Literature Survey: Study of Neural Machine Translation

Jigar Mistry and Ajay Anand Verma
Pushpak Bhattacharyya
CFILT,Indian Institute of Technology Bombay, India
{jigarab,ajayanand}@cse.iitb.ac.in

Abstract

We build Neural Machine Translation (NMT) systems for English-Hindi,Bengali-Hindi and Gujarati-Hindi with two different units of translation i.e. word and subword and present a comparative study of subword NMT and word level NMT systems, along with strong results and case studies. We train attention-based encoder-decoder model for word level and use Byte Pair Encoding (BPE) in subword NMT for word segmentation. We conduct case studies to study the effects of BPE.

Since the NMT approach is a data driven approach, it suffers a lot by resource scarcity. This report also covers the Multitask learning which is an approach of transfer learning or inductive transfer. MultiTask Learning helps the learner to improve generalization performance by adding extra related tasks to the backpropagation net. The nub behind adding extra related tasks is domain specific information contained in the training signals of other tasks helps to learn shared feature of the main task better. We explained Multi-way multilingual model which is based on the MTL approach which learns the translation of several Indian language pairs in parallel. We also covers the performance gained by Multi-way multilingual neural machine translation in contrast with single pair neural machine translation.

1 Introduction

Neural machine translation is a newly emerging approach to machine translation, recently proposed by (Kalchbrenner and Blunsom, 2013), (Sutskever et al., 2014) and (Cho et al., 2014a). Unlike the traditional phrase-based translation system (see, e.g., (Koehn et al., 2003)) which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation (Bahdanau et al., 2014a).

From the probabilistic point of view translation is finding the best possible target sentence that maximizes the conditional probability of $\mathbf{y}$ given $\mathbf{x}$ i.e. $\mathbf{p}(\mathbf{y}|\mathbf{x})$ (Bahdanau et al., 2014a). Phrase based Statistical Machine Translation solves the problem of machine translation by training sub-components separately like *language model* and *translation model*. But Neural Machine Translation on the other hand tries to build end-end to single large neural network model to achieve the same goal.

Basic idea behind NMT is to encode a variable length sequence of words into a fixed length vector that can summarize the whole sentence. Then decode this encoded vector in target language to achieve the translation of source sentence. Whole encoder-decoder model is trained jointly to maximize the conditional probability $(\mathbf{y}|\mathbf{x})$.

2 Neural Machine Translation

A lot of work in NMT has been done with this model by active NMT research groups. In subsection 2.1 and subsection 2.2 we will describe *Basic Encoder-Decoder* model of NMT systems and then core of the current state-of-the-art NMT systems *Attention Based Encoder-Decoder* model.

2.1 Basic Encoder-Decoder model

This model can be divided into two parts *Encoder* and *Decoder*, both are implemented using RNNs. Encoder encodes the variable length sentence into fixed length vector called *summary vector* or *context vector*. Decoder takes this vector representation of sentence and generates target language translation.

Encoder: Encoder is an RNN whose state is updated each time it sees a word in sentence and the last state of it summarizes whole sentence which is called as summary vector h_T.

Step 1: Input word at any point of time while encoding sentence, is input to the encoder as *one-hot* vector w_i.

Step 2: Now *one-hot* vector of input vector w_i is transformed to low-dimension continuous space vector representation s_i. To do so, we can use previous learned word embeddings $\mathbf{E}$ or train them jointly. Word embedding matrix $\mathbf{E} \in R^{d \times V}$ contains as many coloumns as words in vocabulary. Each i^{th} column of word embedding matrix represent continuous vector space representation of i^{th} word of vocabulary. So when as a product of word embedding matrix and one-hot vector, continuous space vector of corresponding word is selected as shown in equation 1.

$$s_i = Ew_i \tag{1}$$

Step 3: In this step, RNNs hidden is updated to take into account new word s_i seen in the sentence.

$$h_i = f(s_i, h_{i-1}) \tag{2}$$

Where f is non-linear transformation function of RNN depending on variant of RNN (Vanilla, LSTM and GRU) used.

After processing last word of the sentence (T-th word if sentence length is T), state h_T that is obtained by encoder is called *summary vector* of sentence, which is a fixed dimensional vector representing whole sentence.

Decoder: Decoder is also an RNN, that takes input as summary vector, previous generated target word and last hidden state of it. After processing input, probability distribution over words in target language vocabulary is obtained. Target words are then sampled from this probability distribution. Process of decoding is described below:

Step 1: First internal state z_i of decoder RNN is calculated as

$$z_i = f'(h_T, z_{i-1}, u_{i-1}) \tag{3}$$

Where z_i and z_{i-1} are current and previous state of decoder, h_T is summary vector, u_{i-1} is previous generated target word. f' is non-linear transformation function of RNN.

Step 2: Based on the current state of the decoder, we compute the compatibility score for each word in vocabulary, later transform this score into probability.

$$e_k = w_k^T z_i \tag{4}$$

$$p(w_i = k | w_1, w_2, ..., w_{i-1}, h_T) = \frac{exp(e_k)}{\sum_j (exp(e_j))} \tag{5}$$

Here e_k, w_k are score and vector representation of k-th word in vocabulary. This score is high if it aligns with decoder well else low. $p(w_i = k | w_1, w_2, ..., w_{i-1}, h_T)$ is the probability of k-th word, for all $k \in 1, 2, ..., V$.

Step 3: From the probability distribution obtained from step 2, we sample target word. Decoder then again repeats steps 1 to step 3 until an *end-of-sentence* is not encountered. Hence a target sentence is generated corresponding to provided input source sentence.

$$e_k = w_k^T z_i \tag{6}$$

$$p(w_i = k | w_1, w_2, ..., w_{i-1}, h_T) = \frac{exp(e_k)}{\sum_j (exp(e_j))} \tag{7}$$

Here e_k, w_k are score and vector representation of k-th word in vocabulary. This score is high if it aligns with decoder well else low. $p(w_i = k | w_1, w_2, ..., w_{i-1}, h_T)$ is the probability of k-th word, for all $k \in 1, 2, ..., V$.

2.2 Encoder-Decoder with Attention Mechanism

In basic encoder-decoder model, encoder compresses the sentence into fixed length vector. This

summary vector contains the information of all the words in sentence. But as the length of sentence it fails to encode it efficiently in fixed summary vector, which degrades the performance of translation (Cho et al., 2014b). So in order to deal with this problem (Bahdanau et al., 2014a) proposed soft-search model, that uses attention mechanism in encoder-decoder model. Idea in this model is to not represent sentence with fixed length vector rather represent each word by a fixed length vector called *annotation* vectors and while generating each word in target language look for source sentences which are more relevant in source sentence.

Only difference with basic encoder-decoder is in encoder part, decoder part is same in both the model. In basic model decoder takes encoded summary vector as input, but in attention model it takes ***context vector*** as input to generate target sentence.

Context vector is the convex combination of annotation vectors of words in source sentence. It is calculated each time decoder generates a new word, while in basic model summary vector used to calculate only once. Convex coefficient used in computation of context vector are called ***attention weights***.

Annotation vectors: Annotation vector for each word is calculated by *bi-directional* RNN. Bidirectional RNN reads sentence from both directions i.e. left-to-right and right-to-left. State from left-to-right $\overrightarrow{h}_i$ and right-to-left $\overleftarrow{h}_i$ is concatenated for each word, this concatenated state is called annotation vector h_i.

Attention Weights: Attention weights gives *soft-alignment*, as these represent the probabilistic alignment of how words in source language and target language are aligned while generating target words. These are calculated each time decoder generate target word, and gives a probabilistic measure of how much each word in source language is important in the generation of current target word. To compute attention weights, first ***alignment score*** $e_{i,j}$ of each source word is computed that measures how relevant j-th word in source sentence is for i-th target word. This is computed for all source words in the generation of each target word by some alignment model a i.e.

$$e_{i,j} = a(z_{i-1}, h_j) \quad \forall j \in 1, 2, ..., T, \forall i \in 1, 2, ..., T' \tag{8}$$

where h_j is the annotation vector of j-th source word, T and T' are the lenghts of source and target sentences respectively and z_{i-1} is decoders last state. Then these alignment scores are transformed to probabilistic measure using softmax function. These probabilistic measures are called attention weights α_{ij}.

$$\alpha_{ij} = \frac{exp(e_j)}{\sum_{j'} exp(e_{j'})} \tag{9}$$

Since attention weights are probabilistic measures, we compute ***context vector*** as expected annotation vector by equation 10.

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{10}$$

Here c_i is the context vector obtained while generating i-th target word, T is the lenght of source sentence, α_{ij} is attention weight of j-th annotation vector and h_j is j-th annotation vector.
Once context vector is computed we compute decoders next state as a non-linear function (as of LSTM or GRU) of context vector c_i, previous target word u_{i-1} and decoders last state z_{i-1} as shown in equation 11

$$z_i = f(c_i, u_{i-1}, z_{i-1}) \tag{11}$$

After that we compute probability distribution over target vocabulary and sample target word in the same way as done basic encoder-decoder model repeatedly till complete sentence is not generated.

3 Subword NMT

NMT systems are trained on a limited size vocabulary, but test data can have different words than those in vocabulary such words are called unseen, rare or out-of-vocabulary words. Most of the out-of-vocabulary words are named entities, compound words and cognates (via morphological transformation) (Sennrich et al., 2015).

Named entities can be copied to target language translation if it shares the alphabets with source language, else transliteration is required. For Cognates and loan words character level

translation rules are sufficient. Translation of compound words can be achieved by translating its morphemes separately (Sennrich et al., 2015).

3.1 Byte-Pair Encoding

(Sennrich et al., 2015) proposed BPE based word segmentation method. In this method two vocabularies are maintained called training vocabulary and symbol vocabulary. Words in training vocabulary are represented as sequence of characters, plus an end-of-word symbol. All characters are added to symbol vocabulary. Then using BPE technique the most frequent symbol pair is identified, and all its occurrences are merged, producing a new symbol that is added to the vocabulary. This BPE step is repeated until a set of merge operations have been learned. Number of BPE merge operations in this method is also a hyper parameter.

Byte-Pair Encoding (BPE)[1] (Sennrich et al., 2015) is originally a data compression technique (Gage, 1994). Idea behind BPE is

"Find the most frequent pair of consecutive two character codes in the text, and then substitute an unused code for the occurrences of the pair." (Shibata et al., 1999)

Below example explains the BPE method:

Let the original text be
T_0 = PQPQRSUQSUVPQSUPQR.

Most frequent pair in T_0 is PQ, so we replace it by A. Modified Text is
T_1 = AARSUQSUVASUAR.

Now most frequent pair in T_1 is SU, so we replace it by B. Modified Text is
T_2 = AARBQBVABAR.

Now most frequent pair in T_2 is AR, so we replace it by C. Modified Text is
T_3 = ACBQBVABC.

In T_3 no pair is repeated so BPE algorithm stops here. Using BPE algorithm text of length $|T_0| = 18$ is compressed to a text of length $|T_3| = 9$. Additional information that is required to decode encoded text is list of encodings i.e. PQ $\rightarrow$ A, SU $\rightarrow$ B and AR $\rightarrow$ C. Since we perform BPE merge operations on character level first, this performs character level segmentation. As the number of merge operations are increased frequent sequence of characters and even full words are also encoded as a single symbol. This a allows a trade-off between the NMT model vocabulary size and the length of training sequence.

If there will be much larger merge operations then almost every word will belong to symbol vocabulary, that will prevent the sub-word level segmentation of words. When using BPE for sub-word segmentation, size of the sentences is increased as sub-words are separated by special symbols to allow decoding later. Larger the sentence size, it becomes difficult for NMT to learn well on them. So number of BPE merge operations is an important hyper parameter that is needed to be tuned properly before use.

4 Experiments and Results

4.1 Experimental setup

4.1.1 Data sets

We have used health and tourism domain ILCI corpus (Jha, 2010) for our experiments. This corpus contains 48k parallel sentences which we divided into three parts for training, testing and tuning/validation. Details of of these is given in table 1.

Corpus	Size
Training	44777 sentences
Tuning	1000 sentences
Testing	2000 sentences

Table 1: (Jha, 2010) data set used in experiments

4.1.2 Preprocessing and BPE setup

We've used Indic NLP Library [2] for preprocessing of Indian languages and BPE word segmentation method as descirbed in (Sennrich et al., 2015) for subword NMT.

[1] https://github.com/rsennrich/subword-nmt

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

4.1.3 NMT setup

For training word level NMT and subword NMT we have used nematus[3] (Sennrich et al., 2017) along with WMT scripts [4].

4.2 Results

We performed experiments for English-Hindi, Bengali-Hindi and Gujarati-Hidi language pairs with ILCI corpus (Jha, 2010), and obtained results shown in Table 2 , which supports use of BPE method for word segmentation.

Language-Pair	NMT_{word}	$NMT_{subword}$
En-Hi	26.22	26.88
Bn-Hi	30.71	33.87
Gu-Hi	48.02	53.95

Table 2: BLEU scores for word level and subword level NMT systems

From table 2 we can see there is a big improvement with BPE in Bengali-Hindi Language pair than English-Hindi. Bengali and Hindi both belongs to *Indo-Aryan* language family, but English belongs to *Indo-European* language family. Bengali and Hindi languages are closer to each other in terms of lexical similarity and morphological similarity than Hindi and English languages. Gujarati-Hindi language pair shows more improvement with BPE than Bengali-Hindi because Gujarati is more closer to Hindi than Bengali. When languages share such similarities on application of BPE they can provide mapping for words which were not in training data.

5 Data scarcity

The Neural machine translation is a data driven approach. In case of less parallel corpus Neural machine translation performs poorly. In case of statistical machine translation approach the issue of data scarcity is handled by Triangulation of two phrase tables. For ex, source-target parallel corpus is small, but there is a pivot language for which we have good amount of source-pivot and pivot-target parallel corpus. The nub of statistical machine translation is a phrase-table. The training process generates a phrase table which is essentially a mapping of word or sub-sequence of one language to a word or sub-sequence of another language with some score that tells how much reliable mapping is. To generate such phrase table for source-target we triangulate the phrase tables of source-pivot and pivot-target. In this way we transfer the learning gained by source-pivot training and pivot-target training to source-target language pair.

In NMT systems which is basically a set of encoder-decoder is a fully end-to-end system. The encoder encodes a source sentence into a point in continuous space or a set of vectors in continuous space, which is then decoded by the decoder into target sentence. We do not have latent structure like phrase-table, because NMT directly maximizes the probability of target sentences given source sentences without generating such latent structures. That is why the addressing data scarcity is not a trivial issue.

(Firat et al., 2016a) has proposed a Multi-way Multilingual NMT model that enables not only multiway translation but also allows trasnfer learning towards resource poor language pair when it is getting trained with resource rich language pairs.

(Cheng et al., 2016) has proposed a joint training of Neural Machine translation with source-pivot and pivot-target translation models to generate transltion for source-target translation.

(Saha et al., 2016) has proposed Interlingua based machine translation aims to encode multiple languages into common lingustic representation and then decode sentences in multiple target languages.

(Firat et al., 2016b) has proposed a novel finetuning algorithm for the recently introduced Multi-way Multilingual neural machine translation model that also enables zero-resource machine translation.

6 A Correlational Encoder Decoder Architecture for Pivot Based Sequence

The nub of this paper (Saha et al., 2016) is Interlingual theory. This idea is exploited in the context of neural encoder decoder architectures. The idea is fairly simple. If we have 3 languages say, X,Y and Z and the objective is to translate the sentences given in X to the sentences of

[3]https://github.com/rsennrich/nematus
[4]https://github.com/rsennrich/wmt16-scripts

Y. The problem is we do not have parallel corpus for training the model to translate from X to Y.

However, we have parallel corpus available for X-Z and Z-Y. In such situation language Z is referred to as a pivot/bridge language. The first and naive solution proposed by the authors is a two-stage model, which in first stage converts from X to Z and in second stage converts from Z to Y. Instead of that, authors suggested another approach. They try to explore interlingua inspired solution which is jointly learns to encode X and Z to a common representation and decode Y from this common representation.

Interlingua inspired machine translation aims at converting a sentence from any source languages into common linguistic representation and then decode this common representation into any target language. In the context of Neural machine translation, this implies that for n languages we require only n encoders and n decoders for translation between any of the possible $nC2$ languge pairs.

The problem of resource scarcity can be tackled if we could learn only n statistical encoders and n statistical decoders where the encoded representation is common across all language pairs, and decoder can decode from this common representation to their respective language.

6.1 Models proposed for Pivot

6.1.1 A two stage encoder-decoder model

A two stage model is just a simple approach of binding two NMT model together. Let us say we have 3 languages X, Y and Z for which we have two parallel corpus available XZ and ZY. A two stage model will learn a generative model for each of the pairs independantly. The encoders and decoders are Recurrent neural networks.

The first encoder uses parallel corpus XZ and learns to encode x_i and decode the corresponding z_i from this encoded representation. The second encoder is trained independently of the first encoderand uses ZY and learns to encode z_i and decode the corresponding y_i. These two independant training processes are indicated by the dashed line in 6.1.1. At the test time these two independant stages are run sequentially. This means this model takes in x_i convert this into z_i using stage1 model and then stage2 model will take this z_i and produces y_i. In this way end to end translatio path of XY is going through the translation path XZ and ZY.

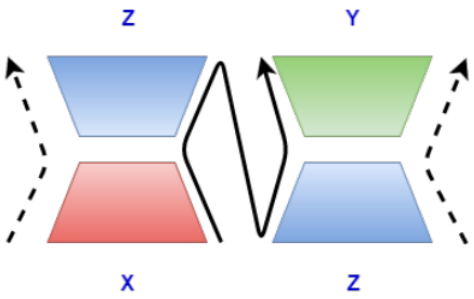


Figure 1: Two stage encoder-decoder model. Dashed lines denote how the model is used during training time and solid line denotes the test time usage. We can see that two encoder-decoders are trained independently but used jointly during testing.

(Saha et al., 2016)

6.1.2 A correlation based joint encoder-decoder model

(Saha et al., 2016) have suggeste a mode elegant solution which could scale even when more number of languages are involved. In proposed model which uses parallel corpus XZ to learn one encoder each for X and Z such that representations of x_i and z_i are correlated. In addition to this, the model uses parallel corpus ZY to learn to decode y_i from z_i. In this way encoder for Z benefits from instances in XZ and ZY.

The model tries to maximize the correlation between the encoded representations of x_i and z_i which is defined as follows:

$$J_{corr}(\theta) = -\lambda corr(s(h_X(X)), s(h_Z(Z)))$$

where h_X is the representation computed by the encoder for X and h_Z is the representation computed by the encoder for Z. $s()$ is a standardization funtion which adjust its hidden representations h_X and h_Y so that they have zero-mean and unit-variance. Further, λ is a scalling hyperparameter and corr is the correlation function as defined below:

$$\sum_{i=1}^{N} s(h_X(X))s(h_Z(Z))^T$$

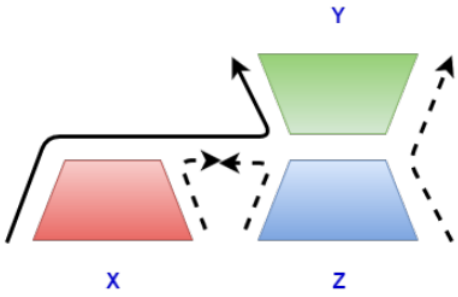
In addition to this, the model tries to minimize the following cross entropy loss :

$$j_{ce}(\theta) = \frac{1}{N_2} \sum_{k=1}^{N_2} P(y_k|z_k)$$

where

$$P(y_k|z_k) = \prod_{i=1}^{L} P(y_{k_i}|Y_{K_{<i}}, z_k)$$

The dashed line in ref represents the training processes where the model simaltaneously learns to compute correlated representations for x_i and z_i and decode y_i from z_i. The solid line represents the testing process where the model computes a hidden representation for x_i and then decodes y_i from it without going through z_i



(Saha et al., 2016)

Figure 1: Correlation based encoder-decoder

7 Multi-way Multilingual NMT

In the this section we will go through the Multi-way multilingual Neural Machine Translation model which helps to improve performance of resource poor language pairs. The nub of Multi-way multilingual NMT model is Multi-task learning. The Multi-task learning is a approach of transfer learning or inductive transfer. The method of achieving this is training multiple tasks in parallel while using a shared representation.

A system based on NMT, can be viewed as two modules. The first module called encoder which is responsible for mapping a source sentence into a continuous space representa- tion which is either a fixed dimensional vector in case of basic encoder-decoder network or a set of vectors in case of attention based encoder-decoder network. The second module called decoder is responsible for generating target sentence based on the continuous source representation. This makes it possible to build a system that maps source sentence in any language to common continuous representation space and decode the representation into target sentence in any language, allowing to make a multilingual machine translation system.

The suggested multi-way, multilingual neural machine translation makes it possible to learn single neural machine translation model to translate between multiple languages, with two incentives. One is that the number of parameters grows only linearly with number of languages. second is that for some language pair it outperforms single neural machine translation model and also it is beneficial to improve translation quality of low resource language pairs. This model is based on Multitask learning philosophy. It is made possible by sharing attention mechanism across all language pairs.

7.1 Existing Approaches

The possibility of building a system that maps a source sentence in any lan- guage into a common continuous space representation and decodes that com- mon representation into a sentence of any target language is straight forward to implement as discussed in (Luong et al., 2015). It has extended basic RNN model rather than the more effective attention based encoder-decoder network.

The critical issue in extending the attention based encoder-decoder network is that attention mechanism is conceptually language pair specific.

The other successful implementation was described in (Dong et al., 2015), in which the issue of language pair specific attention mechanism is cleverly avoided by considering one-to-many translation, where the target language decoder is equipped with its own attention mechanism.

7.1.1 Basic encoder-decoder for Multitask NMT

In (Luong et al., 2015), the aim is to extend the basic encoder-decoder network for multiasking neural machine translation. The model which they implemented is basically a set of encoders and decoders where each of the encoder converts the source sentence into a point in a common vector space. The point in the common vector space

then used by the decoder to generate different languages. The contrast between this model and Multi-way multilingual model is the later one extends the attention based encoder-decoder instead of the basic RNN model. Since the attention based neural machine translation has become de facto standard in NMT literatures that is why this model is an important contribution.

7.1.2 One to Many NMT

(Dong et al., 2015) has earlier proposed the multilingual translation model which is based on attention-based neural machine translation. But Their effort is to enable one-to-many translation. In this model a single pair attention based model is extended into a single encoder and multiple decoder each for a target language where each decoder is embedded with its own attention mechanism. The main contrast between this system and proposed system is the former is one-to-many while later one is multiway.

7.2 Challenges

At a first glance, it looks straight forward for any neural machine translation model to incorporate multiple languages at the source side and multiple languages at the target side. As discussed earlier, the idea looks very simple. we will employ a separate encoder for each source language and a separate decoder for each target languages. Encoder will transform the source sentences into common continuous space and decoder will generate target sentence from this representation.

In contrast to training multiple single pair neural machine translation models, in this system encoders and decoders are shared across multiple pairs. This means If I have to train 3 language pairs Hi-Gu, Gu-Hi and Bn-Gu then I need 3 separate encoders for Hi,Gu and Bn and 2 decoders for Gu and Hi. decoder for Gu will be shared across Hi-Gu and Bn-Gu. This is computationally beneficial since number of parameters grows linearly $O(L)$ with respect to number of languages L, whereas in case of separate single encoder-decoder model the number of parameters grows quadratically $O(L^2)$.

While generating the next target word, the Neural machine translation model use the attention layer to decide which parts of the source sentence need more attention and which part need less attention. Thus, in Neural machine translation described in (Bahdanau et al., 2014b) the attention mechanism was language pair specific layer.

There are issues with this way of including attention layer into the system. First is regarding the number of parameters. By employing language pair specific attention mechanism will cause **the number of parameters grows quadratically** again $O(L^2)$ with respect to number of languages L. Second and more important issue is the model may poorly generalize. It makes less likely for the model to get benefits of multiple related tasks (Caruana, 1998), specially **transfer learning** towards those language pairs for which sufficient amount of data is not available.

So, the main challenge tackled with this Multi-way multilingual model is to avoid separate attention mechanism for each language pairs. In the next section we will go through the architectural details of multi-way multilingual model to understand how to avoid separate attention mechanism. Through this work we also want to answer the following question: Is it possible to share the attention mechanism across the language pairs?

7.3 Architecture of Multi-way Multilingual NMT model

7.3.1 Problem Definition

Let us have N languages at source side where $N > 1$ and M languages at target side where $M > 1$ and we have $L <= M \times N$ parallel corpus $\{D_1, D_2, ..., D_L\}$ available. Let us use $s(D_l)$ and $t(D_l)$ to denote the source language and target language of l^{th} parallel corpus. The purpose is to train a model such that once the training is done, the model is able to translate from any of the source languages to any of the target languages included in the given language pairs. In the next section we describe the architectural details of multi-way multilingual model.

7.3.2 encoders

The multi-way multilingual model uses N encoders (which are basically bidirectional RNN same as used in (Bahdanau et al., 2014b)) $\{\Psi^n_{enc}\}^N_{n=1}$ and M decodes (same as (Bahdanau et al., 2014b)) $\{\Psi^m_{dec}\}^M_{m=1}$ and a attention mechanism f_{score} which is shared across all the encoders and decoders.

We used one encoder per each source language so that for 3 language pairs Gu-Hi, Hi-Gu and Gu-Bn we have 2 encoders one for Gu and Hi. In other words a single encoder is shared for two different target languages Hi and Bn. Because of this the dimension of annotation vectors produced by the encoders will be different across language pairs. To eliminate this variation a linear transformation layer is added to the original bidirectional RNN encoder. This linear transformation layer project each context vector into a common dimensional space.

$$h_t^n = W_{adp}^n[\overrightarrow{h_t}; \overleftarrow{h_t}] + b_{adp}^n$$

where $W_{adp}^n \in R^{d \times (dim\overrightarrow{h_t} + dim\overleftarrow{h_t})}$ is a weight matrix and $b_{adp}^n \in R^d$ is a bias vector.

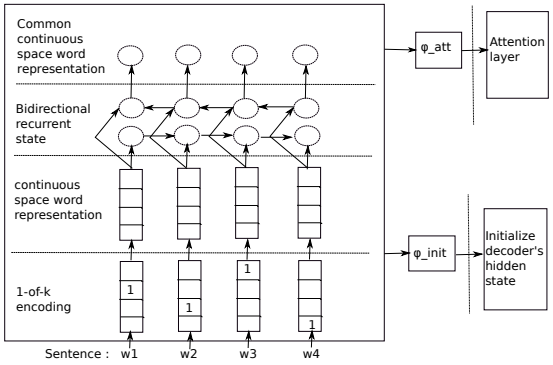


Figure 2: MLNMT: Encoder

In addition to this, each encoder is also equipped with two transformation function ϕ_{att}^n and ϕ_{init}^n. The former transformation function is responsible to convert the history vector to be compatible with a shared attention mechanism:

$$\tilde{h}_t^n = \phi_{att}^n(h_t^n)$$

where the transformation function ϕ_{att}^n can be implemented as any type of parametric function like element-wise tanh function.

The second transformation function ϕ_{init}^n makes the history vector to be compatible with the initializer of the decoder's hidden state.

$$\hat{h}_1^n = \phi_{init}^n(h_1^n)$$

where ϕ_{init}^n can be implemented as simple feed forward network with a single hidden layer and this network is also shared across all encoder-decoder pairs just like attention mechanism.

7.3.3 decoders

We use a separate decoder for each target language. For ex, If we have 3 language pairs Hi-Gu, Gu-Hi, Bn-Gu then we have 2 decoders one for Gu and Hi. This shows that decodes are shared for different source language, like decoder for Gu is shared for source language Hi and Bn.

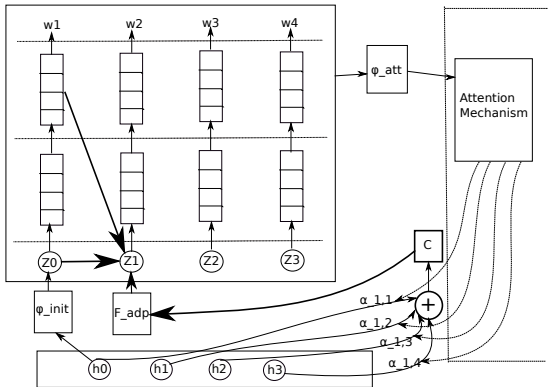


Figure 3: MLNMT: Decoder

Let us start with the initialization of the decoder's hidden state. Each decoder is equipped with φ_{init}^m which is responsible to convert the last hidden state $\hat{h}_{T_x}^n$ of the source encoder into the initial hidden state as follows:

$$z_0^m = \varphi_{init}^m(\hat{h}_{T_x}^n)$$

where φ_{init}^m can be implemented as a simple feed forward network with a single hidden layer equipped with tanh.

In addition to this, each decoder is also equipped with a parametric function φ_{att}^m which is responsible for transformation of its hidden state and the previously decoded symbol to be compatible with a shared attention mechanism.

$$\hat{z}_{t-1}^m = \varphi_{att}^m(z_{t-1}^m, E_y^m[\tilde{y}_{t-1}^m])$$

where φ_{att}^m can be implemented as a simple feed forward network with hidden layer equipped with tanh.

To compute next hidden state, the decoder uses the previous hidden state z_{t-1}^m, previous decoded target symbol $\tilde{y}_{t-1}^m$, and a context vector c_t^m as

follows:

$$z_t = \Psi^m_{dec}(z^m_{t-1}, E^m_y[\tilde{y}^m_{t-1}], f^m_{adp}(c^m_t))$$

where f^m_{adp} is a single affine transformation layer which is also shared along with shared attention mechanism. f^m_{adp} transforms the context vector computed by the shared attention mechanism to be compatible with the decoder.

The decoder outputs the probabilty distribution of next target symbol to be generated once the current hidden state is computed in the similar fashion as being done in (Bahdanau et al., 2014b).

7.3.4 Attention mechanism

The attention mechanism is shared in this model across all paris of encoders and decoders. This attention mechanism always take attention network compatible vectors that is why encoders and decoders are equipped with extra layers which make those vectors which are intended to be used by attention network, compatible with attention network.

The first step in attention is to compute the relevance score of each source token.

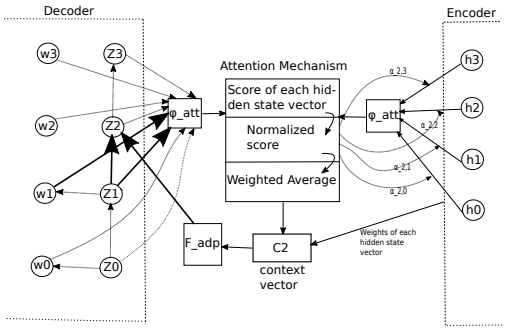


Figure 4: MLNMT: Attention

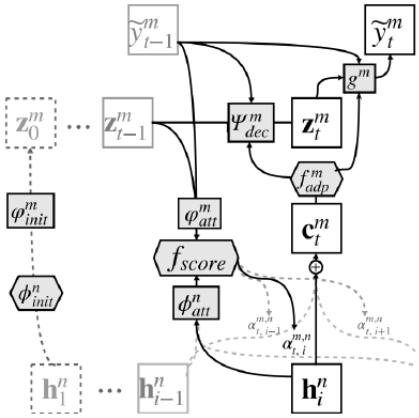
$$e^{m,n}_{t,i} = f_{score}(\tilde{h}^n_t, z^{\tilde{m}}_{t-1}, y^{\tilde{m}}_{t-1})$$

Normalization of these scores are done exactly in the same way described in (Bahdanau et al., 2014b) to carry out the attention weights $\alpha^{m,n}_{t,i}$. The time-dependant context vector is computed as the weighted average of history vectors as follows:

$$c^{m,n}_t = \sum_{i=1}^{T_x} \alpha^{m,n}_{t,i} h^n_i$$

.

To understand how all these entities work, look at the 5 to understand the working of the model at time step t for n-th encoder an m-th decoder.



(Firat et al., 2016a)

Figure 5: MLNMT: A sketch

7.3.5 Training objective

We have $L <= N \times M$ available corpus for respective language pairs. For each parallel corpus log-likelihood function is the same as defined in (Bahdanau et al., 2014b). For sake of simplicity we define a log-likelihood $L^{s(D_l),t(D_l)}$ for each language pair s and t.

The main goal of multi-way multilingual NMT model is to maximize the joint log-likelihood as follows:

$$L(\theta) = \frac{1}{L} \sum_{l=1}^{L} L^{s(D_l),t(D_l)}(\theta)$$

Once the training is done, the model is able to translate from any of the source languages to any of the target languages included in the parallel corpus.

8 Zero resource Translation with Multi-way Multilingual Neural Machien Translation

In this paper (Firat et al., 2016b), a novel finetuning algorithm for Multi-way multilingual NMT model has been proposed to enable zero-resource machine translation or Zero-shot machine translation. Zero-shot means the model is able to carry out translation between those language pairs for which it has not seen any parallel sentence pairs in training.

In the previous section we saw that translation quality can improve by exploiting positive language transfer (Dong et al., 2015; Firat et al., 2016a). In this paper (Firat et al., 2016b), the authors have investigate the potential of Multi-way Multilingual NMT model for Zero-resource machine translation in which there does not exist any parallel corpus during training. This is quite interesting while wondering aroung data scarcity issue because the less or no availability of parallel corpus for a language pair, the proposed model gives a way to use Multi-way multilingual NMT model to translate.

The authors investigate different translation strategies available in Multi-way Multilingual NMT model. They investigate mainly one-to-one translation and many-to-one translation. They tried to carry out zero resource based machine translation on first vanilla Multi-way multilingual model which shows that vannila model can not do zero-resource translation. Then they design a novel finetuning strategy that does not require any parallel corpus for a language pair for which we want to carry out zero resource translation. We will use the term zero resource language pair now onwards. The architectural details are same as in (Firat et al., 2016a) described in previous section.

8.1 One-to-One Translation

One-to-One translation strategy is straight forward to understand. In this strategy we have one encoder for source language and one decoder for target language and a shared attention mechanism. But, this is not the only strategy available in the model proposed by (Firat et al., 2016a). In fact, we have 3 different strategies available in the model like Many-to-One and Many-to-Many. However to carry our Multiway translation the model does not require the multi-way parallel corpus. In the next section we will go through two different methods for carrying out multi-source translation with multi-way multilingual NMT model.

8.2 Many-to-One Translation

In this Multi-way Multilingual NMT model, multi source translation can be thought of as averaging two separate transaltion paths. It turns out that there are two points available where averaging can happen.

8.2.1 Early Average

The first point of taking average of two translation paths is when computing the time dependant context vector. At each time t in the decoder, we compute a time dependant context vector for each source language C_t^1 and C_t^2 respectively. In this method, we just take the average of these two context vectors.

$$C_t = \frac{C_t^1 + C_t^2}{2}$$

Similarly, the decoder's hidden state is initilized by taking the average of the last hidden vectors of two source encoders.

$$Z_0 = \frac{1}{2}(\phi_{init}(h^1_{T_{t_x}}) + \phi_{init}(h^2_{T_{t_x}}))$$

8.2.2 Late Average

Alternatively, The averaging of two translation paths can be done at the output level. At each time t, each translation path computes the distribution over the target vocabulary. we can then average this probability distributions to get the multi-source output distributions.

The incentive of using late averaging over early averaging is both model need not to belong to single multilingual model. They can be two separately trained single pair translation models.

8.2.3 Combination of Early and Late Average

The two methods can be combined by late averaging the output layer's probability distributions taken from early averaged model.

8.3 Finetuning with Pseudo Parallel Corpus

If context vectors returned by the encoder are not compatible with the decoder, then zero resource translation fails. All we need is to adjust the context vectors to be compatible with the target

decoder. In this paper, authors suggested to adjust this zero resource translation path without any additional parallel corpus.

The authors suggested that a small set of pseudo parallel corpus is generated for zero resource language pair. We randomly select N sentence pairs from the target-pivot parallel corpus and translate the pivot language sentences into source language sentences. Then the pivot language sentences are removed and source language sentences are aligned with target language sentences to generate a pseudo parallel corpus for source-target.

We then make a copy of attention mechanism which is then referred to as target specific attention mechanism. We then finetune only this attention mechanism while keeping other parameters of encoder and decoder unchanged, using the generated pseudo parallel corpus. Once the model has been fientuned with the pseudo parallel corpus, we can use any of the translation strategies described in the previous section.

9 Conclusion

We have also gone through the neural machine translation approach. We realized the NMT model suffers from long sentence translation and rare or out-of-vocabulary words. The long sentence translation issue is addressed by attention based NMT model using LSTM activation units. Rare or Out-of-vocabulary words translation problem can be addressed using sub-word segmentation like Byte-pair Encoding (BPE).

We realized that in statistical approach, the pivot technique is applied on the phrase table. The phrase table is a data structure specific to given language pair. But, in case of Neural Machine Translation we do not have such latent structure, that is why to address data scarcity is not a trivial issue in neural machine translation. We revisited the triangulation pivot approach to realize that we transfer information contained in resource-rich language pairs to resource-poor language pair. Such transfer of information is done via phrase table triangulation. The main question that first strike is that in the absence of phrase table how do we transfer information from one NMT model to another NMT model.

we have seen two promising approaches which are Multitask learning based and Interlingua inspired approahes respectively. Interlingua inspired correlational Encoder Decoder model propose a neural network based model which explicitly maximizes the correlation between the source and pivot view and simultaneously learns to decode target sequences from this correlated representation. Futurework would be applying this idea of correlation to attention based encoder decoder models.

The Multitask learning approach suggested that if many related tasks are learned in parallel then the information contained in training signal of one task can help to learn another task better specially in case of lack of resources for another task. The proposed Multi-way multilingual model follows the same approach, which maps a source sentence into a common continuous representation space and and then decodes the representation into any of the target language. The result shows that Multi-way Multilingual model improves the result specially for a resource-poor language pairs.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014a. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014b. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928* .

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR* abs/1409.1259. http://arxiv.org/abs/1409.1259.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014b. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL (1)*. pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073* .

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multilingual neural machine translation. *arXiv preprint arXiv:1606.04164* .

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38. http://dl.acm.org/citation.cfm?id=177910.177914.

Girish Nath Jha. 2010. The tdil program and the indian language corpora initiative (ilci). In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). European Language Resources Association (ELRA)*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. Association for Computational Linguistics, Seattle.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '03, pages 48–54. https://doi.org/10.3115/1073445.1073462.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* .

Amrita Saha, Mitesh M Khapra, Sarath Chandar, Janarthanan Rajendran, and Kyunghyun Cho. 2016. A correlational encoder decoder architecture for pivot based sequence generation. *arXiv preprint arXiv:1606.04754* .

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR* abs/1508.07909. http://arxiv.org/abs/1508.07909.

Yusuke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.