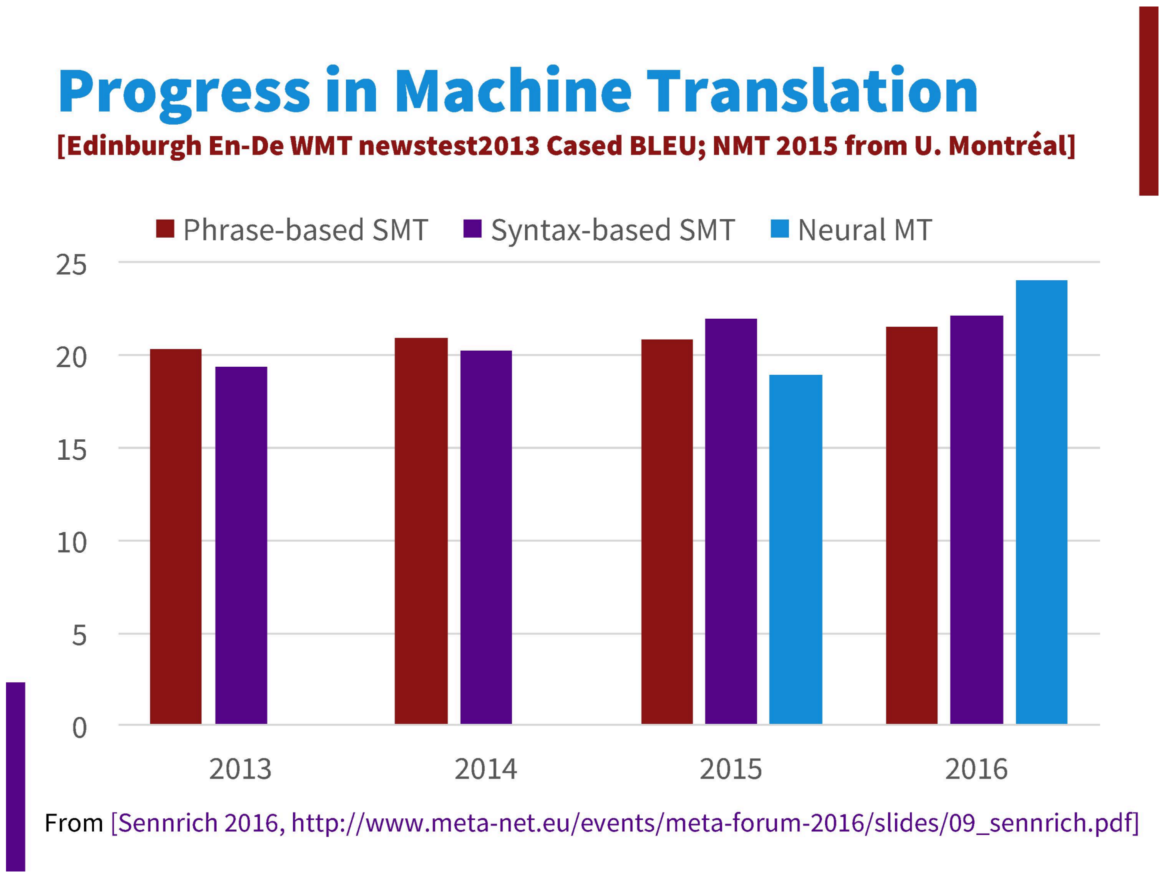


Figure 1.3 *Progress in performance of Machine Translation in recent years*

(SMT). The primary appeal of NMT lies in its ability to employ algorithms which learn linguistic rules on their own from the parallel corpus, thus making it conceptually simple and eliminating the need for complex feature engineering by providing end-to-end translation. Since the training is done end-to-end, *all* of the training parameters are optimized simultaneously to achieve a minimum loss function on the output of the neural network, thereby removing the need of having separate components (like lexicalized re-ordering model, distortion model, transliteration model, and so on) during the intermediate stages of the translation process. The end-to-end nature of the training phase is also very conducive when dealing with sequences of unknown lengths beforehand, thereby making neural models an appropriate choice for other tasks like chatbots, speech recognition, dialogue systems, time series, question-answering and image captioning as well. The distributed representations in the form of dense vectors employed by NMT enable the sharing of statistical strength between similar words and similar phrases and better predictions can be obtained by exploiting these similarities. NMT systems are also adept at exploiting the context information as opposed to traditional n-gram models, where the language models on the target side incorporate a very short context. The context incorporation is made easier due to the distributed representations mentioned above, unlike traditional systems where one-hot representations of words is more commonly used, resulting in data sparsity and thereby inability to incorporate a large context window. Due to all the above-mentioned reasons, NMT generates more fluent translation as compared to phrase based SMT systems especially on lexically rich texts. Furthermore, it eliminates the need for complex feature engineering by providing end-to-end translation. Bentivogli [6] demonstrates that NMT output contains lesser lexical errors (-17%), lesser morphology errors (-19%), and significantly lesser word order errors (-50%) than its closest competitor MT paradigms for each error type. NMT systems have achieved competitive accuracy scores under large-data training conditions for language pairs such as En $\rightarrow$ Fr (English - French) and En $\rightarrow$ De (English - German) [59]. However, on the other hand, NMT models are unable to extract sufficient language constructs like morphology, syntax and word semantics in low resource scenario.

The majority of Indian languages depict a high degree of agglutination and rich morphology. These factors coupled with unavailability of a large parallel corpora makes translation among Indian languages especially challenging. Also, due to the unavailability of large parallel corpora, the vocabulary size tends to be low, due to which any word which is not included in the vocabulary is mapped to a special token representing an unknown word $[UNK]$, also called as out-of-vocabulary word (OOV). This causes a large number of OOV's in the target sentence, which results in a drastic drop in the translation quality. Also, current NMT models do not make employ linguistic information like explicit syntax / semantics directly. Both of these issues are addressed to a certain extent in our proposed approaches, which are described in detail in Chapter 4.

1.4 Contributions

In this thesis, we explore different techniques to build robust and accurate NMT systems for Indian languages. The proposed techniques do not require any modification to the underlying neural network architecture during various training phases. Hence, we believe it can be applied to other languages which have a traditional Machine Translation system and a small parallel corpus but lack in availability of large parallel corpora with little to no modifications. The proposed methods could serve as a baseline for further improving NMT for Indian languages in the future.
The major contributions of this thesis are summarized below:-

- We propose a method which employs linguistic features specific to Indian languages to improve translation quality by leveraging their shared characteristics. We create NMT systems for 110 language pairs using this method and compare their performance with state-of-the-art phrase-based SMT systems trained over the same corpus.
- We propose a technique for synthetic data generation using an existing MT technique and a large monolingual corpus of the source language.
- We propose a three-phase approach which employs synthetic data to improve translation quality . We show results comparable to the state-of-the-art for 10 language pairs in terms of accuracy as well as domain coverage using this approach.
- We compare the effect of weak supervision and semi-supervision across various language pairs with relation to domain coverage and accuracy.
- We propose a method to deal with the problem of Out-of-Vocabulary (OOV) words.

1.5 Thesis Organization

The rest of this thesis is organized as follows: Chapter 3 gives the details of our NMT architecture. It also describes the datasets and resources employed in our experiments. Chapter 4 gives a detailed explanation of our proposed methods and the experiments conducted using them. A summary of the methods is also provided at the end of this chapter. We report the results obtained by our models and analyse them in Chapter 5. We conclude the paper and discuss future work in Chapter 6.

Chapter *2*

Preliminary - Neural Machine Translation

In this chapter, we lay out a concise description of the theoretical background needed to understand the thesis in depth. The thesis proposes the use of Neural Machine Translation techniques for the task of Indian Language MT. Neural Machine Translation (NMT) is a novel approach to MT which utilizes deep neural networks to generate end-to-end translation. The theoretical background behind NMT is described in the following sections. We begin with a brief introduction to NMT, followed by a comparison with statistical machine translation, followed by a description of Recurrent Neural Networks and end with a description of the attention mechanism - a recent but valuable addition to NMT.

2.1 Neural Machine Translation - Introduction and Brief History

2.1.1 The Basic Framework

Artificial Neural Networks are an inevitable building block for recent advances using deep learning for Natural Language Processing. Simplistically, a neural network is a program which is designed to work in a manner similar to that of the human brain. When we employ a single artificial neural network to build a model for the task of end-to-end Machine Translation, the resulting approach is termed as Neural Machine Translation. Figure 2.1 shows a simple block diagram of a Neural MT system.

It can be seen that the model comprises of two main components - the encoder and the decoder. The encoder encodes the source sentence by reading it one symbol at a time and storing the hidden state into a vector representation using a recurrent activation function. These recurrent activation functions could vary depending upon the architecture employed for training the model. Some examples of resulting encoder architectures are Hyperbolic tangent (*tan*) , Convolutional Network (CNN) , Gated Recurrent Unit (GRU) or Long Short Term Memory (LSTM).

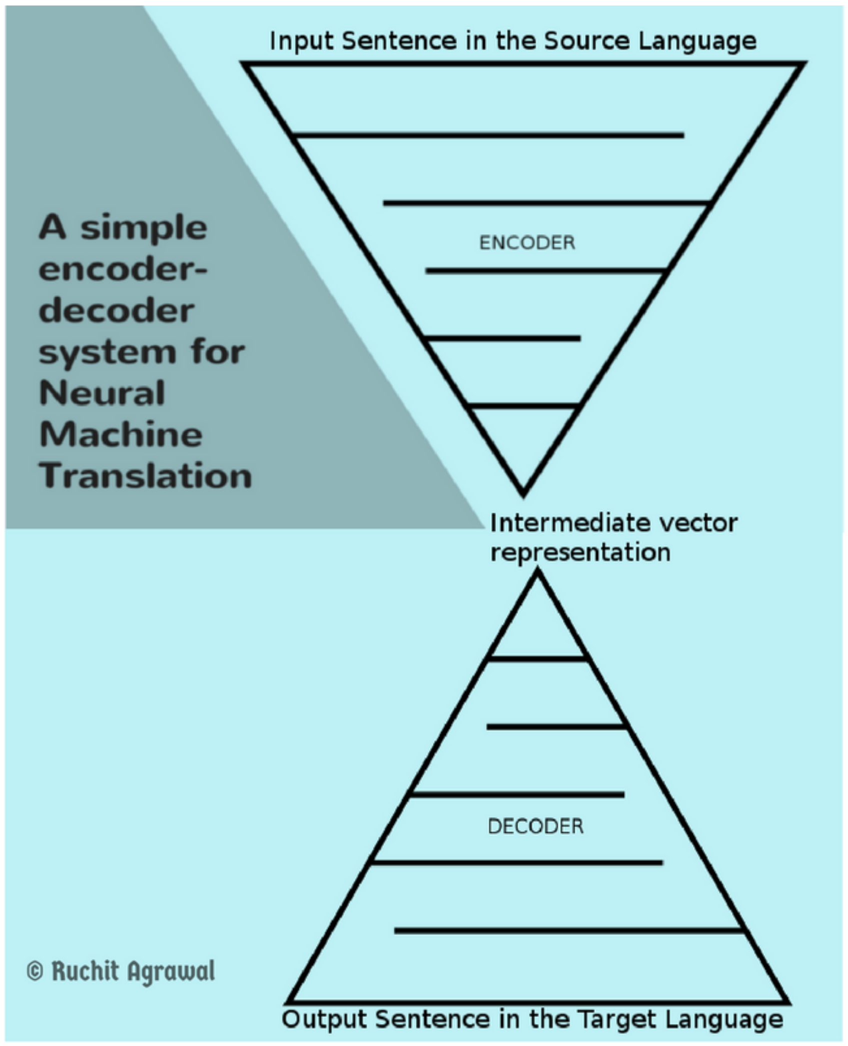

Input Sentence in the Source Language
A simple encoder-decoder system for Neural Machine Translation
ENCODER
Intermediate vector representation
DECODER
© Ruchit Agrawal
Output Sentence in the Target Language

Figure 2.1 *A simple NMT model with an encoder-decoder architecture*

The decoder can be visualized as a conditional recurrent Language Model (LM) where the conditioning is being done based on the source sentence, i.e. one can think of the final hidden state as a representation which summarizes all the information present in the complete source sentence.
Thus the entire process can be understood as passing information in time steps to the encoder which generates a hidden state representation, passes it on to the decoder which then uses this information for prediction of the next correct step in the translation process. The goal of NMT can be described as designing a model which can be completely trained with the components being able to be fully tuned using training corpora to generate end-to-end translations. This implies that NMT does not rely on feature specifications which are pre-designed and fed into the model by the user; rather it learns its own set of features from the training corpus itself. It should be noted that neural MT systems are known to require large amounts of training data to perform well, primarily due to the factors mentioned above.

2.1.2 A Brief History of NMT

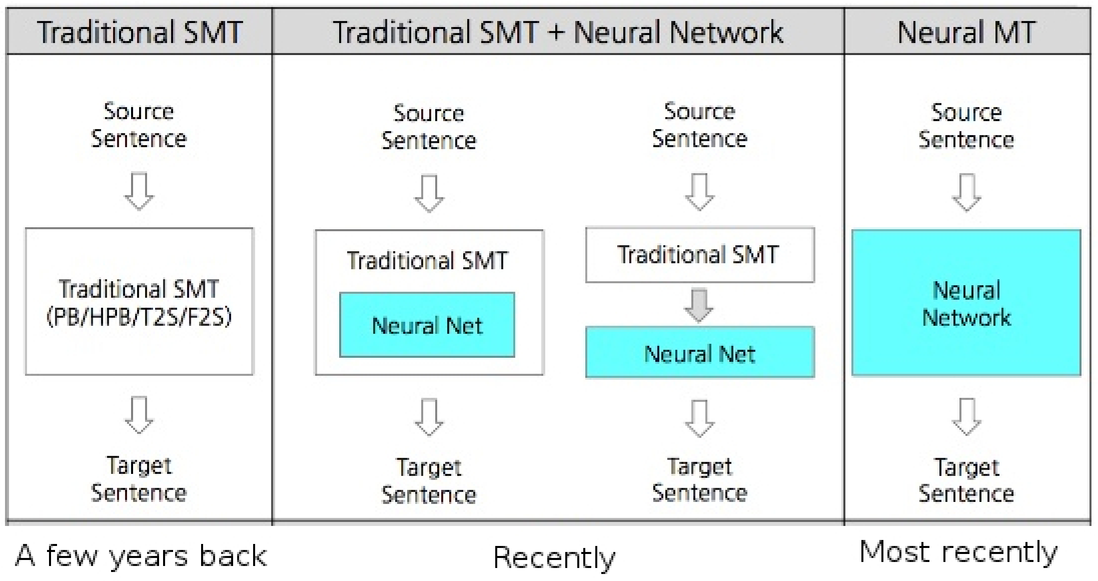


Figure 2.2 *Recent advances in the usage of neural networks in MT.* Image inspired from [38]

The usage of large neural networks for Natural Language Processing (NLP) tasks was initially proposed by [5] in his feed-forward neural language model. The neural LM is very similar to the current existing LMs. The input n-gram is projected into an embedding space for each word and then we have a big output layer. This novel idea was then used by several researchers who tried to integrate it with Machine Translation systems ([3] and [13]).

[17] proposed the usage of a neural network joint model as an additional technique to improve SMT performance. [55] was a breakthrough for MT, introducing the ”seq2seq” (Sequence to sequence) model which was the first model based completely on NN’s and achieving accuracy comparable to the State-of-the-Art SMT systems. They proposed the usage of an RNN (LSTM or GRU) over the source sentence, producing a hidden state and then running another RNN to generate the output one word at a time. The bottleneck to this approach was that the entire translation is a fixed sized vector.

[4] proposed the attention mechanism in NMT, which is employed currently in most State-of-the-Art MT systems. The decoder computes a relevant score for each annotation at each timestep and uses the weighted sum of the annotations as a context. The attention mechanism enables the model to choose which part of the sentence to pay ”attention” to in a differentiable way.

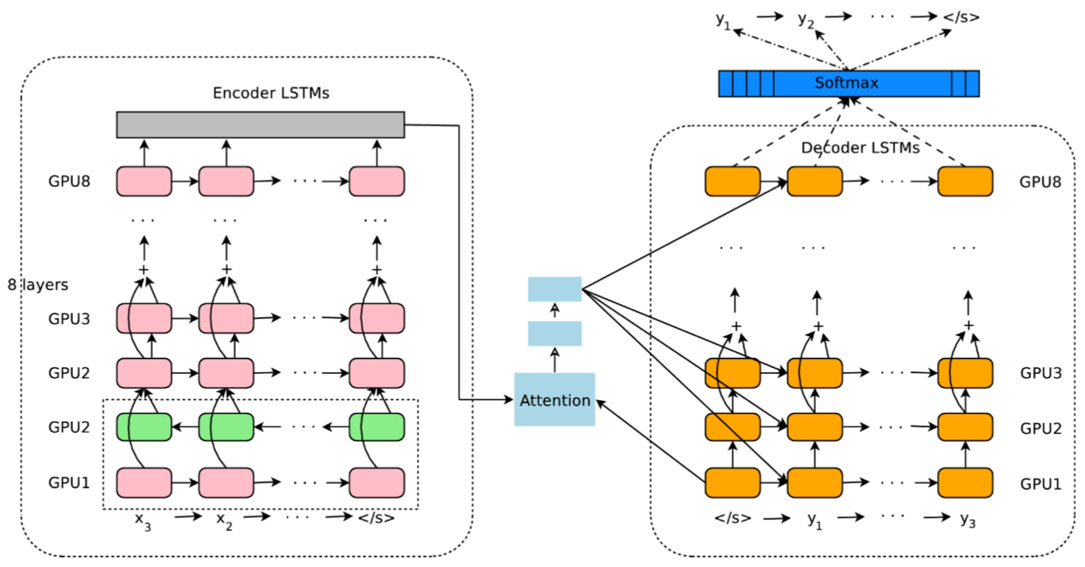


Figure 2.3 *Architecture of GNMT[59] - Google’s NMT model*

2.1.3 Why RNNs?

Neural MT generally uses some form of a Recurrent Neural Network (RNN) for its encoder and decoder components, rather than a normal neural network. The motivation behind this comes from a variety of reasons. Traditional neural networks have a huge RAM requirement and are not quite feasible in their best settings where they achieve their highest accuracies. This is done because RNN’s facilitate the preservation as well as processing of information that has a temporal aspect involved, for eg, a sequence of words has an order, and hence a time element inherent in it. This is not possible if we employ normal neural networks.

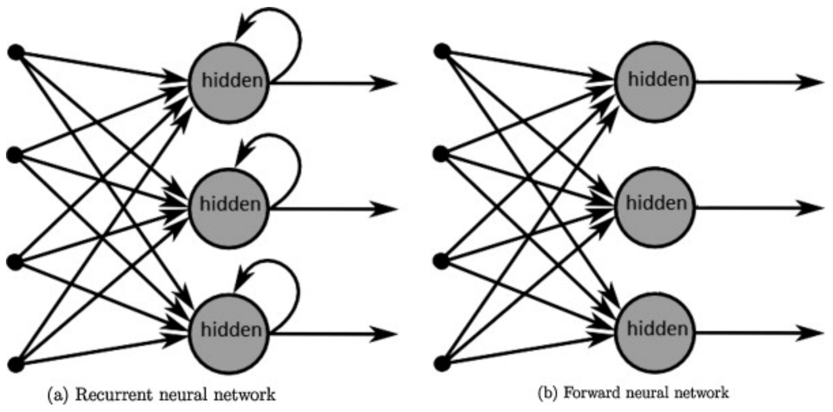


Figure 2.4 *A comparison of feedforward neural networks with Recurrent Neural Networks*

One important property of machine translation, or any task based on natural languages, is that we deal with variable-length input and output. In other words, T and T' are not fixed. On the other hand, one of the major assumptions in feedforward neural networks is the idea of fixed length, i.e. the size of the input layer is fixed to the length of the input sequence. The other major assumption is the idea of independence - that different training examples (like images) are independent of each other. However, we know of sequences such as sentences or speech, where there are short and long temporal dependencies that have to be accounted for. To deal with these types of variable-length input and output, we need to use a recurrent neural network (RNN). Widely used feed-forward neural networks, such as convolutional neural networks, do not maintain internal state other than the network's own parameters. Whenever a single sample is fed into a feed-forward neural network, the network's internal state, or the activations of the hidden units, is computed from scratch and is not influenced by the state computed from the previous sample. On the other hand, an RNN maintains its internal state while reading a sequence of inputs, which in our case will be a sequence of words, thereby being able to process an input of any length.

The RNN's thus help in converting the input sequence to a fixed size feature vector that encodes primarily the information which is crucial for translation from the input sentence, and ignores the irrelevant information.

Long Short Term Memory (LSTM) units are a type of RNN's which are very good at preserving information through time-steps over a period of time. One key advance in LSTMs in recent years has been the concept of bi-directional encoder and decoder framework. When we employ bidirectional LSTMs, we end up with two hidden states - one in the forward direction and one in the backward direction. This allows the network to learn from the text. Often, even more than two layers are used. Thus there will be multiple layers stacked on top of each other - this is generally only in huge training data conditions. Each one of these has a set of weights inside it. Each one learns and affects one above it. The final state represents everything that is in the source words. Bi-directional generally work the best specially when added with the attention mechanism.

After the encoding process, we are left with a Context vector - which is like a snapshot of the entire source sequence and is used further to predict the output. We have a dense layer with softmax similar to a normal NN, but the difference is that it is time distributed i.e. we have one of these for each time step. The top layer thus will have one neuron for every single word in the vocab, and hence the top layer will be huge in size.

2.1.4 A Comparison with Phrase-based Statistical Machine Translation

Human languages are extremely elegant, flexible, efficient and very complex. As an example, one word can have multiple meanings, and one meaning can be expressed by multiple words. Thus, there is a many-to-many mapping between form and function - a scenario not very favorable for machine learning. Meaning depends on context leading to overloading of certain symbols, literal and figurative meanings of an utterance might be different, and so on. All these factors make Machine Translation in general a very challenging task.

When we deal with Indian languages, there are additional complexities which have been discussed in Chapter 1. Rule-based MT is a direct, transfer-based approach to MT, giving reasonable performance. RBMT translates a source sentence using lexical transfer and in some cases a bit of local reordering in the target side. This is done using a large set of rules which employs a bilingual dictionary and some grammatical properties of the source and the target language. The dictionary can be of the order of 10 power 5, which is quite large. Limitations of this approach include the need of highly skilled experts, the time-consuming and expensive nature of building the system, the rule interaction complexity due to the large size of rules, inability to efficiently incorporate local context on the source side, and inability to integrate smoothly into engineering applications.

Machine learning is a robust and scalable approach to this task in the presence of data. Phrase-based Statistical Machine Translation, belonging to the latter category, has been the state-of-the-art paradigm for MT, honed and optimized for a long period of time. In WMT 2016, for the first time in 15 years, Statistical MT was outperformed by Neural MT for three-fourths of the shared tasks. For a comparison, Statistical MT has been around for more than 25 years, whereas Neural MT has been proposed two years back. In such a short span of time, Neural MT has been producing better results than Statistical MT.

Statistical approaches to MT employ predictive algorithms to create models using bilingual parallel corpora. They provide the most probable output on the basis of the examples from the bilingual training corpora. Thus, using an already translated dataset, a statistical model generates prediction of the translated output in the target language. There can be various approaches within Statistical MT : like word-based approach, syntax-based approach, phrase-based approach and hierarchical phrase-based approach. The phrase-based SMT (Also called PBSMT) has been producing the state-of-the-art results before the advent of Neural Machine Translation (NMT).

Although statistical approaches provide the advantage of automation, there are a variety of drawbacks of this approach. These include the following :

- Multitude of local decisions and independence assumptions, leading almost to ”translation in isolation”
- Weak re-ordering and a lot of Out-of-Vocabulary tokens
- Inability to caputre non-local phenomena
- Dependence on heterogenous technologies, i.e. the components are individually estimated rather than joint optimization as done in NMT

  Neural MT is a radically different approach to MT, which uses artificial neural networks to generate end-to-end translation. Recent experiments have demonstrated NMT is able to generate significantly better translations than SMT in terms of accuracy and fluency, especially on lexically rich data and in presence of large training corpora. One big benefit of NMT is the removal of the need for extensive feature engineering, since NMT is capable of learning linguistic rules from the data itself by employing various algorithms. The benefits of NMT include :

  - Small memory footprint as opposed to SMT which has large phrase tables
  - Faster learning and convergence
  - Support for end-to-end training
  - Self-learning capability, no feature design and engineering required
  - Ability to combine supervised as well as unsupervised learning approaches
  - Joint optimization of all components against the same loss function, making it homogeneous
  - Many possible extensions - like sub-word based or character based translation

    Figure 2.5 shows the performance obtained by NMT and SMT on some standard language pairs.

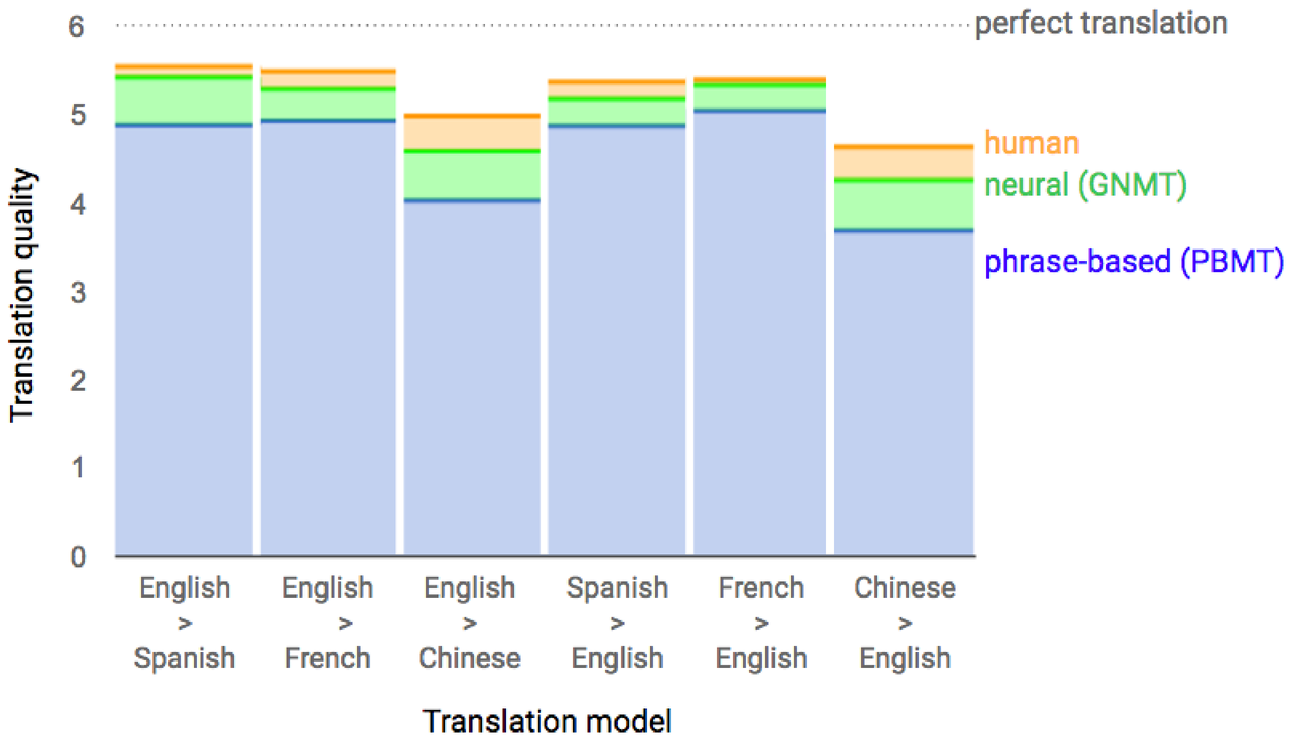


Figure 2.5 *Comparison of performance of NMT and SMT for standard language-pairs by [36]*

2.2 Summary

In his chapter, we discussed the theory and history of NMT briefly and compared it with Phrase Based SMT, the current state-of-the-art for Indian languages. We showed that NMT is a promising direction to pursue for the task at hand and hence we employ the same in this study. We describe our system architecture and experimental setup in the next chapter.

Chapter *3*

System Architecture and Experimental Setup

In this chapter, we desribe the complete experimental setup required for building the NMT systems. We begin by the description of the datasets and resources employed for our experiments, followed by a discussion on the selection of optimal NMT architecture, and conclude the chapter with the details of the selected architecture.

3.1 Datasets and Resources

The normal requirement for efficient NMT performance is a large parallel training corpus. However, for this study, we deal with the less-resourced scenario for majority of the languages. We tweak our system architecture to deliver optimum performance in such scenario, described in detail in Section 3.2. The following subsections describe the datasets and resources employed for our experiments.

3.1.1 Datasets

We employ a small parallel corpus and large monolingual corpora for training. For the former, we use the multilingual Indian Language Corpora Initiative (ILCI) corpus [1], which contains 50,000 sentences from the health and tourism domains aligned across eleven Indian languages. We employed manual preprocessing to eliminate misalignments - the resultant dataset has a size of 47,382 sentences. These are split randomly into training set, validation set and test set containing 44,000, 1382 and 2000 sentences respectively.

The statistics for the ILCI corpus are given in Table 3.1. We use the EMILLE monolingual corpora [43] for five languages [2] and the UrMonoCorp [26] for Coarse

[1]This corpus is available on request from TDIL : https://goo.gl/VHYST

[2]The corpora for the remaining languages are insufficient in size.

Table 3.1 Corpus statistics - ILCI

	Tokens	Vocabulary
hin	850968	39170
pan	849679	849679
guj	759380	62780
tam	849679	86462
ben	715886	50553
urd	832776	36738
tel	632995	86997
kon	643605	70030
eng	808370	35134
mar	663597	77057
mal	599422	101869

Learning detailed in Chapter 4.[3]. These statistics are given in Table 3.2. In addition to these, we extract samples from the EMILLE [43] parallel corpus for the Housing and Legal domains. These datasets are used as test sets to show coverage of our NMT model. Details are given in Table 3.3.

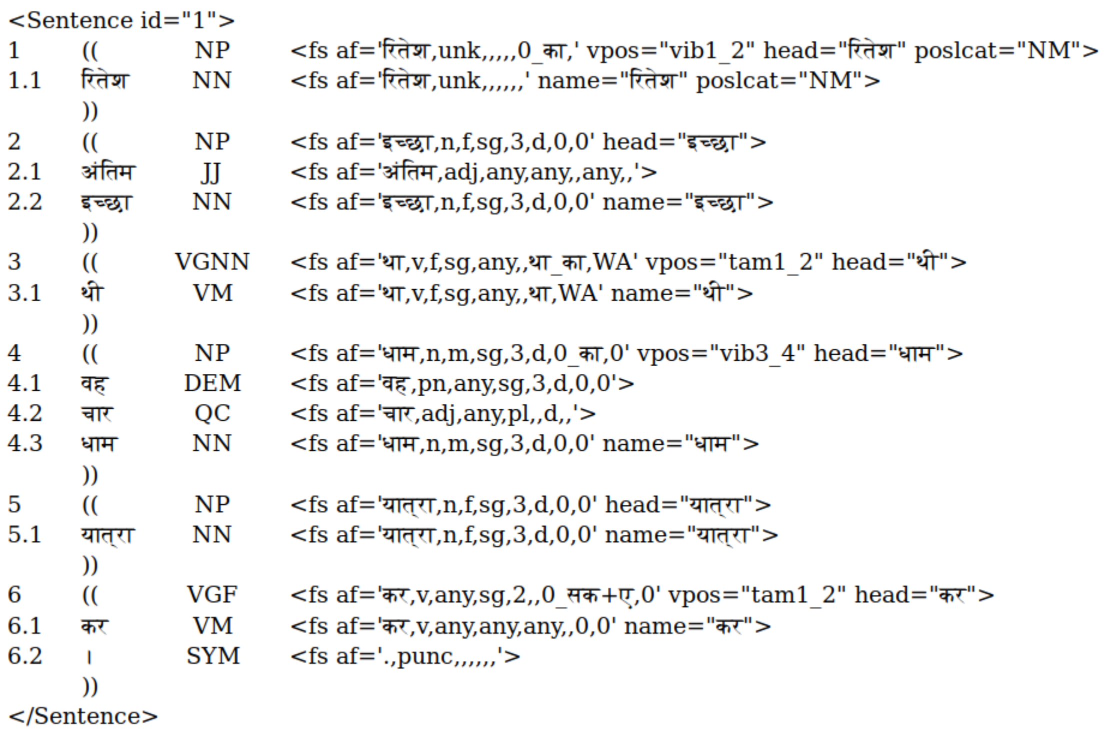
Table 3.2 Monolingual Corpora statistics - EMILLE and *UrMonoCorp

	Sentences	Tokens	Vocabulary
hin	612705	11986152	321356
pan	488985	14285063	272771
tam	827439	17170697	1285031
guj	272526	12766111	660465
ben	259145	2671369	243531
urd*	500000	8744825	157133

3.1.2 Resources

For our experiments, we use synthetic data in addition to the gold data (described in detail in Chapter 4) to compensate for the relatively lower size of our gold corpus. The generation of synthetic data from the monolingual corpora is done using the *Sampark* [2] systems, which are available for 9 Indian language pairs[4]. Sampark is a multipart machine translation system developed under the Indian Language Machine Translation project. It uses a transfer-based engine and has a huge repository of rules for dealing with Indian language specific constructs. The motivation behind this

[3]We extract a sample containing 500,000 sentences from UrMonoCorp
[4]https://goo.gl/yu7KUT



```
<Sentence id="1">
1       ((      NP      <fs af='रितेश,unk,,,,,0_का,' vpos="vib1_2" head="रितेश" poslcat="NM">
1.1     रितेश    NN      <fs af='रितेश,unk,,,,,,' name="रितेश" poslcat="NM">
        ))
2       ((      NP      <fs af='इच्छा,n,f,sg,3,d,0,0' head="इच्छा">
2.1     अंतिम    JJ      <fs af='अंतिम,adj,any,any,,any,,'>
2.2     इच्छा    NN      <fs af='इच्छा,n,f,sg,3,d,0,0' name="इच्छा">
        ))
3       ((      VGNN    <fs af='था,v,f,sg,any,,था_का,WA' vpos="tam1_2" head="थी">
3.1     थी      VM      <fs af='था,v,f,sg,any,,था,WA' name="थी">
        ))
4       ((      NP      <fs af='धाम,n,m,sg,3,d,0_का,0' vpos="vib3_4" head="धाम">
4.1     वह      DEM     <fs af='वह,pn,any,sg,3,d,0,0'>
4.2     चार      QC      <fs af='चार,adj,any,pl,,d,,'>
4.3     धाम      NN      <fs af='धाम,n,m,sg,3,d,0,0' name="धाम">
        ))
5       ((      NP      <fs af='यात्रा,n,f,sg,3,d,0,0' head="यात्रा">
5.1     यात्रा    NN      <fs af='यात्रा,n,f,sg,3,d,0,0' name="यात्रा">
        ))
6       ((      VGF     <fs af='कर,v,any,sg,2,,0_सक+ए,0' vpos="tam1_2" head="कर">
6.1     कर      VM      <fs af='कर,v,any,any,any,,0,0' name="कर">
6.2     ।       SYM     <fs af='.,punc,,,,,,'>
        ))
</Sentence>
```

Figure 3.1 The output obtained after shallow parsing a Hindi sentence

choice for synthetic data generation stems from the quality of performance obtained using Sampark for Coarse Learning. This is described in detail in Chapter 5.

The features used in Section 4.1 are generated from the intermediate outputs of *Sampark* [2] and the shallow parsers. The shallow parsers are available for nine Indian languages[5]. Figure 3.1 shows the shallow parsed output for the Hindi sentence : "रितेश की अंतिम इच्छा थी की वह चार धाम की यात्रा कर सके |" .
Detailed information on feature extraction and addition is provided in Chapter 4.

3.2 System Architecture

3.2.1 Fundamental Skeleton of the Network

The main component of our NMT model is a single neural network trained jointly to provide end-to-end translation [28, 55, 13, 4]. Our main architecture consists of an encoder-decoder framework, comprising of bidirectional Long Short Term Memory (LSTM) units, a type of bidirectional Recurrent Neural Network (RNN) [47] as shown in Figure 3.2.

[5]https://goo.gl/Dt3zHi

Table 3.3 Parallel Corpus Statistics - EMILLE. H: Housing, L: Legal

		Sentences	Tokens	Vocabulary
hin	H	1183	23178	3131
	L	1321	27700	3880
ben	H	1109	17815	3310
	L	1288	21690	4567
guj	H	1113	17537	4405
	L	1382	21377	5689
pan	H	1308	20729	3771
	L	1368	24971	3763
urd	H	1327	22691	2871
	L	1386	27207	3945

The encoder encodes the source sentence into a vector from which the decoder extracts the target translation sentences. This facilitates learning of long-distance dependencies, thereby enabling the system to learn an end-to-end model.
Specifically, we model the conditional probability $p(y|x)$ of translating a source sentence x = $x_1, x_2...x_u$ to a target sentence y = $y_1, y_2, ...y_v$. Let 's' be the representation of the source sentence as computed by the encoder. Based on the source representation, the decoder produces a translation, one target word at a time and decomposes the conditional probability as :

$$\log p(y|x) = \sum_{j=1}^{v} \log p(y_j|y_{1:j\text{-}1}, x, s) \tag{3.1}$$

The entire model is jointly trained to maximize the (conditional) log-likelihood of the parallel training corpus (via a softmax layer) with back-propagation through time [57].

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y^{(n)}|x^{(n)}) \tag{3.2}$$

where $(y^{(n)}, x^{(n)})$ represents the n^{th} sentence in parallel corpus of size N and θ denotes the set of all tunable parameters.
We also use an attention mechanism [4] that allows the target decoder to look back at the source encoder.

This is the fundamental skeleton of our architecture. We select the individual components after experimentation with different possible architectures. We describe the experiments and the detailed formulation specifying the RNN type, number of layers, attention mechanism etc. selected after the experimentation in the following subsections.

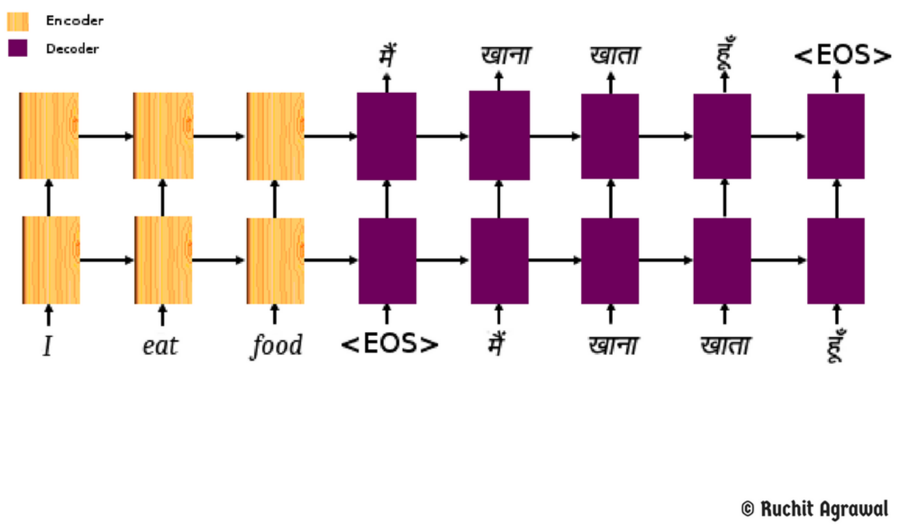


Figure 3.2 *A two-layered encoder-decoder based NMT architecture as proposed by [55]. 'EOS' denotes the end of the sentence.*

3.2.2 Selection of optimal NMT architecture

In order to train the RNNs (encoders and decoders) mentioned in Section 3.2.1, we take the cost function and obtain its derivative with respect to the weight in question. We then move this derivative through the nested layer of computations using the chain rule.
In other words, the output of the previous layer is multipled by the weight matrix and added to a bias and then passed on to an activation function.

$$y_k = g(W\mathbf{y}_{k-1} + \mathbf{b}) \tag{3.3}$$

We use a recurrent connection to convert the linear unit of feed-forward neural network to a recurrent unit so that now the activity of the unit h_t not only depends on x_t (the input) multiplied by the weight matrix, but also on its activity at the previous timestep. The following equation shows this phenomenon :

$$h^{(t)} = g_h(W_1 x^{(t)} + W_R h^{(t-1)} + b_h) \tag{3.4}$$

The second term $W_R h^{(t-1)}$ depends on the activity at the previous timestep multiplied by a recurrent weight matrix. We also want to be able to retrieve an output from this unit and this is done by adding a linear operation as described in the following equation :

$$y^{(t)} = g_y(W_y h^{(t)} + b_y) \tag{3.5}$$

Here, $y^{(t)}$ is a function of $h^{(t)}$ multiplied by weight matrix w and passed through a non-linear activation function. This is the basic element of the recurrent neuron which we employ in the RNN architectures described further. The encoding process can be visualized as the input sequence being compressed by the RNN into an intermediate representation in the form of a fixed dimensional vector. So, if the vector h_{t-1} describes the history of the sequence at timestep t, the new internal state (the updated vector) h_t will be computed by the network, effectively compressing the preceding symbols $(x_1, x_2, \ldots, x_{t-1})$ as well as the new symbol x_t. The following equation shows this :

$$h_t = \phi_\theta(x_t, h_{t-1}) \tag{3.6}$$

Here, ϕ_θ is a function which takes the new information unit x_t and the hidden state h_{t-1} as input. (h_0 can be assumed to be a vector containing zeroes).
We employ an affine transfer passed through a nonlinear function for the implementation of the recurrent activation function ϕ. This is shown by the following equation:

$$h_t = \tanh(W x_t + U h_{t-1} + b) \tag{3.7}$$

Here, W is the input weight matrix, U is the recurrent weight matrix and b denotes the bias vector.
Now, we let our basic RNN model p($x_t \mid x_{<t}$) at each time t to be described by

$$p(x_t | x_{<t}) = g_\theta(h_{t-1}) \tag{3.8}$$

$$h_{t-1} = \phi_\theta(x_{t-1}, h_{t-2}) \tag{3.9}$$

g_θ outputs a probability distribution which is conditioned on the entire history up to the (t-1)-th token via h_{t-1}. Thus, the RNN tries to predict the next token at each time step given the history of the input tokens.

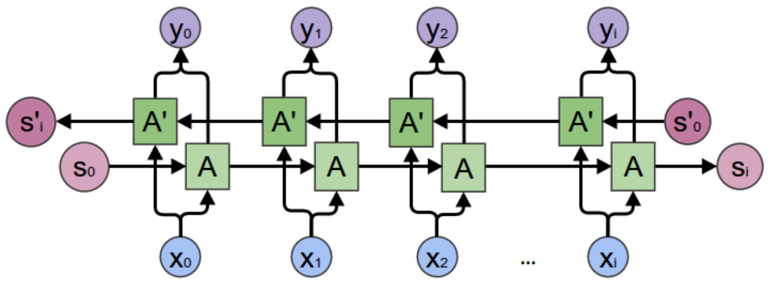


Figure 3.3 *Structure of a bi-directional Recurrent Neural Network. Image Source : https://goo.gl/spMC3J*

3.2.2.1 Bidirectional RNN and Stacking

The Bidirectional recurrent neural network has two recurrent cells which scan the information in different ways, i.e. one in forward direction and one in backward direction. The output is obtained by addition / concatenation depending upon the application. Figure 3.3 shows the structure of a bidirectional recurrent neural network.

The RNN cells can also be stacked on top of each other as shown in Figure 3.5

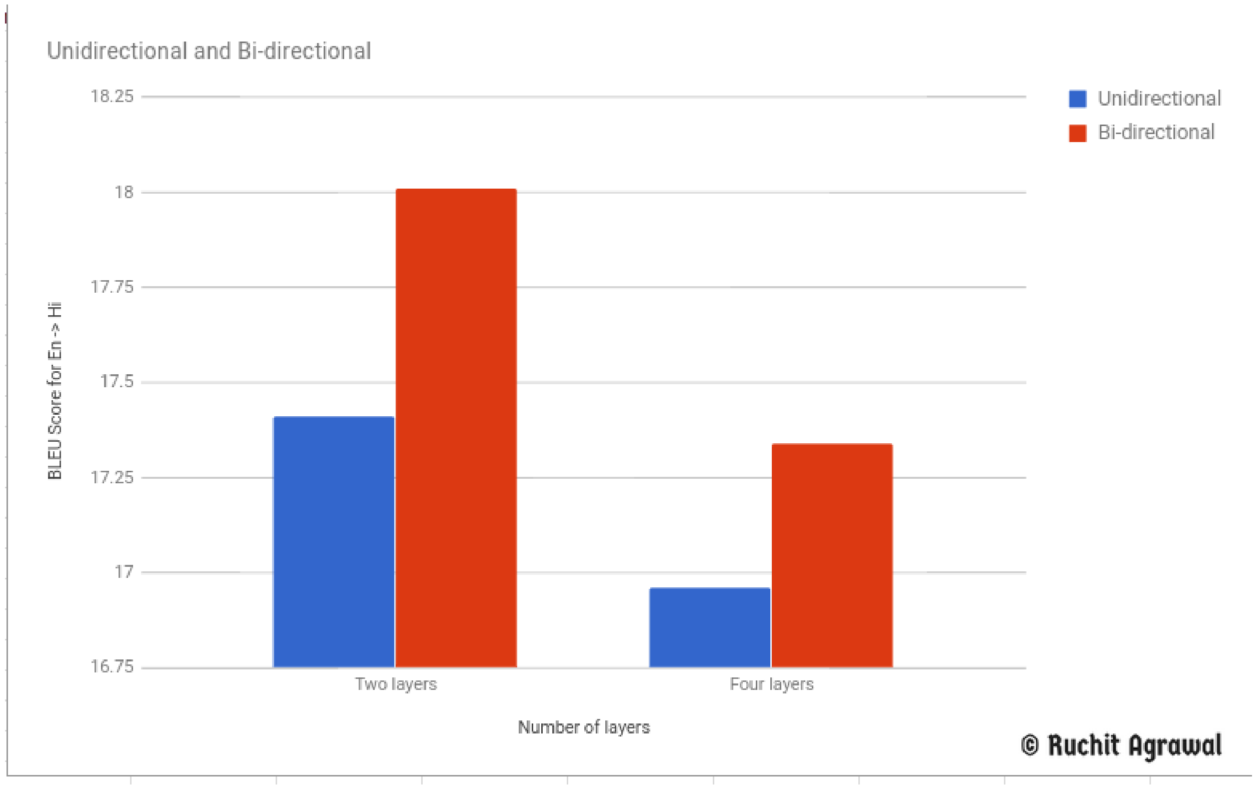


Figure 3.4 *Performance comparison on different layered architectures*

generating a deep neural network. We show the results obtained using unidirectional and bidirectional RNNs obtained on two and four layers of encoder and decoder in Figure 3.4. Since the two-layered architecture shows the best performance, we employ it for our experiments described in Chapter 4.

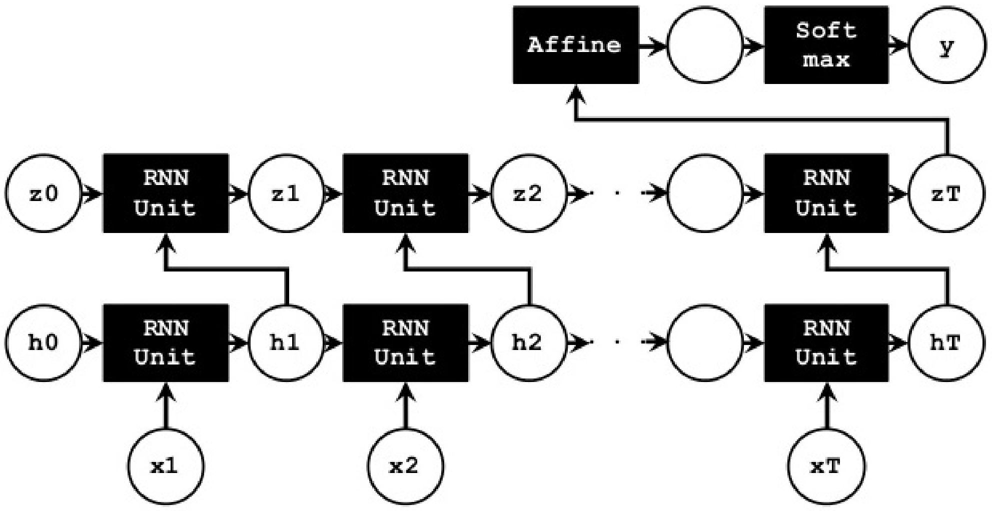


Figure 3.5 *A neural network model with stacked Recurrent Neural Networks. Image Source : https://goo.gl/U1fdqA*

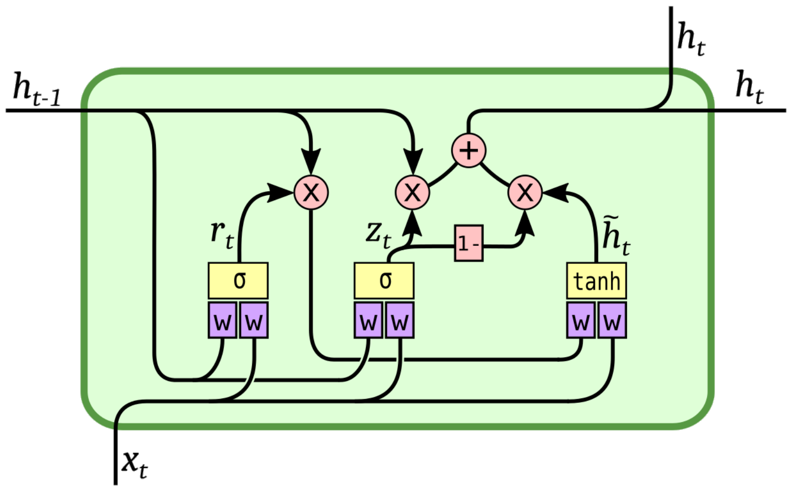


Figure 3.6 *Structure of a Gated Recurrent Unit. Image Source : goo.gl/D848d8*

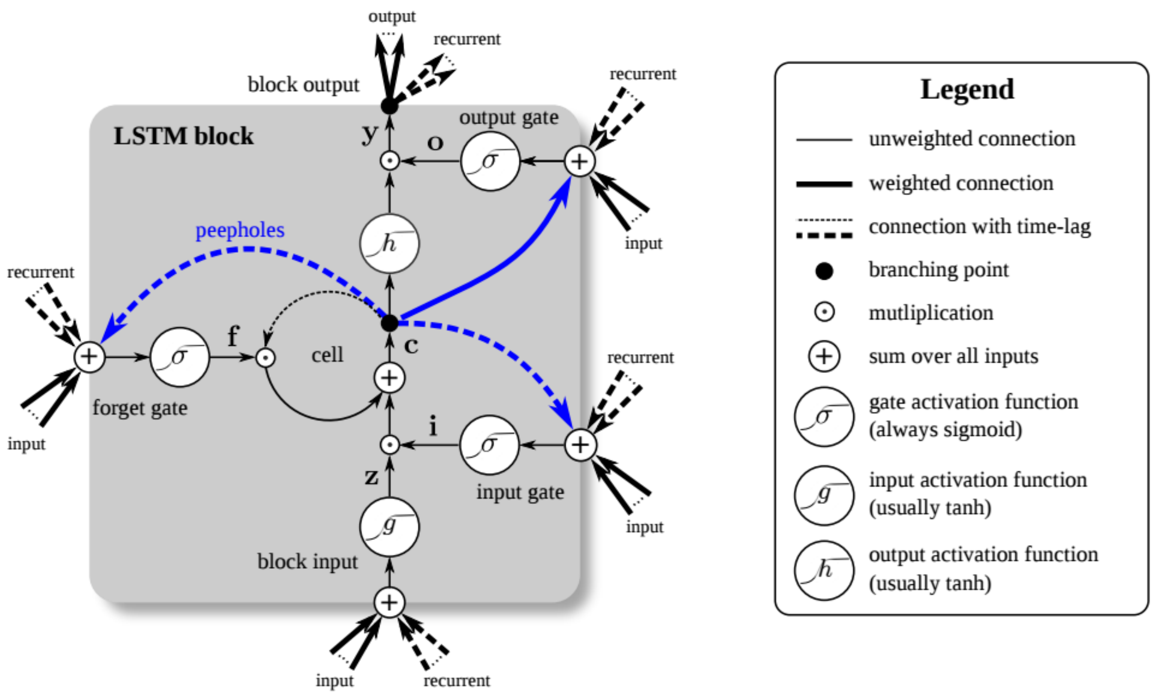


Figure 3.7 *Structure of an LSTM unit. Image Source : https://goo.gl/WpDh9Y*

3.2.2.2 LSTM and GRU cells

The LSTM cells are an improvement over vanilla RNN units. The structure of an LSTM cell is shown in Figure 3.7. In addition to introduction of hidden layers, there is a gating mechanism consisting of an input gate, forget gate and an output gate. We have an inner memory, which is linear to time backwards. The forget and the input gate control what we want to keep from the past as an information and what we want to accept new from the input. [4] use a Gated Recurrent Unit, whose structure is shown in Figure 3.6.

The equations for the formulation of LSTM as well as GRU are described in Figure 3.8. Here, W, W_r and W_u are the input weight matrices; U, U_r and U_u are the recurrent weight matrices and b, b_r and b_u are the bias vectors. The main difference between a GRU and LSTM is how the u_t works. So rather than having a forget gate and an input gate that are both different to get the information inside the inner state, we have the u_t which takes the past hidden value and we have (1 - u_t) which lets the entering of new information. Since u_t is vectorial, it is a combination of many neurons, and the computation is neuron wise, unlike a typical dot product.

LSTM

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)\\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)\\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)\\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)\\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

GRU

$$
\begin{aligned}
z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)\\
r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)\\
\tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)\\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

Figure 3.8 *Mathematical formulation of LSTM and GRU architectures*

3.2.2.3 Encoders and Decoders

We have many time-steps for an encoder. The input vector x is encoded into a state ”thought vector” S_i as shown in the Figure 3.2. Decoders, in turn, decode the state, ”thought vector” through time, thereby generating a sequence-to-sequence architecture when pipelined with the encoder. The thought vector which we use in our experiments is of the of the dimension of 1000. One bottleneck using this approach is the handling of long sequences. The backpropagation offset through time with the backward link makes the neural network more resilient to learning, correcting its errors as it goes forward.
One common technique to improve accuracy is the reverse the input sequence and feed the reversed sequence to the encoder. This makes the related words closer to each other in the encoder and the decoder.

3.2.3 Add ons

3.2.3.1 Lookup Tables and Embeddings

We pass on the numbers as input to the neural network rather than plain words. This also helps the model to learn related concepts easily. Lookup tables are used for this. Words that occur with very low frequency are discarded and replace with an <UNK> (UNK stands for Unknown) token. This is done to reduce the size of the vocabulary, since the computational power and complexity increases linearly with increase in the vocabulary size. For example, if there we have a vocabulary size of 200000, there has to be a dense layer of 200000 neurons at the top, which goes into

a softmax to predict the word output. To avoid this, we use lookup tables and word embeddings.
Word embeddings allow us to extract more semantic information from the words. Often, pre-trained embeddings like word2vec or GlOVe are used. Since the embeddings are generally trained across billions of words, they are able to spot relationships and leverage semantic information in the neural network.

3.2.3.2 Padding

A sequence to sequence model generally has a fixed length for the sequence, for example 30 time steps. To achieve a common length across input sequences, we use padding. We pad the sequence with special tokens to achieve the desirable length. All the input sequences must have a common length and all output sequences must have a common length, however the input sequence length need not be the same as the output sequence length.
The padding tokens employed by us belong to the following categories (make a table): <PAD> denotes a padded zero input , <EOS> denotes end of sentence, <GO> tells the decoder to start functioning, <OOV> represents an out-of-vocabulary token, <UNK> represents an unknown token and <ES2> specifies the target language. These tokens thus can be thought of as giving conditional information to the network as to what it should be doing.

3.2.4 Challenges in "pure" NMT

A major limitation in NMT is that it is not able to incorporable larger contextual information efficiently. Vanilla sequence to sequence models work well on short sentences but not on long ones. LSTMs can remember up to about 30 time steps, dropping off quite quickly after 30. So it cannot go back in time across some paragraphs and use that contextual information to produce output. One work-aoround to this is that we can flip the input sentence and feed it to the seq2seq model thus training it backwards. ie. backwards going in forwards coming out, thereby making the gap between the end words significantly lesser. Some other approaches to use this are attention mechanism, peeking, teacher forking, and beam search. Attention mechanism looks at the whole thing and works out "Which word is most important for this word" i.e. it computes a score for every word in the sentence and with that it is able to get a sense that there are certain words which rely much more on other words than other words. With attention mechanism the performance for short sentences is improved as well. Without the attention mechanism, NMT often ends up producing sentences with correct grammatical structures but sometimes with many

repetitions (for example : Ram bought five five five kilograms of mangoes).
Attention can be thought of as a memory module which sits above the network, looks at the words and finds out which are the most important. Another method is teacher forcing - When training the network; instead of letting the decoder pass its predictions to the next layer, we pass the correct word/state. So when it makes the prediction, we check whether it is right or wrong and use it when we are back-propoagating the network, but we dont feed it to the next timestep, rather we feed the correct answer to the next timestep. Thus we are forcing the decoder to not just use the output (last hidden state) but we are forcing it to use the correct answers. Obviously this cannot be used when dealing with real predictions. Often we can train with teacher forcing, and save the weights of the model and load them in the model and train without teacher forcing and then use it for normal predictions. Teacher forcing can enable faster and more accurate learning.
Next technique is peeking - Normally we feed the hidden state of our context vector straight through every step of the next RNN or LSTM. Generally, each of these steps the context vector gets changed. For peeking we also give the version just outputted by the encoder along with the current vector so that checking and correction can be done.
Out of these methods, we employ attention mechanism in combination with input feeding, briefly described in the following subsections.

3.2.5 Attention mechanism

The attention mechanism was proposed by [4]. It can be applied to a wide variety of deep learning applications like control problems, image captioning, speech recognition and machine translation. Their intuition was "why encode a single thought vector between the encoder and the decoder when we can have everything?" Attention mechanism is depicted in Figure 3.9. In normal RNN architectures, the entire input sequence is used for prediction of the output. However, we know that some words are more important and some words are not so important when dealing with the task of Machine Translation. Attention mechanism is based on this idea - it gives "attention", i.e. weightage to specific parts of the input which are more significant for the task at hand, thereby helping to bring down the computational complexity of the model. This is especially helpful when dealing with long sentences.

The traditional encoder-decoder framework at first encodes the source sentence into a single vector representation which is used by the decoder to predict every single word in the output. In other words, each input word is used equally used for translating the sentence, and all words are used by the decoder at each timestep.

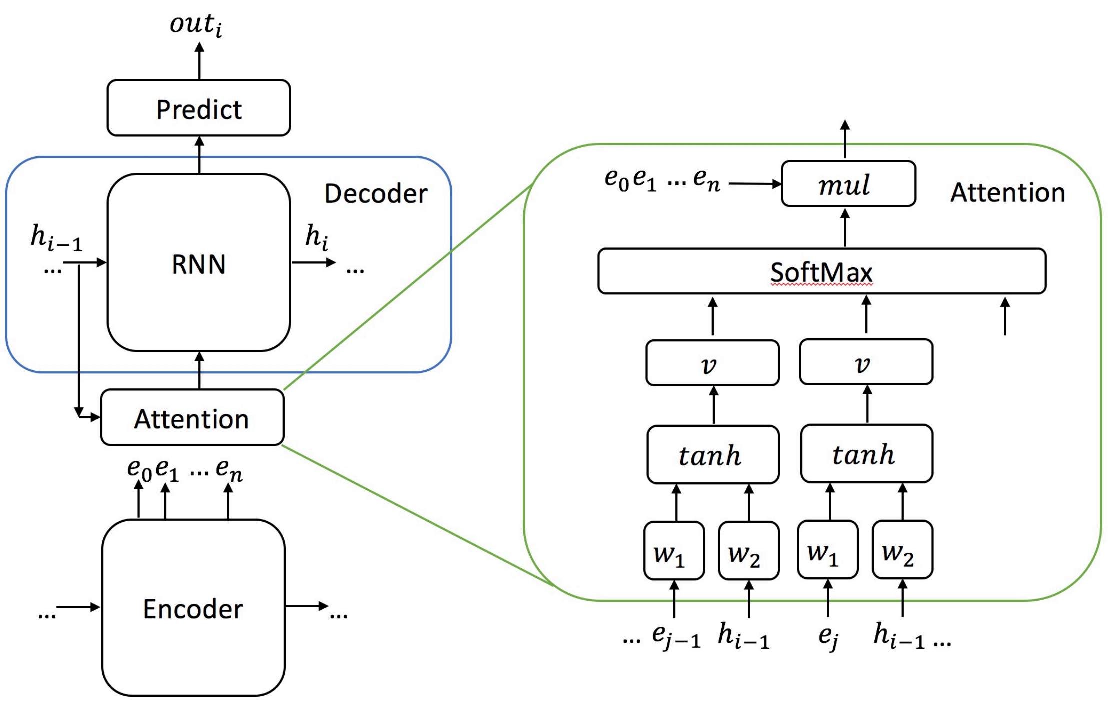
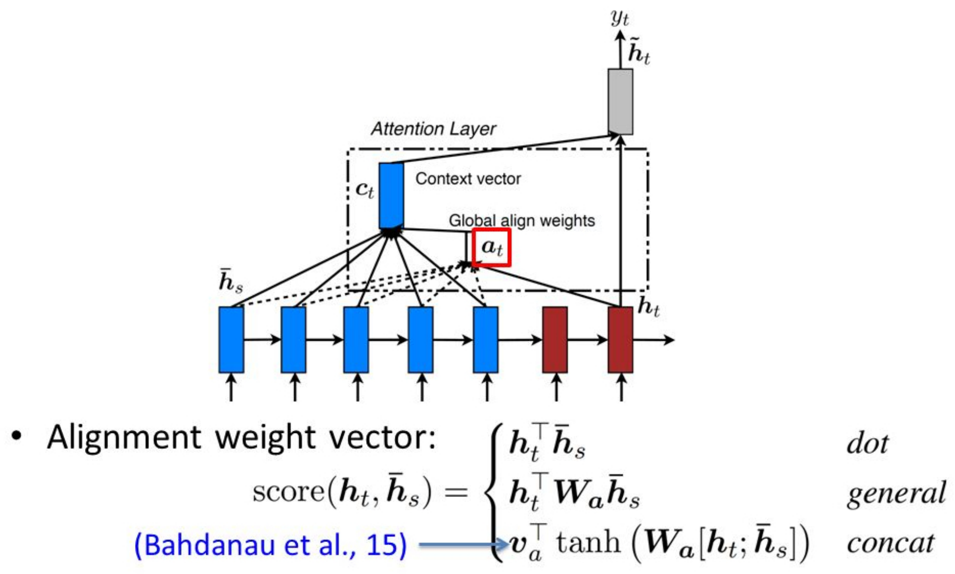


Figure 3.9 *Attention mechanism in RNNs.*

With the attention mechanism, the decoder computes a set of attention weights which is applied at the input sequence at each timestep. The set of attention weights changes over time. A weighted sum using the attention weights of the input generates each word. Since the attention weights change with time, the model can "focus" in different places as the translation process moves forward.

The attention model keeps track of the source hidden states as a memory pool. The reference is then done to the words according to the weights assigned to them. Various extensions have been proposed to the attention mechanism. [40] proposed local attention to have focused attention. Instead of looking at the entire source hidden states (also called Global Attention or soft attention), they look at a subset of source states at each timestep. Global attention is shown in Figure 3.10. In order to compute the alignment weight vector a_t, a scoring mechanism is needed to compare the target hidden state h_t with the source hidden state h_s. Different scoring functions can be used for this. The last one is the one proposed by [4]. Once the alignment vector is obtained, the context vector can be obtained by performing a weighted average of source hidden states. Given the context vector and the target hidden state history, the attentional vector ($\tilde{h}_t$) can be computed by a small feed-forward neural network.



- Alignment weight vector:

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

(Bahdanau et al., 15)

Figure 3.10 *Global Attention Mechanism*

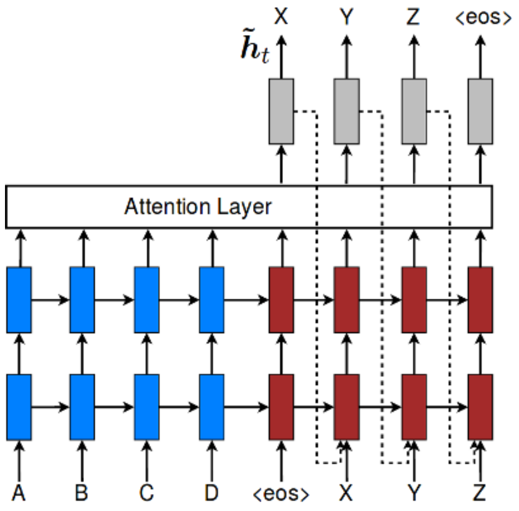


Figure 3.11 *Input-feeding in Attention Mechanism*

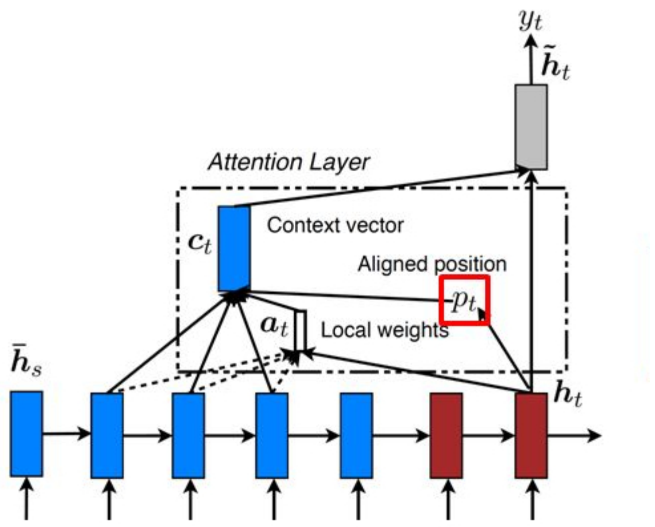


Figure 3.12 *Local Attention Mechanism*

An example for this is :

$$\tilde{h}_t = \tanh(W_c[c_t; h_t]) \tag{3.10}$$

Given the attentional vector, we can give the predictions to generate the next output word y_t.

For local attention, the structure is similar, but now we try to predict the aligned position p_t, which defines a focused attention. This implies that we look only at a context window of (p_t - D, p_t + D) rather than looking at entire source hidden state. [60] proposed a combination of soft as well as hard attention - the advantage of this method is that it is differentiable, unlike hard attention where we need to employ reinforcement learning or ensembling based techniques.

We also employ Input-feeding, an effective extension to attention mechanism where the attentional vectors are fed to the next timesteps as shown in Figure 3.11.

We perform preliminary experiments with different RNN architectures and the results are shown in Figure 3.13. We select the optimal architecture after this experimentation. The final resultant architecture of our model is shown in Figure 3.14 and Figure 3.15.

3.3 Summary

In this chapter, we described the dataset and resources employed in this study along with the detailed formulation of the system architecture employed. The architecture is finalized after preliminary experimentation on different RNN architectures and the

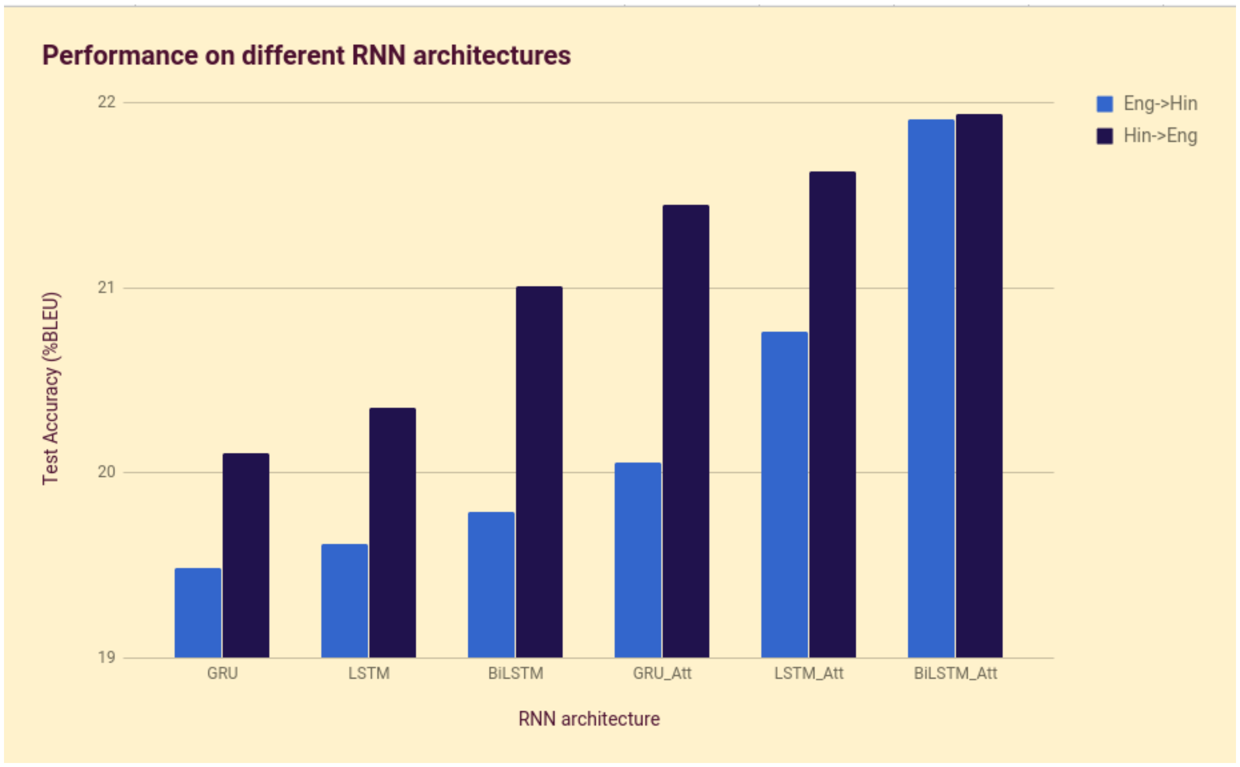


Figure 3.13 *Performance of different RNN architectures for Eng $\leftrightarrow$ Hin*

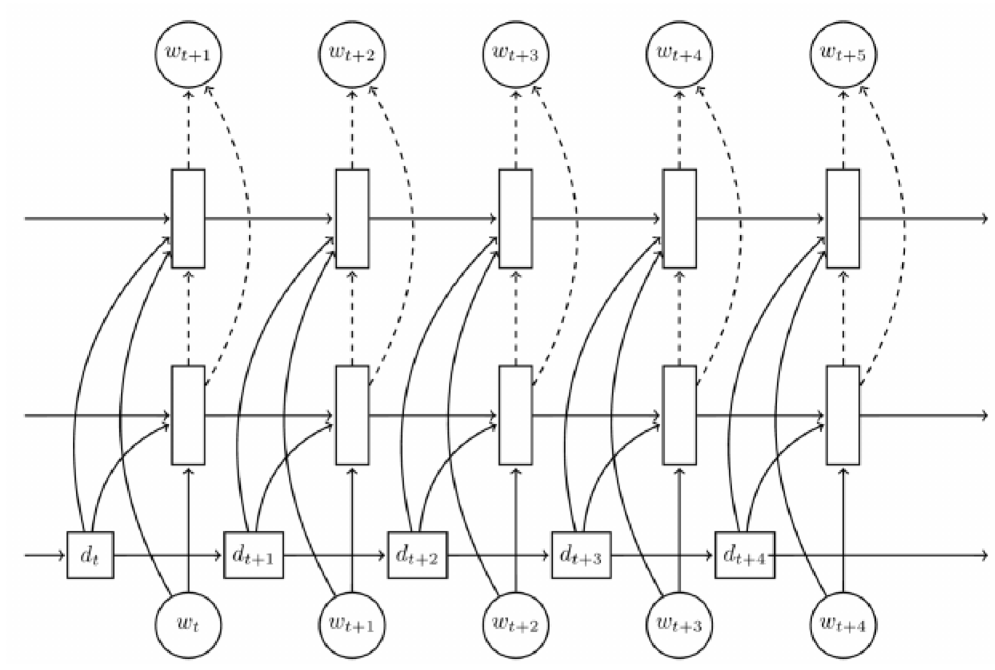


Figure 3.14 *Our detailed architecture (a)*

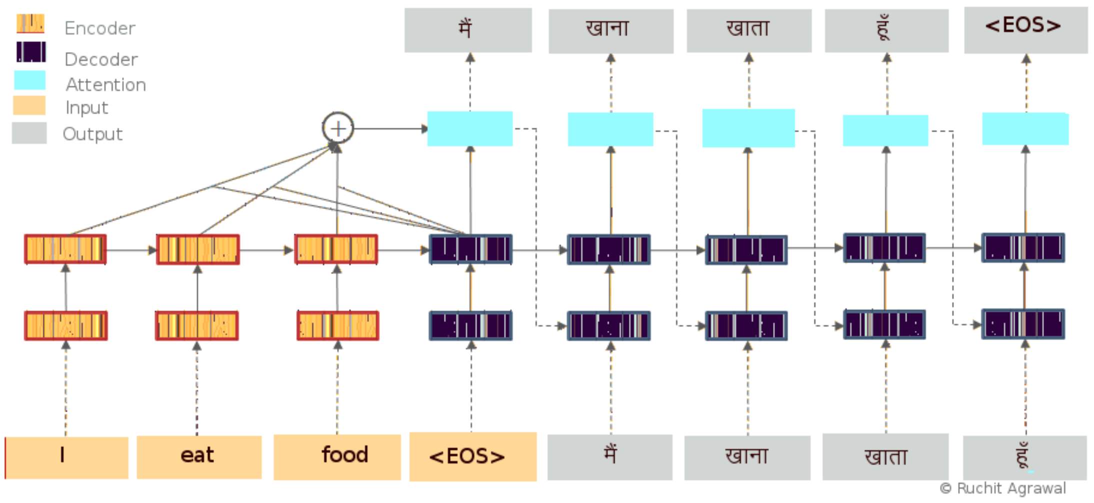


Figure 3.15 *Our detailed architecture (b)*

one producing the best results is chosen for the main experiments described in the next chapter.

Chapter 4

Approaches and Experiments

In this chapter, we describe our two proposed methods to generate efficient translations and deal with the challenges mentioned in Chaper 1. Both of the methods use certain additional information to aid a baseline NMT system trained over the ILCI corpus. The first method employs linguistic features specific to Indian languages in addition to some generic features to improve translation quality. The second method comprises of three-phase training, leading to better accuracy as well as domain coverage. It employs a large monolingual corpus of the source language in addition to an existing MT system to generate a robust NMT model. The second method can be pipelined with the first method or employed independently in case of absence of quality feature generators needed for the first method. We show results using the pipelined approach. We use the NMT architecture as described in Section 1 of Chapter 3. The datasets and resources used for training are detailed in Section 2 of Chapter 3. We employ the tool OpenNMT [30] for the implementation of the architecutre. The parameters are tuned using grid search. These are discussed in detail in Section 4.3. The two methodologies along with the experiments are described in the following subsections. We report the results obtained on each method in Chapter 5 using BLEU score [44] as the evaluation metric.

4.1 Exploiting Linguistic Information to aid NMT

Large parallel corpora for Indian languages are not easily available. In order to compensate for the lack of large parallel corpora needed for NMT training, we propose the usage of specific linguistic features as additional information to improve translation. This helps in leveraging characteristics of Indian languages - like a relatively free word order paradigm (with the Subject-Object-Verb (SOV) structure commonly used) where constituents of a sentence can occur in any order without affecting the overall meaning; high degree of inflection like syncretism, agglutination and allomorphism; usage of converbs; reduplication; relative participial forms and correlative clause constructions.

We extract the following features specific to Indian languages using the tools described in Section 3.1.2. We provide examples for each feature in Hindi and Tamil, languages belonging to two different language families (Indo-Aryan and Dravidian respectively). This helps in understanding the encoding of linguistic knowledge which these features provide.

1. **Part-of-Speech tags** : Indian languages have several unique POS tags; like the following:

   * Quotative
     · 'माने' ('maane', meaning 'means')
     · 'என்று' ('en̲r̲u', A quotative particle)
   * Demonstrative
     · 'वहाँ' ('wahaan', meaning 'there')
     · 'அந்த' ('anta', meaning 'that')
   * Noun denoting spatial and temporal expressions (NST)
     · 'आगे' ('aage', meaning 'ahead' or 'further')
     · 'பின்பு' ('pin̲pu', meaning 'after/behind/later')
   * Reduplication (RDP)
     · 'छोटे–छोटे' ('chhote (JJ) chhote (RDP)', where 'chhote' means 'small' and reduplication implies the meaning 'all of the small ones')
     · 'धीरे धीरे' ('dheere (RB) dheere (RDP)', where 'dheere' means 'slow' and the reduplication implies the meaning 'slowly, slowly')
     · 'பார்த்து பார்த்து' ('pār-tt-u (VB) pār-tt-u (RDP)', where 'pār-tt-u' means 'see' (PST-CONJ) and reduplication adds emphasis).
2. **Vibhakti** - 'Vibhakti' is a Sanskrit term for inflecting nouns and verbs, more generally used for case markers for nouns. Indian languages depict different behavior in how nouns mark their cases. Some languages have suffixes (surface case endings); some, such as Hindi, use post positions and some use a combition of the two. For example:

   Vibhakti in Hindi

   (a) मनीष ने दीपा को किताब दी।
   Manish ne Deepa ko kitaab di.
   manish -ne deepa -ko book give.

(Manish gave the book to Deepa.)
.

(b) दीपा को मनीष ने किताब दी।
Deepa ko Manish ne kitaab di.
Deepa -ko Mansh -ne book give.
(Manish gave the book to Deepa.)
.

Vibhakti in Tamil

(a) மனீஷ் தீபாவுக்கு புத்தகத்தைக் கொடுத்தான்.
Manish deepaavukku puttakattai koduttaan.
Manish deepaavu-kku puttakatt-ai koduttaan
(Manish gave the book to Deepa.)

(b) தீபாவுக்கு புத்தகத்தைக் மனீஷ் கொடுத்தான்.
deepaavukku puttakattai maneesh koduttaan.
deepaavu-kku puttakatt-ai maneesh koduttaan
(Manish gave the book to Deepa.)

In the Hindi sentences, 'ne' and 'ko' act as the vibhakti markers. In the Tamil sentences, 'kku' and 'ai' act as the vibhakti markers. The Vibhakti information helps in mapping of semantic relations. This phenomenon allows the language to have a relatively free word-order. It can be seen that both sentences convey the same meaning, possibly with a different emphasis - which is clarified only from context. The difference in the way the markers attach to the words in Hindi and Tamil should also be noted. These subtle phenomena make translation between languages from different language families especially challenging. We specifically discuss the challenges faced when translation between an Indo-Aryan and a Dravidian language in more detail in Chapter 5.

3. **Sandhi** - This is a particular characteristic which leads to merging of words into a composite word. For example:
   (a) 'हिमालय' ('Himalaya', meaning 'Abode of snow') is a composite of 'हिम' ('Heem', meaning snow) and 'आलय' ('Aalay', meaning abode).
   (b) 'योगासन' ('Yogasana', meaning 'A yogic posture') is a composite of 'योग' ('Yoga', meaning union) and 'आसन' ('Asana', meaning posture).
   (c) 'சொகுசுப்பேருந்து' ('Cokucuppēruntu', meaning 'Luxury bus') is a composite of 'சொகுசு' ('cokucu', meaning comfort) and 'பேருந்து' ('pēruntu', meaning bus).

(a) and (b) are compounds and may not pose much problem. However, for highly agglutinative languages such as Dravidian languages (c); any two words can combine if the conditions for Sandhi are met, making it much more challenging to handle.

4. **Verbal inflections** - Indian languages depict significantly rich verbal inflections, which represent important grammatical information like person, number, gender, case and so on. For example, The following sentences depict the information represented by the inflections 'ता' ('tā'), 'गा' ('gā') and 'ள' ('āḷ') :

   * मैं खेलता हूँ |
     main khel-tā hoon.
     I play.
     Here, 'tā' represents the following information : singular, male and participle.
   * वह नाचेगा |
     vaha naache-gā.
     He will dance.
     Here, 'gā' represents the information of singular, male and future tense.
   * அவள் வந்தாள்
     avaḷ vantāḷ.
     She came.
     Here 't' represents past aspect and 'āḷ' indicates that the subject of the action is female, singular.

Apart from the above-mentioned features, we also use two generic features - lemma and chunk heads. We train an NMT model over the ILCI corpus for 110 language pairs after performing features addition to the corpus. This helps in easier extraction of language constructs from the corpus, leading to faster learning and convergence. We call the resultant model as NMT_f. We observed significant gain in performance over a baseline NMT model (NMT_{Base}) trained over the same corpus. In order to compare our results with the state-of-the-art, we train a phrase based SMT model using the same corpus. The SMT model is trained using Moses [32] for phrase extraction and lexicalized reordering as described in [34][1]. We call this model SMT_{SA}. Table 5.1 compares the results obtained by NMT_f and SMT_{SA} on the ILCI test set. We discuss these in Chapter 5.

[1]We train our own SMT model since the training, validation and testing sets used by Sata-Anuvadak are unavailable to us.

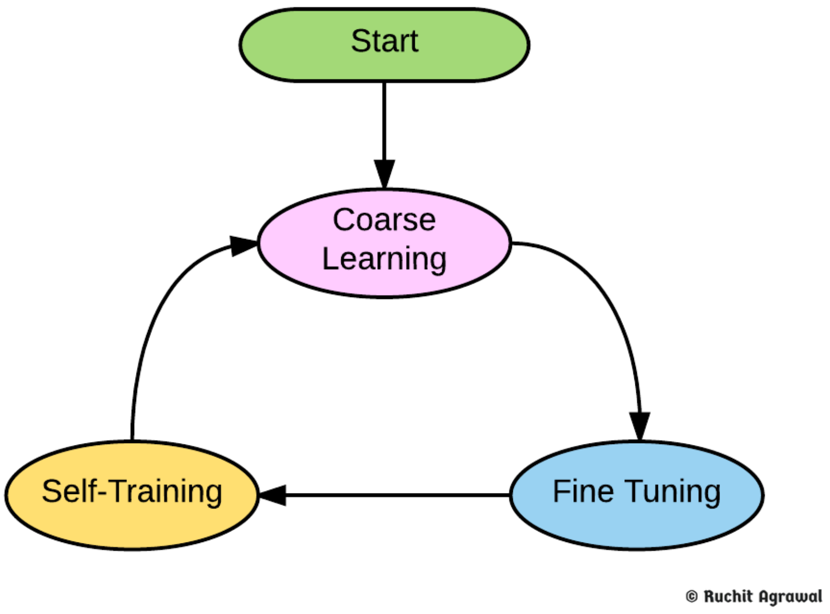


Figure 4.1 *Three-phase approach to improve robustness and accuracy. The entire cycle is repeated until the increase in accuracy is minimal. We conduct three self-training iterations.*

4.2 Three-phase Training to improve Domain Coverage and Accuracy

Although NMT_f shows significantly better performance than NMT_{Base} on in-domain data, the scores drop significantly when dealing with data from other contextually distant domains (discussed more in Chapter 5). In this section, we describe a three-phase integrated approach which leverages a large monolingual corpus of the source language and an existing MT tool to improve translation accuracy as well as domain coverage.

Figure 4.1 shows the block diagram of this approach. The entire process is divided into three stages : **Coarse learning**, **Fine-tuning** and **Self-training**. We begin by Coarse Learning, which can be thought of as providing the neural model with some information about grammatical constructs of the target language. The second phase employs Fine-tuning to enrich the linguistic knowledge of the model with the help of a hand-annotated gold parallel corpus. This is then followed by self-training, where the fine-tuned model is employed to generate a synthetic corpus again, on which we perform Coarse Learning for the next training iteration. Thus, this is a cyclical process, which is stopped when further increase in accuracy is observed to

be negligible.
The following sections explain the three phases in detail :

4.2.1 Coarse Learning

Coarse Learning is a form of weak supervision, which is a machine learning paradigm where the model learns from noisy data or prior knowledge. [23] used rich syntactic and semantic features to induce prior knowledge for the task of coreference resolution. [46] uses an ensemble of weak learners using rules to identify biomedical entities from medical documents.

Large annotated parallel corpora are not easy to obtain for Indian languages. However, it is easier to use an existing MT system to generate a sub-optimal translation of a monolingual corpus, which is referred to as synthetic data.
Building upon this insight, we generate a synthetic corpus for 10 language pairs[2] using *Sampark* [2] to translate the EMILLE monolingual corpora. We use this tool rather than $Sata - Anuvadak$ [34] due to its uniform domain coverage - a trait desirable for synthetic data generation when dealing with multiple domains
We train an NMT model over this synthetic corpus after performing feature addition as described in Section 4.1. This helps the model to learn significant linguistic information about the target language in the form of syntax, word order and morphology, along with the vocabularies, although with certain noise. The resulting model would naturally not perform with high accuracy, but it adds sufficient vocabulary and serves as a baseline to improve upon in further phases. We call the resulting model as NMT_{Coarse}. NMT_{Coarse} including both the encoder and decoder is jointly trained to maximize the conditional log likelihood of the synthetic corpus as shown in Equation 3.

$$\max_{\theta_w} \frac{1}{N_w} \sum_{j=1}^{N_w} \log p_{\theta_w}(y_w^{(n)} | x_w^{(n)}) \tag{4.1}$$

where $(y_w^{(n)}, x_w^{(n)})$ represents the $n - th$ sentence in the weak corpus of size N_w and θ_w denotes the set of all tunable parameters. The dropout and learning rate are kept high whereas the number of epochs is kept low since the primary motive for coarse learning is to learn only the general characteristics of the target language from the synthetic corpus, thereby making it easier to fine-tune the model. Detailed parameters used are provided in Table 4.1.
[45] proposes a rule-based MT system using bigram dictionaries. As part of future

[2]Language pairs for which both large monolingual corpora and *Sampark* were available.

work, this method can be employed in addition to our method to generate synthetic corpora for languages in which there is no existing MT tool available.

4.2.2 Fine-Tuning

This is the second and most important phase of our three-phase training approach. During this phase, a gold parallel corpus is needed.
This phase comprises of improving performance by fine-tuning the pre-trained model NMT_{Coarse} using the gold parallel corpus. This allows the model to be initialized with the weights learnt by the coarse model, rather than random weights.
In this phase, we employ the ILCI parallel corpus (with added linguistic features) for fine-tuning the pre-trained model - NMT_{Coarse}. This means that the low-data NMT model is not initialized with random weights, but with the weights learnt by the coarse model. The coarse model contains some amount of linguistic knowledge, in terms of lexical and semantic structure, word order and vocabulary. This information is imparted to the new model being trained using transfer learning. [62] uses transfer learning to increase translation quality between resource scarce language pairs by incorporating the weights learnt during training for high resource language pairs. It was also found that languages having similar structure, like Fr $\longleftrightarrow$ En (French - English) showed better improvement in performance as compared to other languages having little similarity, like Uz $\longleftrightarrow$ En (Uzbek - English). Our approach is based on the intuition that transfer learning between the same language pair should perform better than its multilingual counterpart. Our experiments confirm this (Chapter 5). During fine tuning, the change in weights in each epoch learnt through transfer learning allows the model to align more towards the correct model. This is because the quality of the corpus employed during this phase is significantly better than the quality of the corpus employed for phase 1, i.e. Coarse Learning. However, since the size of this corpus is lesser, it is not a good idea to train the model directly on this corpus. This is evident from the scores obtained by $Base_{NMT}$, the baseline NMT model trained only on the ILCI corpus.
We call the model generated after fine-tuning NMT_{FT}. Table 5.2 gives the results obtained by NMT_{FT}. Our experiments demonstrate that the quality of translation obtained using this technique is significantly better than SMT_{SA} as well as NMT_f. The hyper-parameters for training the model are carefully tweaked to achieve optimum performance. For example:
We use lower dropout and learning rate but considerably higher number of epochs in this phase as compared to Coarse Learning. This is done since the emphasis in this phase is to fine-tune the already learnt language characteristics and further learn new ones from the gold data.

Coarse Learning and Fine Tuning combined can be visualized as weakly supervised learning for our NMT model. Weak supervision is a technique of learning from noisy data or prior knowledge. [23] used rich syntactic and semantic features to induce prior knowledge for the task of coreference resolution. [46] uses an ensemble of weak learners using rules to identify biomedical entities from medical documents.

4.2.3 Self-Training

Self-training is a form of semi-supervised learning, which is a technique of using both labelled and unlabelled data to improve the performance of a machine learning system. Self-training [11] involves iteratively classifying unlabelled data using a classifier trained on labelled data. The unlabelled data classified with highest confidence is used to further create the classifier along with the labelled data.

As part of the self-training stage, we generate a synthetic corpus using the fine-tuned model from the previous cycle. For example: NMT_{FT} from the first cycle is now used to translate the monolingual corpus rather than *Sampark* [2]. Coarse learning is then performed using this synthetic corpus as training data. This leads to better accuracy during coarse learning for the second cycle as compared to the previous iteration due to lesser noise in the synthetic corpus. The coarse model thus generated is again fine-tuned using the ILCI corpus. This forms one iteration of self-training. This entire cycle is repeated until there is minimal increase in translation accuracy.

This is an effective method specially when employed in the proposed three-phase training pipeline, since the quality of the synthetic data generator used during the first phase heavily influences the trasnlation accuracy. Since the fine-tuned model has a better quality than a rule-based or statistical MT system, we see significant gains on employing self-training. The effect of the quality of synthetic data generator is discussed in more detail in Chapter 5.

The number of cycles to be performed for Self-Training (and in effect three-phase training) depends on the sizes of the monolingual corpus employed in the first phase as well as the parallel corpus employed in the second phase. If the latter is especially large in size, more self-training iterations can be performed. The size of our parallel corpus is 50,000 sentences. We perform three self-training iterations for our experiments since there was minimal to no increase in BLEU scores after that. The resultant model after three iterations is called NMT_{ST}. The results obtained by NMT_{ST} are given in Table 5.2 and discussed in Chapter 5.

Confidence estimation : OpenNMT [30] generates a prediction score for each translation, which is the cumulated log likelihood of the generated sequence. We use a threshold of -5.0 to filter out the low confidence translations. This ensures that the synthetic corpus employed for Coarse Learning in Training iteration 2 is of much better quality than the previous iteration. We observe improvement in scores by 2-5 percentage on employing this method, as opposed to using the same size of synthetic corpus in each training iteration.

4.3 Parameter Tuning

We conduct experiments with different set of parameters and choose the ones producing optimum results for the experimentation tracks described in the paper.

4.3.1 Bidirectional RNN vs Unidirectional RNN

We experiment with unidirectional as well as bidirectional RNN's and oberve that better results are produced by the latter, as can be seen from Figure 4.2.

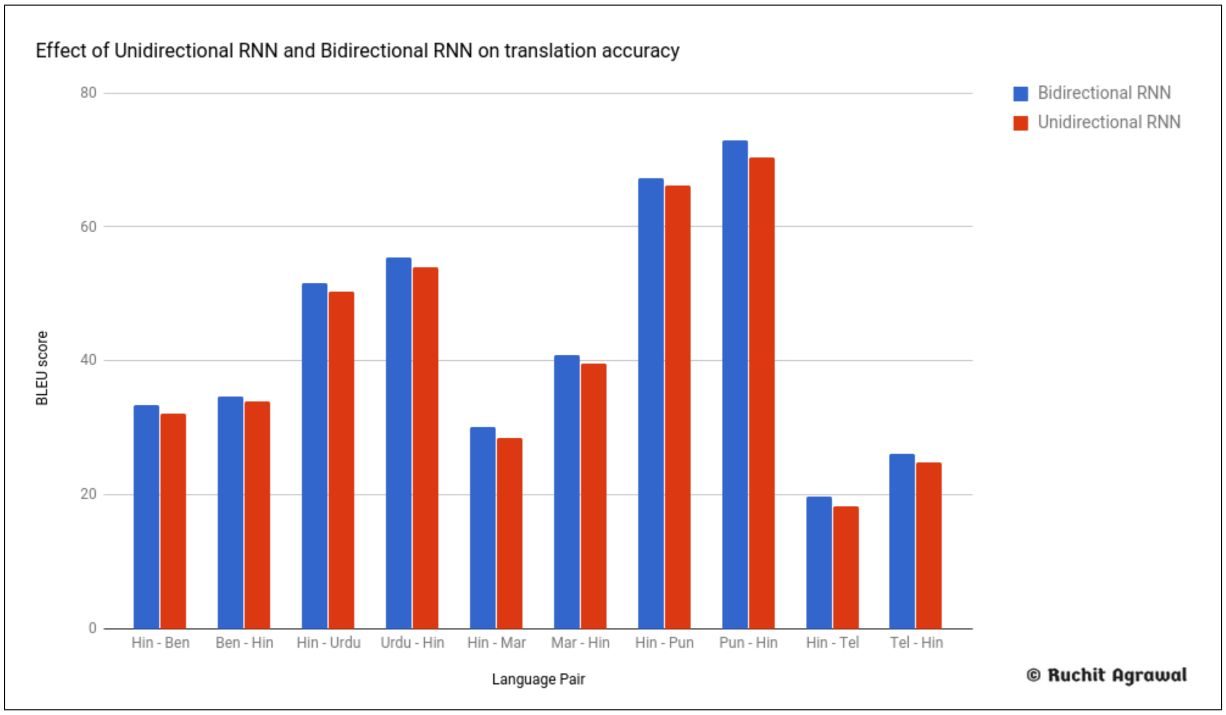


Figure 4.2 Results obtained on Unidirectional and Bidirectional RNN's.

We performed grid search to obtain best set of hyper-parameter with validation data for each phase including learning rate, learning decay-rate and drop out. We did hyper-parameter tuning for Hindi-Gujarati language pairs and used the same parameters values for corresponding to each phases for all the other language pairs. Some of the parameters like optimisation function, word vector size and brnn parameters were set to the default values. Detailed set of parameters used is provided in Table 4.1.

Table 4.1 Detailed parameters for training the NMT models

Phase	Parameters						
	Sample	WordVecSize	Layers	Dropout	Learning rate	LR decay	Start Decay at
Baseline	80%	500	2	0.2	0.76	0.325	10
Baseline+Features	80%	500	2	0.2	0.8	0.25	10
Coarse Learning	60%	500	2	0.55	0.9	0.75	5
Fine Tuning	80%	500	2	0.3	0.5	0.15	10
Coarse ST1	50%	500	2	0.55	0.9	0.75	5
Fine ST1	80%	500	2	0.2	0.8	0.25	10
Coarse ST2	50%	500	2	0.4	0.8	0.6	5
Fine ST2	80%	500	2	0.15	0.3	0.326	10
Coarse ST3	50%	500	2	0.4	0.8	0.6	5
Fine ST3	80%	500	2	0.3	0.5	0.15	10

4.3.2 Number of epochs

For NMT_{base}, NMT_{FT}, NMT_f and NMT, 4.4. It can be observed that performance reaches peak after 50. Thus we kept the number of epochs to 60.

Also, it can be observed from Figure 4.3, the value of perplexity reaches optimum minimum at 30 for coarse training. The reason behind this is that during coarse learning, we want only strong linguistic features of target langauge to be learnt by the model Hence, we train it for smaller epochs, where increase in perplexity is reduced to a threshold, which was 5 here.

4.3.3 Number of layers

We observe that the best translation quality is obtained using two layers of encoder-decoder architecture. Although increasing the number of layers allows learning of more complex higher level representation for language translation, it induces increase in the number of parameters exponentially. With increase in the number of parameters, the quantity of corpus required for training increases proportionately. It was observed from the size of our dataset that the best balance between the level of

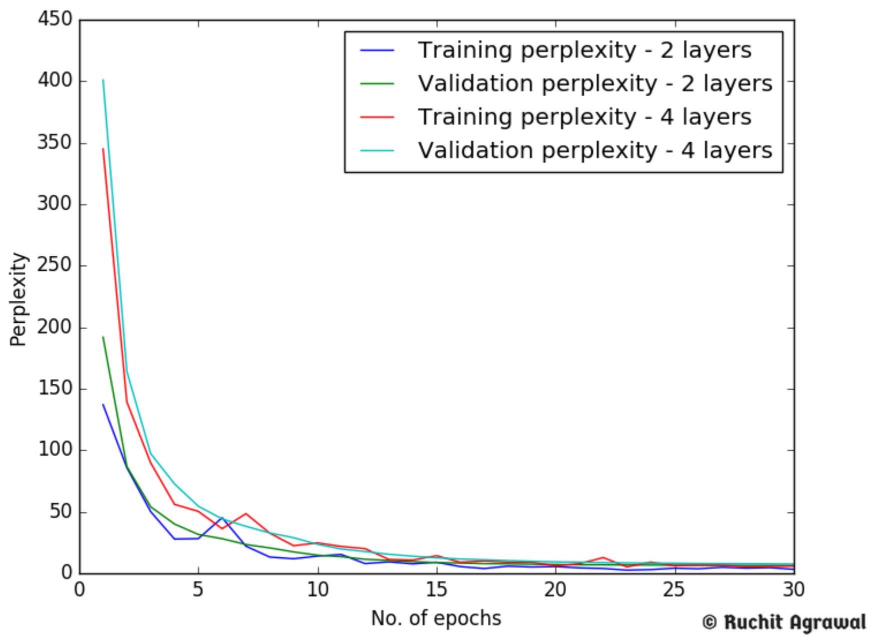


Figure 4.3 *Perplexity curve for 2-layered and 4-layered architecture vs. Number of Epochs*

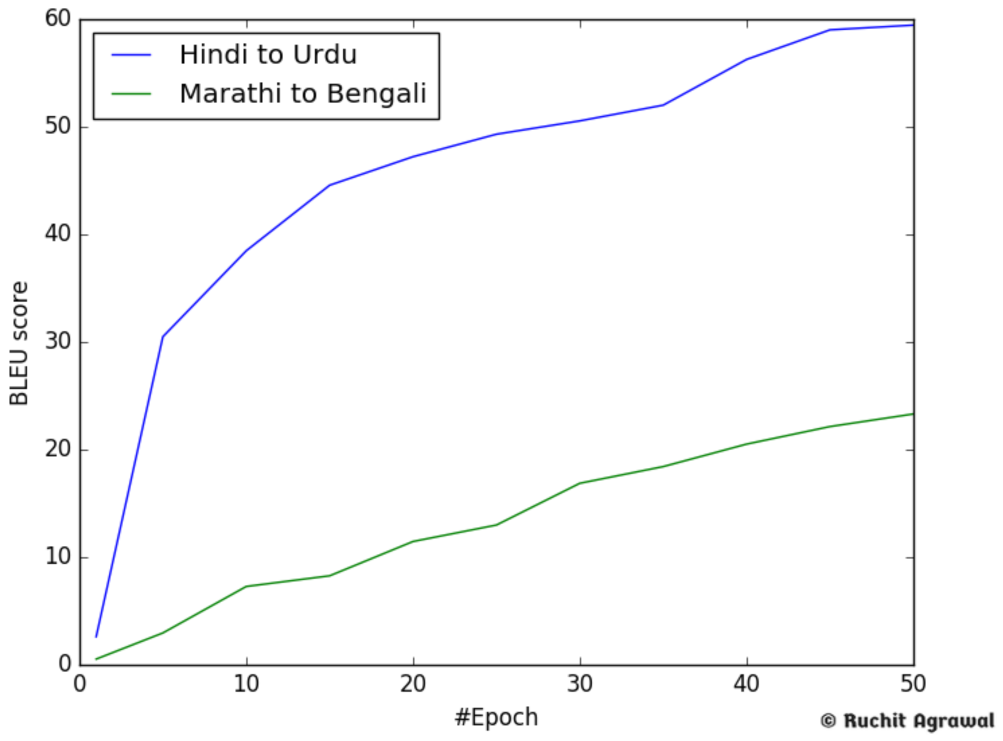


Figure 4.4 *Effect of number of epochs on translation performance.*

representation for language translation and number of parameters of neural network was obtained with two layers. Hence we use a two-layered architecture throughout the experimentation.

4.4 Summary

In this chapter, we described the major theoretical contributions of this study. We proposed two methods to increase the effectiveness of Neural Machine Translation in the scenario of Indian language to Indian language translation. The training method can be chosen (or combined) depending upon the language pair under consideration. We analyze the effectivess of both methods in the next chapter.

Chapter *5*

Evaluation - Results and Analysis

This chapter discusses the evaluation of our models. We report our results using the BLEU[44] metric. We see a gradual improvement in scores as we move from the baseline NMT model to Method 1 (Incorporating linguistic information) and finally the best results are obtained using Method 2 (Three-phase training).

5.1 Results on the ILCI test set

Table 5.1 shows that the results obtained by NMT_f are comparable to SMT_{SA}, although NMT systems are more data hungry. We observe that although NMT models are good at learning language constructs from the parallel corpus itself, exploiting additional linguistic information in the form of features - specially in low data conditions, provides further improvement in performance.

The scores obtained by NMT_f follow a general trend - better performance for Indo-Aryan Languages and considerably poor performance for Dravidian Languages. The primary reason behind this can be attributed to larger structural similarity among Indo-Aryan Languages as well as lesser inflections as compared to Dravidian Languages, which are much more agglutinative in nature.

We can observe from Table 5.2 that a significant gain in scores is observed on employing three-phase training. Figure 5.1 shows the incremental increase in performance with number of epochs. We can see that NMT_{ST} shows faster convergence owing to better initializations. The performance of NMT_{ST} is comparable to [33], the state-of-the-art in phrase-based SMT for language pairs involving related languages. We obtain either nearly equal or higher gain in scores over SMT_{SA} as they obtain over Sata-Anuvadak [34].

Table 5.3 shows the results obtained by SMT_{SA}, NMT_{Base}, NMT_f, NMT_{FT} and NMT_{ST} on test sets from different domains. We see improvement in accuracy as

Table 5.1 Comparison of NMT_f with SMT_{SA} in terms of BLEU score

		hin	**urd**	**pan**	**ben**	**guj**	**mar**	**kok**	**tam**	**tel**	**mal**	**eng**
SMT_{SA}	**hin**	-	50.1	70.13	36.69	53.45	33.5	35.63	11.64	21.54	10.4	27.87
NMT_f		-	**51.04**	**71.01**	36.34	**53.65**	**33.74**	35.07	10.61	20.57	8.86	27.76
SMT_{SA}	**urd**	57.51	-	52.3	26.36	39.08	20.7	24.84	8.36	14.9	7.92	20.64
NMT_f		**58.91**	-	**53.81**	26.04	**40.43**	20	24.55	7.11	13.62	6.15	19.85
SMT_{SA}	**pan**	70.83	44.75	-	30.25	46.33	25.55	29.87	9.25	18.03	7.25	24.21
NMT_f		**71.59**	**44.87**	-	29.38	**46.63**	24.86	**30.14**	7.47	16.82	5.86	**24.3**
SMT_{SA}	**ben**	36.4	24.67	31.61	-	31.13	19.84	23.18	8.68	13.6	8.94	18.44
NMT_f		**37.56**	**25.31**	**32.31**	-	**31.62**	19.75	**23.36**	7.21	12.03	7.96	17.89
SMT_{SA}	**guj**	52.98	34.33	47.61	28.99	-	26.51	29.17	9.35	16.57	7.64	19.42
NMT_f		**53.32**	**34.88**	**48.75**	**28.85**	-	25.83	**29.3**	7.86	14.59	6.04	**19.62**
SMT_{SA}	**mar**	41.97	24.99	34.51	23.89	33.54	-	27.77	8.34	12.34	7.63	16.11
NMT_f		**43.02**	**26.38**	35.44	**24.37**	**34.77**	-	**27.9**	6.82	10.96	6.11	16.01
SMT_{SA}	**kok**	38.59	25.86	33.26	24.68	31.44	23.31	-	7.41	13.25	8.41	16.93
NMT_f		**39.15**	**26.01**	**33.53**	23.87	**32.76**	**23.4**	-	5.55	11.51	7.39	17.46
SMT_{SA}	**tam**	21.87	15.96	19.19	14.94	17.09	11.21	14.18	-	9.13	6.61	10.7
NMT_f		20.52	14.27	17.91	13.35	15.78	9.45	12.44	-	8.17	5.93	10.2
SMT_{SA}	**tel**	27	19.24	24.89	16.98	22.02	13.06	17.09	7.08	-	6.76	11.98
NMT_f		25.98	18.19	23.65	15.36	20.24	11.81	16.13	5.97	-	5.89	10.48
SMT_{SA}	**mal**	13.9	10.44	12.08	10.31	10.64	7.03	8.83	4.98	6.7	-	8.2
NMT_f		12.56	9.28	10.81	8.96	9.61	5.93	7.27	4.13	6	-	7.88
SMT_{SA}	**eng**	26.84	17.53	22.4	14.24	17.14	10.47	13.14	4.18	5.96	5.15	-
NMT_f		**27.24**	**18.94**	**23.19**	**14.76**	**17.83**	**10.56**	**13.31**	2.95	4.34	3.76	-

well as coverage - discussed below:

Two-phase vs. Three-phase Training : Accuracy vs. Coverage

Since the large monolingual corpus contains data from a variety of domains, NMT_{Coarse} develops a significantly big vocabulary, which leads to lesser number of Out of Vocabulary (OOV) words on out-of-domain data, as compared to NMT_f and SMT_{SA}. The word order and lexical constructs learnt during coarse learning are retained and improved upon fine-tuning on the gold corpus.

NMT_{FT} exhibits best domain coverage results as can be seen from Table 5.3. This suggests that two-phase training obtains best results on out-of-domain data. Three-phase training includes self-training as well - it produces best results on in-domain data as can be observed from Table 5.2). Since the fine-tuned model is used to generate the synthetic corpus for the next self-training iteration, the quality of synthetic corpus thus obtained is higher than the one used during the previous iteration. Better synthetic data leads to better fine-tuning. This explains overall increase in accuracy after self-training over the ILCI test set. However, the coverage is affected a little. The reason can be attributed to a slight development of bias towards the

Table 5.2 Performance Comparison during various phases over ILCI test set in terms of BLEU scores

		urd	**pan**	**ben**	**guj**	**tam**
NMT_{FT}	**hin** $\Rightarrow$	52.93	72.57	37.75	54.87	11.94
NMT_{ST}		53.95	73.71	38.77	55.52	12.27
NMT_{FT}	**hin** $\Leftarrow$	60.22	73.2	38.97	54.64	22.04
NMT_{ST}		61.33	73.63	39.31	55.14	22.37

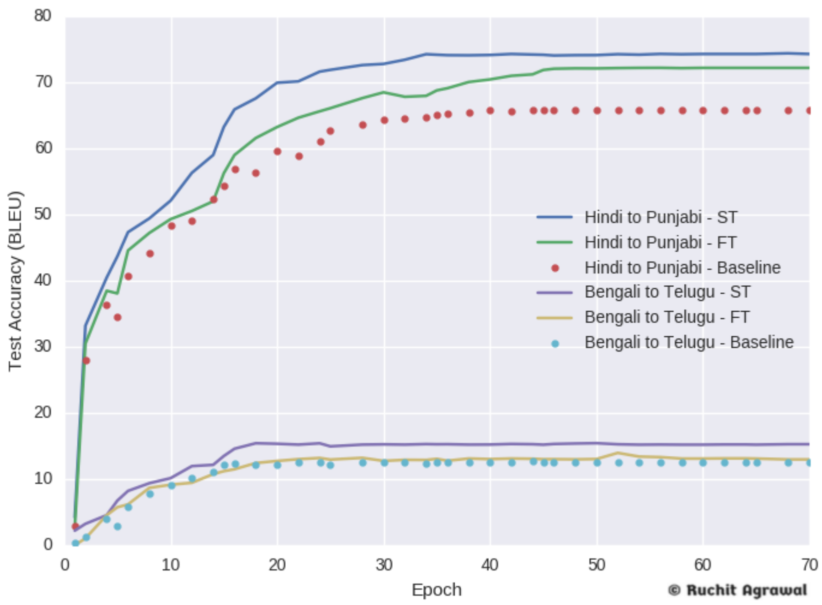


Figure 5.1 *Performance gain observed using three-phase approach in terms of BLEU scores. Self-training results are shown for the third iteration.*

health and tourism domains due to iterative fine-tuning. The domain coverage of three-phase training is still significantly better than SMT_{SA} and NMT_f.

We conclude that the two-phase approach (Coarse Learning + Fine-Tuning) is more suitable for out-of-domain data, whereas the three-phase approach is better suited to translate in-domain data.

5.2 Results on test sets from different domains

We test the coverage of our model after three-phase training on test sets from different domains. We extract data samples from Education and Social domains respectively from the EMILLE parallel corpus. We use these samples as test sets to evaluate the coverage of our models. We show the results of our models an these domains in Figure S2 and Figure S3.

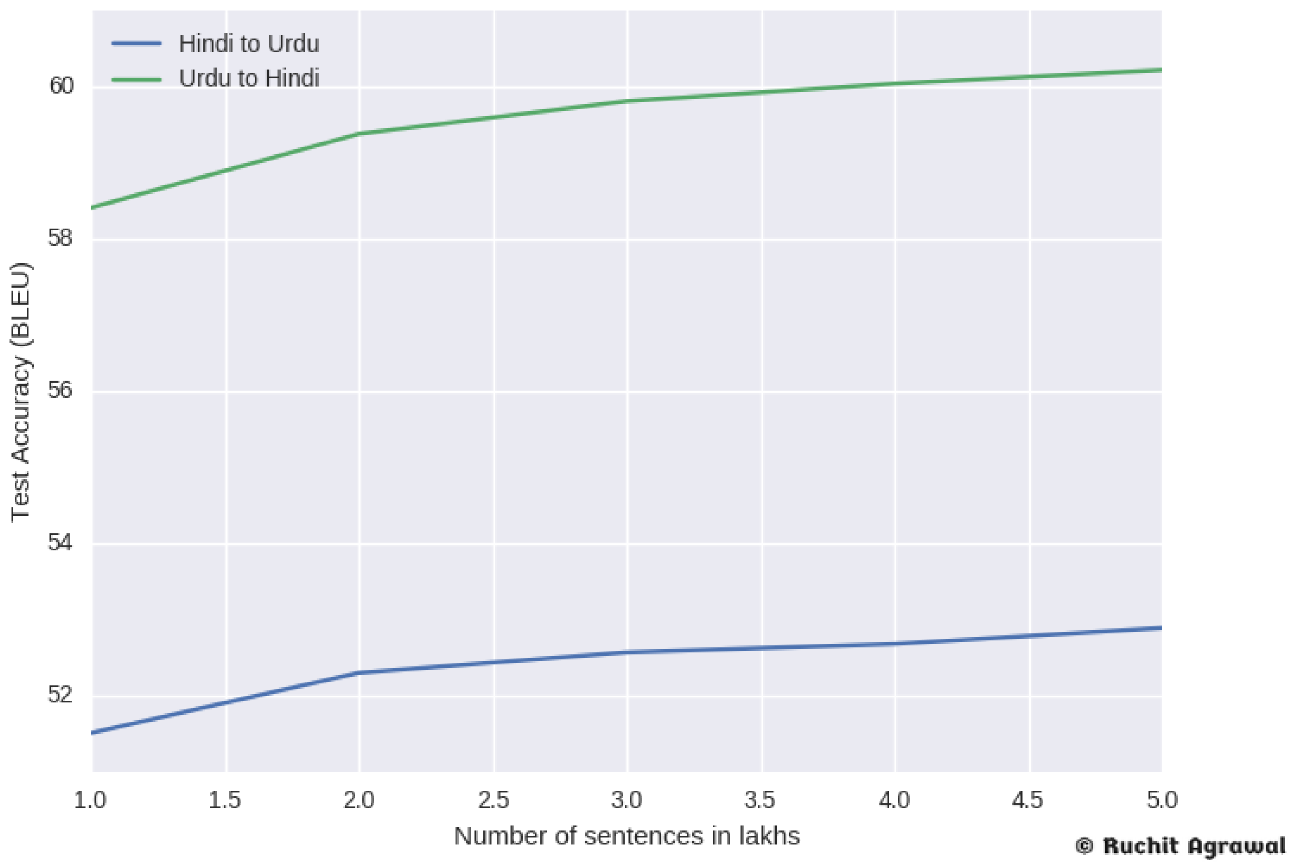


Figure 5.2 *Effect of size of monolingual corpus used during Coarse Learning on accuracy of* NMT_{FT} *for the language pair hin<->urd in terms of BLEU score*

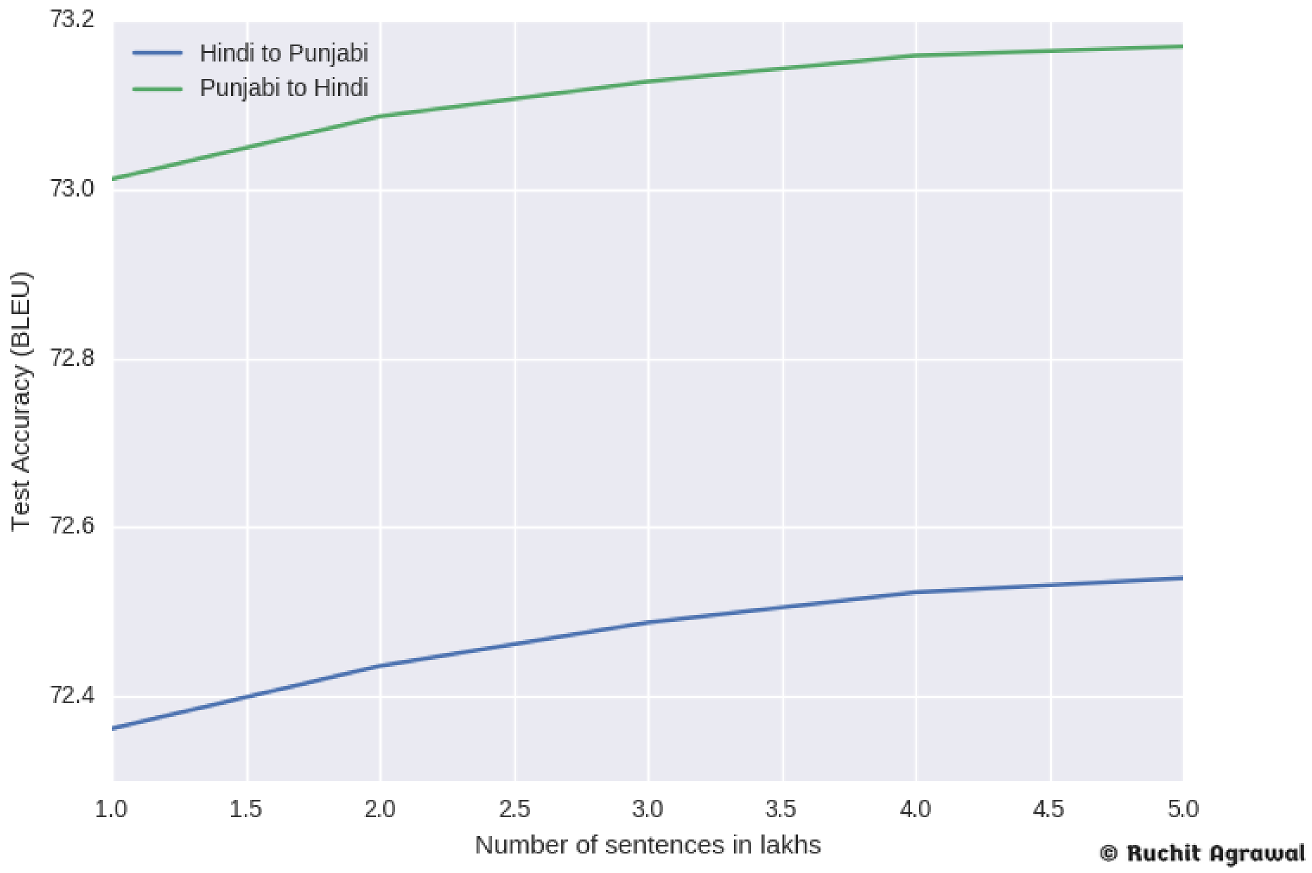


Figure 5.3 *Effect of size of monolingual corpus used during Coarse Learning on accuracy of* NMT_{FT} *for the language pair hin<->pun in terms of BLEU score*

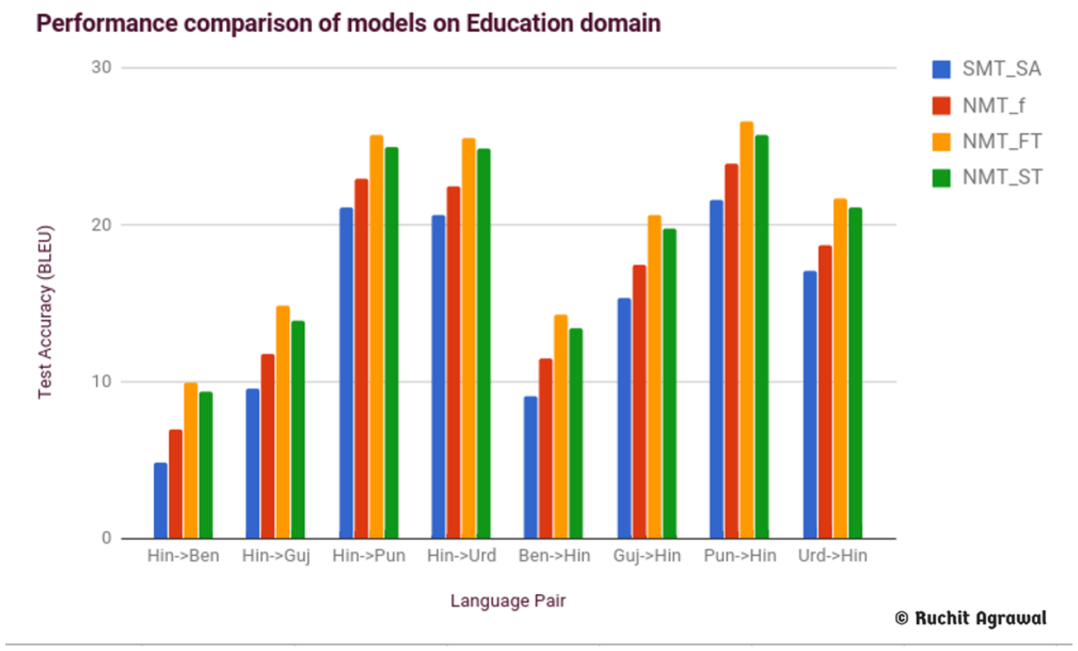


Figure 5.4 *Coverage results on education domain*

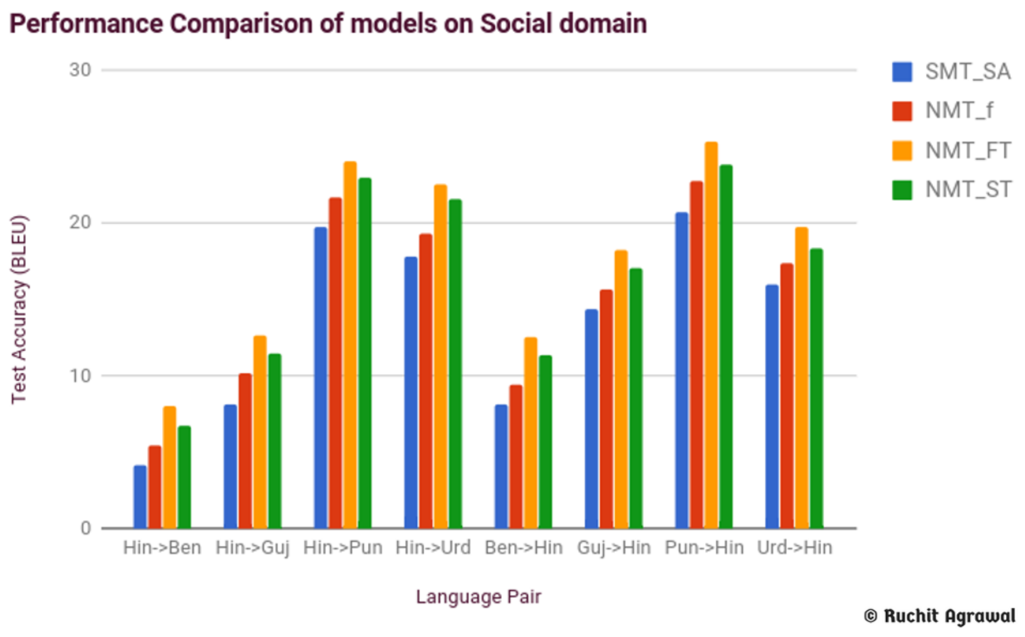


Figure 5.5 *Coverage results on social domain*

Table 5.3 Robustness comparison of models over different domains (in terms of BLEU scores)

		Housing				**Legal**			
		pan	**guj**	**urd**	**ben**	**pan**	**guj**	**urd**	**ben**
SMT_{SA}	**hin** $\Rightarrow$	16.45	11.46	18.11	3.62	15.13	8.93	17.75	1.83
NMT_f		17.48	13.23	19.53	4.74	16.42	11.35	19.02	2.89
NMT_{FT}		**23.71**	**17.62**	**24.49**	**13.22**	**22.27**	**14.07**	**25.41**	**7.7**
NMT_{ST}		22.69	16.92	22.23	11.03	19.13	13.29	23.69	6.1
SMT_{SA}	**hin** $\Leftarrow$	13.85	12.73	14.78	3.0	12.45	11.93	15.63	2.72
NMT_f		15.09	14.52	15.72	3.88	13.9	14.07	17.0	3.5
NMT_{FT}		**20.7**	**17.52**	**20.88**	**9.41**	**19.6**	**17.26**	**24.16**	**11.18**
NMT_{ST}		19.65	16.71	18.03	8.11	18.05	16.09	22.54	9.52

Table 5.4 Two different translations corresponding to the same English sentence - from ILCI test data (Many-to-many mapping between vocabulary)

$ILCI_{Test}$	ताजा साँसें और चमचमाते दाँत आपके व्यक्तित्व को निखारते हैं ।
Tl	taaja saansen aur chamachamaate daant aapake vyaktitv ko nikhaarate hain
Ts	Fresh breath and shining teeth enhance your personality .
NMT_{FT}	ताजी साँस और चमकदार दाँत आपके व्यक्तित्व में चार चाँद लगाते हैं ।
Ts	taajee saans aur chamakadaar daant aapake vyaktitv mein chaar chaand lagaate hain
Tl	Fresh breath and shining teeth enhance your personality.

5.3 Effect of quality of synthetic data generator

We experiment the usage of different MT systems (NMT_f, SMT_{SA} and *Sampark*) to generate the synthetic data from the monolingual corpus used during coarse learning. We show the results in Figure S4. We note that although SMT_{SA} has the best performance over the ILCI test set, *Sampark* has the most uniform domain coverage - leading to significantly higher increase in performance over NMT_f after fine-tuning. SMT_{SA} and NMT_f do not perform good - owing to the limited parallel corpora on which they are trained, leading to domain-specific accuracy but lesser coverage.

5.4 Fluency comparison using sample translations

We observe on manual inspection of samples that there is a significant improvement in the vocabulary as well as linguistic knowledge after coarse learning and fine-tuning, thereby producing quality translation as shown by the use of semantically correct synonyms. For example, Table 5.8 and Table 5.9 show the Tamil and Gujarati translations generated by the models for the input sentence "ताजा साँसें और चमचमाते दाँत आपके व्यिक्त्व को निखारते हैं ।" (Transliteration : Taaja saansein aur

Table 5.5 Evaluating output quality : Coarse learning vs. Fine tuning

$ILCI_{Test}$	इसका उपचार सभी अस्पतालों में है ।
Tl	isaka upachaar sabhee aspataalon mein hai
Ts	Its treatment is available in all hospitals.
NMT_{Coarse}	इसके लिए अब उपलब्ध भी एक गोली है ।
Tl	isake lie ab upalabdh bhee ek golee hai
Ts	For this, there is now available also a pill.
NMT_{Base}	इसका निदान सभी सभी अस्पतालों में उपलब्ध है ।
Tl	isaka nidaan sabhee sabhee aspataalon mein upalabdh hai
Ts	The solution for this is available in all all hospitals.
NMT_{FT}	उसका इलाज सभी अस्पतालों में उपलब्ध है ।
Ts	usaka ilaaj sabhee aspataalon mein upalabdh hai
Tl	The treatment for that is available in all hospitals.
NMT_{ST}	इसका उपचार सभी अस्पतालों में उपलब्ध है ।
Tl	isaka upachaar sabhee aspataalon mein upalabdh hai
Ts	The treatment for this is available in all hospitals.

Table 5.6 Evaluating output quality : Fine tuning vs. Self-Training

$ILCI_{Test}$	अपनी रोज की दिनचर्या में व्यायाम को जरूर शामिल करें ।
Tl	apanee roj kee dinacharya mein vyaayaam ko jaroor shaamil karen
Ts	Do include exercise in your daily routine.
NMT_{FT}	एक्सरसाइज को अपने दैनिक दिनचर्या में शामिल करें ।
Tl	eksarasaij ko apane dainik dinacharya mein shaamil karen
Ts	Include exercise in your everyday routine.
NMT_{ST}	व्यायाम को अपनी दैनिक दिनचर्या में शामिल करें ।
Tl	yaayaam ko apanee dainik dinacharya mein shaamil karen
Ts	Include exercise in your everyday routine.

chamchamaate daant aapke vyaktitva ko nikhaarte hain; Translation: Fresh breath and shining teeth enhance your personality) during different training phases. An incremental improvement in translation quality can be observed. The fine tuned output for Gujarati displays an impressive usage of the phrase "સુશોભિત કરી દે છે" -- a contextually suitable and semantically correct idiom in Gujarati which conveys “enhancing of personality”. However, since the words used are different from the test sentence, this translation will be heav- ily penalised by the scoring mechanism, although it is correct. Apart from this, the model learns the correct gender for 'તાજો' (fresh), which is Masculine in Gujarati, unlike Hindi (Feminine) ('ताजी'). This is in alignment with the expectation of better grammar learning through three-phase learning.

Table 5.7 Evaluating output quality : Two layers vs. Four layers

$ILCI_{Test}$	40 साल से अधिक आयु के सभी व्यक्तियों की वार्षिक जाँच अवश्य की जानी चाहिए ।
Tl	40 saal se adhik aayu ke sabhee vyaktiyon kee vaarshik jaanch avashy kee jaanee chaahie
Ts	An annual check-up of everybody above the age of 40 years must be done .
$NMT_{Coarse_{2l}}$	40 वर्षों की उम्र के ऊपर प्रत्येक व्यक्ति की वार्षिक जाँच करनी होगी ।
Tl	40 varshon kee umr ke oopar pratyek vyakti kee vaarshik jaanch karanee hogee
Ts	An annual checkup of each person above 40 years of age will have to be done.
$NMT_{Coarse_{4l}}$	40 वर्षों की उम्र के ऊपर प्रत्येक व्यक्ति का वार्षिक जाँच किया जाना चाहिए ।
Tl	40 varshon kee umr ke oopar pratyek vyakti ka vaarshik jaanch kiya jaana chaahie
Ts	An annual check-up of each person above 40 years of age should be done.
$NMT_{FT_{2l}}$	40 वर्ष की उम्र से अधिक प्रत्येक व्यक्ति की वार्षिक जांच निश्चित रूप से की जानी चाहिए ।
Tl	40 varsh kee umr se adhik pratyek vyakti kee vaarshik jaanch nishchit roop se kee jaanee chaa
Ts	After 40 years of age, every year an annual checkup should be done.
$NMT_{FT_{4l}}$	40 वर्ष की उम्र से अधिक प्रत्येक व्यक्ति की वार्षिक टेस्ट करनी चाहिए ।
Tl	40 varsh kee umr se adhik pratyek vyakti kee vaarshik test karanee chaahie
Ts	After 40 years of age, every one annual test should be done.

For Tamil, although the output quality is not completely comparable to a human translation, we observe that the incorrect grammar is corrected on incorporating features (Sentence 1 to Sentence 2). The usage of a more apt phrase "உங்கள் ஆளுமையை மேம்படுத்தும்" (Sentence 2 to Sentence 3) suggests the increased coverage of the NMT model after coarse learning and fine-tuning. The output after self-training uses a similar phrase to "enhancing your personality" which conveys - "You become an altogether new person", which is the closest way to express the idea of "personality enhancement" in Tamil. NMT_{ST}'s output is close to an good translation of the input sentence, showcasing the effectiveness of three-phase training - some grammar still needs to be corrected to reach human translation quality. This general trend is observed - Translation quality for IA languages is consistently better than Dravidian languages. We discuss the reasons behind this in Section 5.5.

5.4.1 Error Analysis

Since the evaluation metrics do not capture how well different linguistic phenomena are handled by our model, we perform a manual investigation and error analysis with the help of linguists. In order to have a clear insight of NMT performance as compared to SMT on various aspects, we do a side-by-side comparison of the output sentences generated by the SMT and the NMT models respectively. The linguists were asked to identify the strengths and weaknesses of NMT and SMT by ranking 200 output sentences produced by the respective models in terms of the following parameters:

Tag	Meaning
ADJ	Adjective
SG	Singular
PL	Plural
PRES	Present tense
PST	Past tense
FUT	Future tense
N.PST	Non past aspect
1P	First person
2P	Second person
3P	Third person
ACC	Accusative case
PRT	Particle
R.PART	Relative participial
CONJ	Conjunctive participial
AUX	Auxiliary verb
NH	Non human
INF	Infinitive

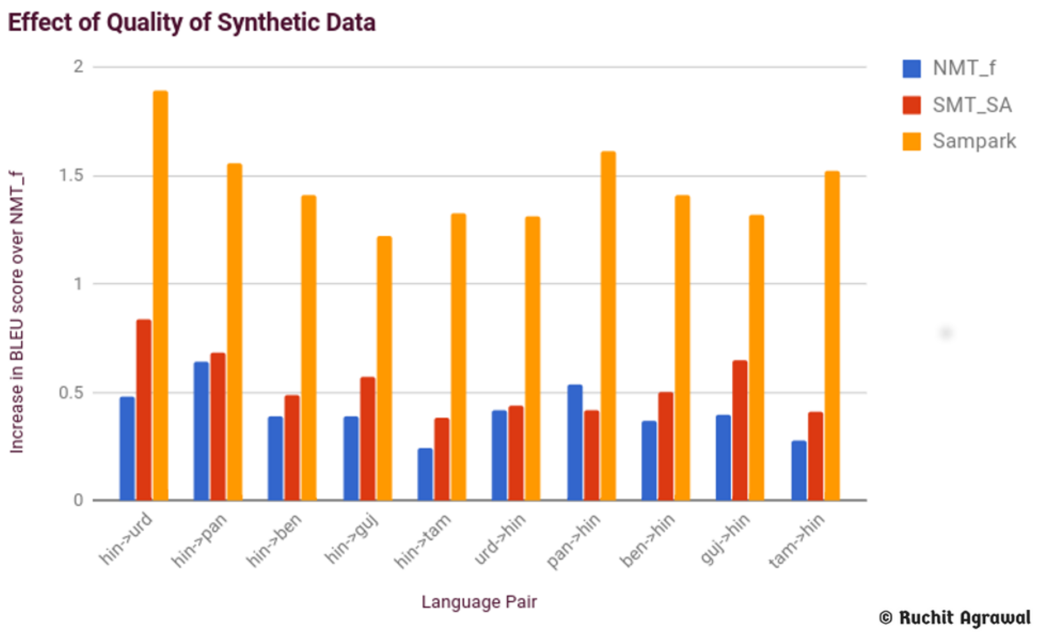


Figure 5.6 *Effect of quality of synthetic data generator. Best results are obtained using Sampark*

	Variant	Outputs of the NMT model at different training stages
T	O	புத்துணர்ச்சியான சுவாசம் மற்றும் பளபளப்பான பற்கள் தங்களின் தோற்றத்தை மேம்படுத்துகிறது.
	Tl	puttuṇarcciy-āṉa cuvācam maṟṟum paḷapaḷapp-āṉa paṟ-kaḷ taṅkaḷiṉ tōṟṟatt-ai mēmpaṭuttu-kiṟ-atu
	G	new-feeling-ADJ. breathing and glitter-ADJ. tooth-PL your appearance-ACC enhance-PRES-3.NH.SG
	Ts	Fresh breath and shining teeth enhances your appearance
P1	O	புதிய மூச்சு மற்றும் பளபளக்கும் பற்களை உங்கள் ஆளுமை அதிகரிக்க முடியும்
	Tl	putiya mūccu maṟṟum paḷapaḷa-kk-um paṟ-kaḷ-ai uṅkaḷ āḷumai atikari-kka muṭiyum
	G	new breath and glitter-N.PST-R.PART tooth-PL-ACC your personality increase-INF possible
	Ts	Your personality can increase new breath and shining teeth
P2	O	புதிய சுவாசமும் பிரகாசிக்கும் பற்களும் உங்கள் ஆளுமையை மேம்படுத்துகின்றன.
	Tl	putiya cuvācam-um pirakāci-kk-um paṟ-kaḷ-um uṅkaḷ āḷumaiy-ai mēmpaṭuttu-kiṉṟ-aṉa
	G	new breath-PRT shine-N.PST-R.PART tooth-PL-also your personality-ACC enhance-PRES-3.NH.PL
	Ts	New breath and shining teeth enhance your personality
P3	O	புத்தம் புதிய சுவாசமும் பளபளப்பான பற்களும் ஆளையே மாற்றி விடுமே.
	Tl	puttam-putiya cuvācam-um paḷapaḷapp-āṉa paṟ-kaḷ-um āḷaiyē māṟṟi viṭumē.
	G	newish-new breath-PRT glitter-ADJ tooth-PL-PRT person-ACC-PRT change-CONJ AUX
	Ts	Fresh breath and shining teeth changes the persona totally

Table 5.8 Outputs of our model at different stages : Tamil. *T: Test sentence, O : Model Output, G : Gloss, Tl : Transliteration, Ts : Translation*

	Variant	Outputs of the NMT model at different training stages
T	O	તાજા શ્વાસ અને ચમક્તા દાંત તમારા વ્યક્તિત્વને નિખારે છે.
	Tl	Tāja śvāsa anē camakatā dānta tamārā vyaktitvanē nikhare chē
	G	fresh-ML breath and shining-ADJ teeth-PL-ACC your personality enhance-INF is
	Ts	Fresh breath and shining teeth enhance your personality
Base	O	તાજી શ્વાસ અને ચમકદાર દાંત તમારી વ્યક્તિત્વ સુધારે છે.
	Tl	Tājī śvāsa anē camakadāra dānta tamārī vyaktitva sudhārē chē.
	G	Fresh-FL breath and shiny-ADJ teeth-PL your-FL personality-ACC improve is
	Ts	Fresh breath and shiny teeth improves your appearance
P1	O	તાજી શ્વાસ અને ચમકતા દાંત થી તમારી વ્યક્તિતવ સુધરે છે.
	Tl	Tājī śvāsa anē camakatā dānta thī tamārī vyaktitava sudharē chē.
	G	Fresh-FL breath and shining-ADJ teeth-PL due-to your-FL personality improve-INF is
	Ts	Your personailty is improved by fresh breath and shining teeth
P2	O	તાજો શ્વાસ અને ચમકતા દાંત તમારા વ્યક્તિત્વ ને નિખારે છે.
	Tl	Tājō śvāsa anē camakatā dānta tamārā vyaktitva nē nikhārē chē.
	G	Fresh-ML breath-PRT and shining-ADJ teeth-PL your-ML personality enhance PRES-3.NH.PL is
	Ts	Fresh breath and shining teeth enhance your personality
P3	O	તાજો શ્વાસ અને ચમકતા દાંત તમારા વ્યક્તિત્વ ને સુશોભિત કરી દે છે.
	Tl	Tājō śvāsa anē camakatā dānta tamārā vyaktitva nē suśōbhita karī dē chē.
	G	Fresh-ML breath and shining-ADJ teeth-PL your-ML personality pleasing does make-FV
	Ts	Fresh breath and shining teeth enhances your personality

Table 5.9 Outputs of our model at different stages : Gujarati. *T: Test sentence, G : Gloss, O : Model Output, Tl : Transliteration, Ts : Translation*

- * Word order
- * Morphology :
  - · How appropriate is the surface form selection
  - · Usage of correct syntactic structures
  - · Morphological agreement between words
- * Phrase handling :
  - · Non-translated phrases / phrases missing in the output
  - · Additional phrases - Phrases occuring in the output but not in the input source sentence
- * Lexical Choice - Quality and appropriateness of content words and terminology errors

We show the results in Figure 5.7.

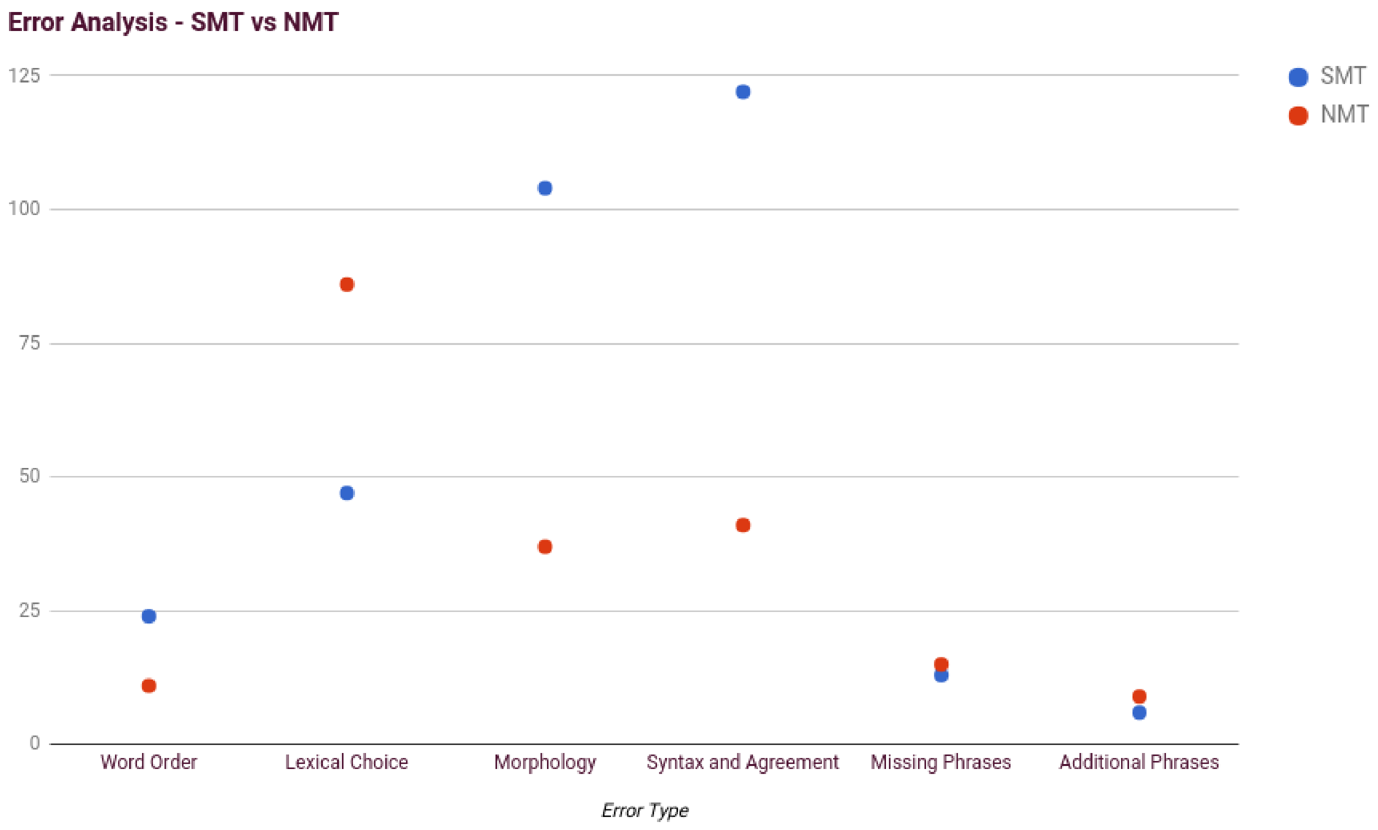


Figure 5.7 *Manual Error Analysis of performance of NMT with SMT*

It can be observed from Figure 5.7 that SMT produces about twice as more errors in word order and almost thrice as more errors in syntactic and morphological structures and agreement than NMT. Thus the NMT model is able to perform significantly better than SMT for these phenomena. This results in much more fluent translations produced by the NMT model - making it a better choice in most scenarios. At the same time, the errors made in terms of lexical choice are much more in NMT than SMT. NMT also produces slightly greater number of errors in terms of missing or additional phrases. On deeper investigation, it is made clear that a majority of the lexical choice errors are due to the noise present in the training data. This leads to the insight that NMT is more prone to greater sensitivity to training noise than SMT.

To summarize, NMT performs better than SMT in most linguistic aspects, particularly in the presence of a high quality training corpus.

5.5 Comparison of performance on Indo-Aryan and Dravidian languages

We calculate the average percentage increase in BLEU score for both approaches with respect to the following three categories:
1. Indo-Aryan to Indo-Aryan language translation (IA-IA)
2. Indo-Aryan to Dravidian language translation (IA-DR)
3. Dravidian to Dravidian language translation (DR-DR)

These are detailed in Table 5.10[1].

Table 5.10 Average percentage increase in scores for the proposed methods on different language families

Language Families	Method 1 (NMT_f)	Method 2 (NMT_{ST})
IA-IA	2.45	2.67
IA-DR	1.79	1.86
DR-DR	1.98	-

Even though Indian languages are all typologically SOV, there are distinct syntactic peculiarities in Dravidian languages (DR) that makes MT challenging between Indo-Aryan (IA) and Dravidian languages. Two such phenomena are shown by the examples below:

1. * Hindi Sentence : राम ने बोला कि वह घर जा रहा था

[1]Method 2 scores for the third category are unavailable, as mentioned in Chapter 3

   * Transliteration : rām nē bōl-ā ki vah ghar jā rahā thā
   * Gloss : Ram ERG tell-PST S.CONJ 3.SG.D.PRON home go AUX1-CONT AUX2-PST
   * Meaning : Ram said that he is going home.
2. * Telugu Sentence : రాముడు తాను ఇంటికి వెళ్తున్నట్టుగా చెప్పాడు
   * Transliteration : rāmuḍu tānu iṇṭi-ki veḷ-tunn-aṭṭugā cepp-ā-ḍu
   * Gloss : Ram 3P.REFL.PRON home-DAT go-PRES-MANNER.ADV tell-PST-3.M.SG
3. * Tamil Sentence : ராமன் தான் வீட்டுக்கு செல்வதாக கூறினான்
   * Transliteration : rāmaṉ tāṉ vīṭṭu-kku cel-vat-āka kūṟ-iṉ-āṉ
   * Gloss : Ram 3P.REFL.PRON home-DAT go-NPST.R.PART-MANNER.ADV tell-PST-3.M.SG

Above example shows that in Hindi the main clause is followed by subordinate clause and both the clauses are connected by a subordinating conjunction 'ki'. in Telugu and Tamil (Dr), the subordinate clause is embedded within the main clause and connection between them is established morphologically through adverbial inflections or sometimes a quotative marker is used to connect the two clauses. These phenomena explain the relatively lower performance on Dravidian languages as compared to Indo-Aryan languages.

5.6 Summary

This chapter analyses the results obtained by our models on multiple test sets from different domains. We observe that three-phase training is an effective method which produces good performance in terms of accuracy as well as domain coverage. The scores obtained thus are either nearly equal or bettter than the state-of-the-art method.

Chapter *6*

Conclusion and Future Work

6.1 Conclusion and Future work

We conclude that Neural Machine Translation is a favourable method to approach the challenging problem of translation among Indian languages. We demonstrate two methods to leverage NMT techniques for Indian language translation.

The first method exploits linguistic information specific to Indian languages to improve NMT performance. We observe significant improvement over a baseline NMT model. This suggests that although NMT is good at learning language constructs from the training data itself; addition of linguistic information to aid learning is not redundant, especially in low-sized parallel corpora conditions.

The second method employs a three-phase approach for NMT training which increases the domain coverage of the model as well as produces better translations. We achieve comparative scores to the state-of-the-art for multiple language pairs. We propose that this is a effective method in the presence of an existing MT system and large monolingual corpora but inadequate parallel corpora. We show by filtering out manual samples from the model outputs and providing a linguistic analysis of these samples that the three-phase training approach works well in most of the cases and shows better performance than other experimental approaches.

As part of future work, we would like to work on further improving the coverage of NMT models and enabling Zero Shot Machine Translation of Indian languages like Manipuri, Dogri and Maitheli which do not have any parallel corpora. Also, this thesis uses the same set of features for all language pairs for Method 1. In the future we would like to experiment with tailor-designed features designed keeping the language pair in mind. We would also like to experiment with using different atomic units for NMT, for eg Orthographic syllables as units when dealing with translation among closely related languages OR subword-level units to ensure lesser number of OOV words.

The usage of monolingual corpora to learn morphological information in combination with the exploitation of similar characteristics like word order and syntactic structures among linguistically similar languages is a promising research direction. This method can be applied to say, North East Indian languages, which especially lack large parallel corpora, but have similar word order and other grammatical constructs. Usage of paragraphs as units can also be explored, specially since that would enable the usage of popular books and their translations as parallel corpora for training. We would like to explore the usage of source as well as target translations for coarse learning and fine-tuning, in addition to exploring methods for vocabulary compression.

It is a general observation that discourse performance in NMT is not very good. We would like to work on this by using longer (more than a single sentence) atomic units of information on the encoder and the decoder side of the neural network. Multilingual training using a single model can also be explored. The attention mechanism which we employed in this thesis is a standard one proposed by . We would like to tweak the attention mechanism to cater to the linguistic phenomena depicted by Indian languages. Our intuition is that this would help significantly in improving translation performance between languages belonging to different language families.

In addition to these, NMT also struggles to deal with the translation of idioms - a very challenging aspect in any Machine Translation system. To approach this problem, we propose building of a parallel set containing idiom mappings across Indian languages as a starting point. Lookup and morphological variation depending upon context can then be performed for a good quality translation. We are currently working on the dataset, which can be used for better idiom translation in the future.

Related Publications

* **Experiments on different Recurrent Neural Network Architectures for English-Hindi Machine Translation**
  Ruchit Agrawal and Dipti Misra Sharma
  In *Proceedings of Third International Conference on Artificial Intelligence and Soft Computing* (AIS), Dubai, August 2017.
* **Integrating knowledge encoded by linguistic phenomena of Indian Languages with Neural Machine Translation**
  Ruchit Agrawal, Mihir Shekhar and Dipti Misra Sharma
  Submitted to *Fourteenth International Conference on Natural Language Processing* (ICON), India, December 2017.
* **Three-phase Training to address data sparsity in Neural Machine Translation**
  Ruchit Agrawal, Mihir Shekhar and Dipti Misra Sharma
  Submitted to *8th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics* (LTC), Poland, November 2017.
* **Leveraging Language Similarity and Sequence-to-Sequence Learning for Machine Translation among North East Indian languages**
  Ruchit Agrawal and Dipti Misra Sharma
  Submitted to *The Fifth International Conference on Mining Intelligence and Knowledge Exploration* (MIKE), India, December 2017.
* **A vis-à-vis evaluation of MT paradigms for linguistically distant languages**
  Ruchit Agrawal, Jahfar Ali and Dipti Misra Sharma
  Submitted to *Fourteenth International Conference on Natural Language Processing* (ICON), India, December 2017.

Bibliography

[1] R. Ananthakrishnan, M. Kavitha, J. H. Jayprasad, R. S. C. Shekhar, and S. M. S. Bade. Matra: A practical approach to fully-automatic indicative english-hindi machine translation. In *Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL'06)*, 2006.

[2] G. Anthes. Automated translation of indian languages. *Communications of the ACM*, 53(1):24–26, 2010.

[3] M. Auli, M. Galley, C. Quirk, and G. Zweig. Joint language and translation modeling with recurrent neural networks. In *EMNLP*, volume 3, page 0, 2013.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[6] L. Bentivogli, A. Bisazza, M. Cettolo, and M. Federico. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*, 2016.

[7] A. Bharati, V. Chaitanya, and R. Sangal. Anusaraka or language accessor: A short introduction. *Automatic Translation, Thiruvananthpuram, Int. school of Dravidian Linguistics*, 1994.

[8] A. Bharati, V. Chaitanya, R. Sangal, and K. Ramakrishnamacharyulu. *Natural language processing: a Paninian perspective.* Prentice-Hall of India New Delhi, 1995.

[9] R. M. Blench and M. Post. Rethinking sino-tibetan phylogeny from the perspective of north east indian languages. paper accepted for a volume of selected papers from the 16th himalayan languages symposium 2-5 september 2010 school of oriental and african studies, london. ed. *Nathan Hill. Mouton de Gruyter.*

[10] O. Bojar, V. Diatka, P. Rychlỳ, P. Stranák, V. Suchomel, A. Tamchyna, and D. Zeman. Hindencorp-hindi-english and hindi-only corpus for machine translation. In *LREC*, pages 3550–3555, 2014.

[11] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[12] Y. Cheng, W. Xu, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*, 2016.

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] J. Chung, C. Gülçehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075, 2015.

[15] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa Italy, 2006.

[16] W. De Mulder, S. Bethard, and M.-F. Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98, 2015.

[17] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. M. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380, 2014.

[18] G. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc., 2002.

[19] P. Dungarwal, R. Chatterjee, A. Mishra, A. Kunchukuttan, R. Shah, and P. Bhattacharyya. The iit bombay hindi english translation system at wmt 2014. *ACL 2014*, page 90, 2014.

[20] S. K. Dwivedi and P. P. Sukhadeve. Machine translation system in indian perspectives. *Journal of computer science*, 6(10):1111, 2010.

[21] O. Firat, B. Sankaran, Y. Al-Onaizan, F. T. Y. Vural, and K. Cho. Zero-resource translation with multi-lingual neural machine translation. *arXiv preprint arXiv:1606.04164*, 2016.

[22] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

[23] A. Haghighi and D. Klein. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1152–1161. Association for Computational Linguistics, 2009.

[24] W. He, Z. He, H. Wu, and H. Wang. Improved neural machine translation with smt features. In *AAAI*, pages 151–157, 2016.

[25] H. Isozaki, T. Hirao, K. Duh, K. Sudoh, and H. Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for

Computational Linguistics, 2010.

[26] B. Jawaid, A. Kamran, and O. Bojar. A tagged corpus and a tagger for urdu. In N. C. C. Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[27] G. N. Jha. The tdil program and the indian langauge corpora intitiative (ilci). In *LREC*, 2010.

[28] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *EMNLP*, number 39, page 413, 2013.

[29] N. J. Kalita and B. Islam. Bengali to assamese statistical machine translation using moses (corpus based). *arXiv preprint arXiv:1504.01182*, 2015.

[30] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*, 2017.

[31] P. Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[32] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

[33] A. Kunchukuttan and P. Bhattacharyya. Orthographic syllable as basic unit for smt between related languages. *arXiv preprint arXiv:1610.00634*, 2016.

[34] A. Kunchukuttan, A. Mishra, R. Chatterjee, R. Shah, and P. Bhattacharyya. Sata-anuvadak: Tackling multiway translation of indian languages. *pan*, 841(54,570):4–135, 2014.

[35] A. Lavie and M. J. Denkowski. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2):105–115, 2009.

[36] Q. V. Le and M. Schuster. A neural network for machine translation, at production scale. *Google Brain Team, Google Research Blog*, 2016.

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[38] H.-G. Lee, J. Lee, J.-S. Kim, and C.-K. Lee. Naver machine translation system for wat 2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 69–73, 2015.

[39] M.-T. Luong and C. D. Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, 2015.

[40] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[41] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

[42] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics, 2006.

[43] A. McEnery, P. Baker, R. Gaizauskas, and H. Cunningham. Emille: Building a corpus of south asian languages. *VIVEK-BOMBAY-*, 13(3):22–28, 2000.

[44] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[45] R. Rapp and C. M. Vide. Example-based machine translation using a dictionary of word pairs. In *Proceedings, LREC*, pages 1268–1273, 2006.

[46] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[47] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[48] H. Schwenk. Investigations on large-scale lightly-supervised training for statistical machine translation. In *IWSLT*, pages 182–189, 2008.

[49] R. Sennrich and B. Haddow. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*, 2016.

[50] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[51] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2016.

[52] M. T. Singh, R. Borgohain, and S. Gohain. An english-assamese machine translation system. *International Journal of Computer Applications*, 93(4), 2014.

[53] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200, 2006.

[54] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

[55] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[56] N. Ueffing, G. Haffari, A. Sarkar, et al. Transductive learning for statistical machine translation. In *Annual Meeting-Association for Computational Linguistics*, volume 45,

page 25, 2007.

[57] P. J. Werbos. Backpropagation through time, what it does and how to do it. *Proceedings of the IEEE*, 78, 1990.

[58] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560, 1990.

[59] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[60] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[61] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[62] B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*, 2016.