Literature Survey: Neural Machine Translation in Low Resource Setting

Aditya Jain, Aakash Banerjee, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
{adityajainiitb, abanerjee, pb}@cse.iitb.ac.in

Abstract

Neural Machine Translation(NMT) is the dominant paradigm in the field of Machine Translation(MT) now. It has significantly improved the performance of MT models compared to previous approaches like Statistical or rule-based MT. One problem with Neural models is that they require lot of clean parallel data to perform well which makes it unsuitable for low resource language pairs. In this paper we first look at the basics and recent developments in the field of NMT and then move on to approaches which help mitigate the problem of low resourceness. We look at RNNs and its various types, transformer model, byte pair encoding, backtranslation, pivoting, multilingual models, phrase table injection and cmbined corpus.

1 Introduction

Machine Translation has existed for a long time and has become more relevant in recent times due to proliferation of the web. Machine Translation became very important during the Cold War in the 1960s when Alan Turing was working on Enigma to decipher secret code of war messages. In recent times with the internet becoming a big thing automatic translation tools is in big demand and software giants like Google, Microsoft are constantly working to build better models. Rule-based MT is the first paradigm where the translation rules were manually developed. In 1980s Example-based MT was introduced which relied more on data. Finally Statistical MT was introduced and it became the ruling paradigm for a long time. Now Neural MT is the big thing with neural architectures being used. With automatic means of translation in demand and with a lot of work to be still done Machine translation is a very exciting field.

2 Neural Machine Translation (NMT)

In NMT, we make use of a neural network model to learn a statistical model for machine translation. One of the key advantages of NMT over SMT is that in NMT, a single system can be trained directly on source as well as target text thereby removing the dependency on a pipeline of specialized systems as that in SMT. Various MT models that makes use the RNN or the LSTM architecture falls into this category. Also the Attention mechanism, which is used in most of the encoder-decoder based models, is a part of this paradigm of MT.

2.1 LSTMs

The concept of LSTMs was first introduced in the work by (Hochreiter and Schmidhuber, 1997). Traditional Neural Networks fail to deal with sequence-to-sequence model problems like MT, ASR, etc. In order to overcome this issue, Recurrent Neural Networks (RNNs) were introduced. In simple terms, RNNs is exactly same or an extension of the traditional Neural Network, with an extra loop in them.

2.1.1 Issues with RNN

The RNNs are capable of restoring the past information, as a result, these models can be assumed to have memory, unlike traditional Neural Networks.This capability of memory restoration has proved to be a huge success of RNNs is as this enables them to connect or align previous information while processing the current task in hand.

Example: Say, you are trying to predict the next word in the sentence: "*The color of the sky is mostly Blue*", which means, the word **Blue** looks very

[0] $ImagesSource : jalammar.github.io/illustrated-transformer/$

probable. But, when teh sequence-to-sequence tasks involve a long term dependency of any of the current task on previous set of inputs, such tasks cannot be handled with the traditional Neural Network as well as the RNNs. Such sequence-to-sequence tasks involving long-temr dependencies can be handled by making use of the memorization concept of LSTMs. For example in "*My birthplace is Gujarat..... I am fluent in speaking* ***Gujarati***". In this case, the dots represents one or more missing sentences or phrases.

While dealing with such long-term dependencies of a current and the previous phrases and their context, experimentally RNNs are found to fail often and this paves the need to have a mechanism that handles this shortcoming of RNNS, and hence, LSTM or Long Short Term Memory comes into play.

2.2 NMT using BiRNN and Attention Mechanism

There has been a gradual progress in the field of translation, progressing from community gatherings and scheduling meetings, to hotel accommodation and flight reservations. However, still there is great need to further expand the supported fields to include a wide range of day-to-day conversations and official business meetings. This paves the need to have Sequence-to-Sequence Translation Models to cater to such situations.

In a high level view, a Sequence-to-Sequence model usually consists of three main components

1. **Encoder:** the encoder encodes the source sentence into a fixed-length vector.
2. **Decoder:** the decoder generates the translation word by word.
3. **Attention module:** as the per the decoder's need, it selectively retrieves the source side information.

We will look at each of these components in detail in the upcoming subsections.

2.2.1 Introduction to Attention

The paper (Bahdanau et al., 2016) introduced the concept of the Attention Mechanism. This paper states that there is a need to have a Neural Machine Translation model that can be used to improve the translation performance, which cannot be achieved using the Statistical MT approach. Moreover, such Neural MT based models usually belong to the category of encoder-decoder models that performs translation using a fixed-length vector space, which proves to be a bottleneck in improving the translation performance of such models. As a result, there is a great need to have a mechanism which is not restricted to a fixed-length vectors and can automatically determine the parts of the source sentence that are more relevant in predicting the target word.

2.2.2 Description of the Model

The model consists of two major components namely Encoder and the Decoder, the detailed functionalities provided by them are as follows:
Encoder in detail: The proposed model expects the annotation of each word to summarize the preceding as well as the succeeding words. To achieve this, it makes use of a Bi-directional Recurrent Neural Network (BiRNN).

As the name suggests, a BiRNN consists of two RNNs, the forward and the backward RNN. The forward RNN is used to traverse and read the input sequence in the forward direction whereas the backward RNN is used to traverse and read the input sequence in the backward or reverse direction, starting from the last word and moving towards the first, thereby producing two different sets of hidden states represented as $(\overrightarrow{h_1}, ..., \overrightarrow{h_{T_x}})$ and $(\overleftarrow{h_1}, ..., \overleftarrow{h_{T_x}})$ respectively.

Finally, these two sets of hidden states are combined or concatenated to form the required annotation h_j, that focuses on the words around x_j.

Decoder in detail: The proposed model calculates the the conditional probability as follows:

$$p(y_i|y_1, ..., y_{i-1}, x) = g(y_{i-1}, s_i, c_i) \quad (1)$$

where, s_i : RNN hidden state for time i which is computed as follows:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (2)$$

It can be noted that as compared to previous models, in this model, the probability is conditioned on distinct context vector c_i each time and not on fixed length context vector.

The context vector c_i is calculated as weighted sum of annotations h_i :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (3)$$

The weight α_{ij} is evaluated as follows:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \quad (4)$$

where,

$$e_{ij} = a(s_{i-1}, h_j) \quad (5)$$

is an alignment model which scores how well the inputs around position j and the output at position i match.

The probability α_{ij}, or its associated energy e_{ij}, reflects the importance of the annotation h_j with respect to the previous hidden states s_{i1} in deciding the next states s_i and generating y_i.This Attention mechanism then decides the parts of the source sentence to be focused on.

The architecture of the model described in the paper cite paper here can be seen below:

The upcoming subsection introduces the concept of the **Transformer Model** that makes use the above proposed Attention Mechanism along with the Encoder-Decoder pairs to efficiently train the end-to-end speech-to-text models.

2.3 The Transformer Model

The transformer consists of two major components, the first being the stack of encoders and the second is the stack of decoders, both of which are of same numbers.

2.3.1 Encoder

The sequence of tokens, provided as input to the Transformer model is first converted into an **embedding space** of vectors of dimension d_{model}. Positional Encoding As the position of the words in the input sentence effects the context in which it is used, there is a need to provide some information about the relative or absolute position of the tokens in the sequence. This is achieved by the Positional Encoding mechanism, which is applied to the input embedding before it enters the Encoder and the Decoder stacks. This positional encoding has the same dimension as that of the embedding space.

In the paper (Rabiner, 1989), Positional Encoding concept is implemented by making use of sine and cosine functions of different frequencies as shown below:

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}}) \quad (6)$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_{model}}) \quad (7)$$

In the proposed model, the Encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward neural network. The abstract view of the Encoder is given below:

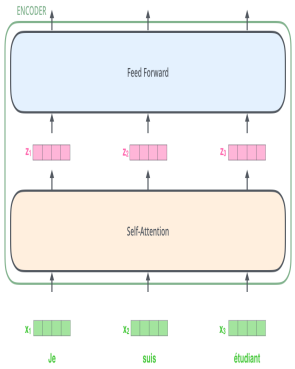

Figure 1: Architecture of an Encoder in Transformer Model.

Here, the self-attention layer maps an input embedding $x = (x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$, which are then passed through the feed-forward layer to form the final representation which is then provided as input to the decoder.

2.3.2 Decoder

The decoder is also composed of a stack of 6 identical layers. In addition to the two layers present within the encoder, the decoder consists of an extra layer, which is a multi-head attention layer. The various types or variants of the attention mechanism used within the Transformer model are discussed below.

2.3.3 Attention

The various variants of the Attention mechanism used within the Transformer model are listed below:

1. **Encoder-Decoder Attention:** This variant of Attention is only used within the Decoder and not the Encoder. It is placed in between the Self-Attention and the Neural Network Layer within the Decoder. The key functionality provided by this variant is to help the decoder focus on appropriate places in the sequence provided by the Self-Attention layer below it. The Encoder-Decoder Attention is already discussed in the above subsections.

[0] http://jalammar.github.io/illustrated-transformer/

2. **Self-Attention:** Self-attention, also referred to as **intra-attention**, is a variant of the attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence.

   For example, consider the input sequence to be

   The big red dog.

   The below image shows the weight or the importance given to each of the word of the input sentence while computing the self-attention of each of the words with reference to the input sentence itself.

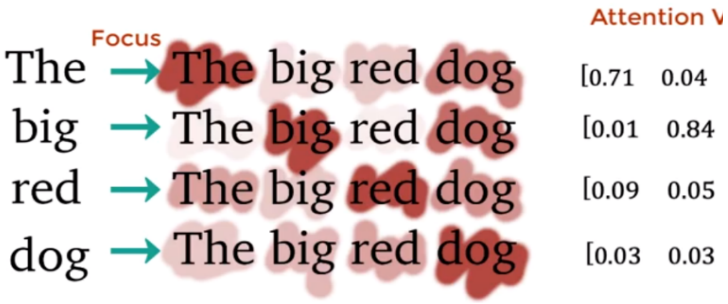

Figure 2: Example of Self-Attention.

   In the above example, the Self-attention mechanism enables the model to learn the co-relation between the current words and the remaining part of the sentence. On the basis this mechanism, a Attention matrix is generated which denotes the importance given to each word with reference to the word under consideration.

3. **Scaled Dot-Product Attention:** The two forms of most commonly used Attention functions are the **Dot-Product or Multiplicative** attention function and the **Additive** attention function. Among these, **Dot-Product attention is more time and space efficient as compared to the Additive attention**, as it can be implemented by making use of highly optimized matrix multiplication codes whereas, Additive attention is implemented by making use of a Feed-Forward Neural Network architecture.

   In this variant of Attention, the input consists of Keys and Queries vectors of dimension d_k, and Values vector of dimension d_v. The dot product is then computed between the Query (Q) and all the Key (K) vectors, the result of which is scaled down by dividing it by a factor of $\sqrt{d_k}$. Finally, a Softmax function is applied to obtain the weights on the Values (V). The pictorial representation of these series of steps is shown below

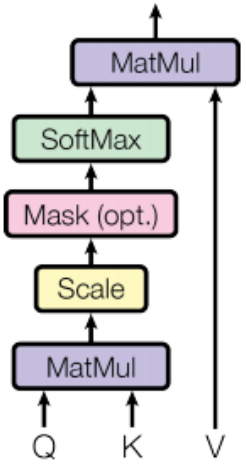

Figure 3: Scaled Dot-Product Attention.

   The output matrix provided by the Self-attention layer is computed as follows

$$\textbf{Attention (Q, K, V)} = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (8)$$

   The attention function is computed on a set of queries simultaneously, packed together into a Query matrix Q. The keys and values are also packed together into matrices K and V respectively.

4. **Multi-Head Attention:** In simple terms, a Multi-head attention is a combination of attention heads, each having different weight vectors. The pictorial representation of the multi-head attention mechanism can be seen below: This variant of the attention mechanism, expands the model's ability to focus on different positions. The basic idea behind this is to compute the attention heads again and again by changing the value of the weight vector, each time, to come up with a set of attention heads. These multiple attention heads are then concatenated to compute the multi-head attention.

[0] https://www.youtube.com/watch?v=TQQlZhbC5ps

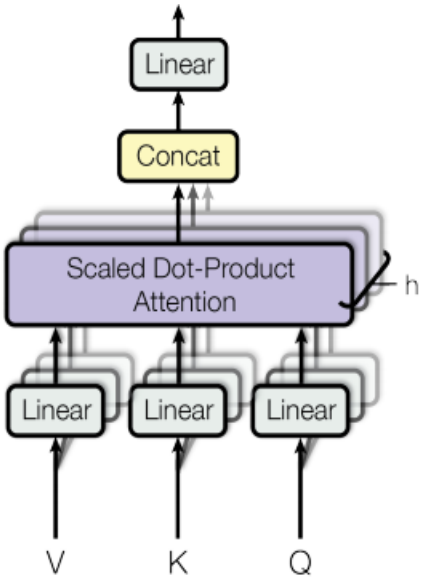

Figure 4: Multi-Head Attention.

3 Byte Pair Encoding

Byte Pair Encoding(BPE) is a technique which was introduced in the paper (Sennrich et al., 2016b) for word segmentation in Neural Machine Translation(NMT). Byte Pair Encoding was previously used as a data compression technique which combined the most frequent pair of bytes. In NMT it is used for merging characters or character sequences. We know that Machine Translation is an open vocabulary task whereas NMT models operate on a fixed size vocabulary. To address this issue, BPE is used which encodes rare and out of vocabulary(OOV) words as sequences of subword units. Previously the translation of out-of-vocabulary words was achieved by backing off to a dictionary. The two main contributions of BPE can be summarized as:

- Open-vocabulary neural machine translation is possible by encoding (rare) words via subword units. This technique is much more effective than large vocabularies or backoff dictionaries
- As BPE is a compression technique, it allows for a compact representation of an open vocabulary through a a fixed-size vocabulary of variable-length character sequences.

3.1 Subword Translation

The main motivation behind the paper is that translation of some words is transparent. This means that even a translator who has never seen that word can translate it based on translation of known subword units like morphemes or phonemes. Word Categories whose translation is transparent include:

- **Named Entities**: : For translation between languages that share common alphabets, named entities can directly be copied or else Transliteration comes into picture when languages do not share alphabets.
- **Cognates and Loanwords**: Cognates and loanwords with a common origin can differ in regular ways between languages, so that character-level translation rules are sufficient
- **Morphologically Complex Words**: Words containing multiple morphemes, for instance formed via compounding, agglutination, or inflection, may be translatable by translating the morphemes separately.

3.2 How BPE works?

As mentioned earlier BPE is a compression technique which iteratively merges the most frequent character/character sequence. The number of merge operation to use is a hyperparameter which is different for different languages and corpora size. The main difference with other compression techniques(like Huffman Encoding) is that the symbol sequences are still interpretable as subword units. The main steps of BPE algorithm can be summarized as below:

- Initialize the symbol vocabulary with the character vocabulary
- Each word is represented as a sequence of characters with an additional symbol (/w) at the end representing end of word
- In each iteration, the two most frequent pair say (‘A’, ‘B’) is replaced with a new symbol ‘AB’ and vocabulary size increases by one after each merge operation
- The number of merge operations is the only hyperparameter

4 Dealing with Low Resource languages

4.1 Phrase Table Injection

(Sen et al., 2018) and (Dewangan et al., 2021) used this technique, shown in Figure 5, to combine both SMT and NMT. We know that the phrase table, generated during training of a SMT model, plays

a key role in the SMT translation process. It contains a probabilistic mapping of phrases from the source language to the target language. The phrases present in the phrase table are combined with the available parallel corpora; thereby increasing the data available to train the NMT model. This also helps the model to learn translation of short correct phrases along with long sentences.

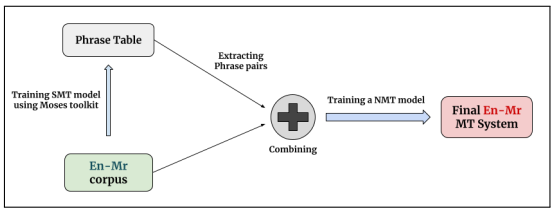

Figure 5: Phrase Table Injection

4.2 Backtranslation

Neural Machine Translation has achieved state-of-the-art results for many languages pairs while only using parallel data for training. In Statistical Machine Translation target side monolingual data plays a pivotal role. The two main reasons for target side monolingual data importance in SMT are:

- It is used to train a Language Model which is used in a log-linear combination with other components which helps a lot in increasing Fluency.
- The amount of available monolingual data in the target language typically far exceeds the amount of parallel data,

Target side monolingual data has been previously used in NMT by incorporating external Language Models but they have not been very effective and they also needed a change in model architecture. The concept of Backtranslation introduced in the paper (Sennrich et al., 2016a) which uses target side monolingual data in the same model and gives much better results compared to previous approaches. The main contributions of this paper are:

- The machine translation quality of NMT systems can be improved by mixing monolingual target sentences into the training set.
- Two different methods are investigated to fill the source side of monolingual training: using a dummy source sentence, and using a source sentence obtained via backtranslation.

4.2.1 NMT Training with Monolingual Data

In Machine Translation, more monolingual data serves to improve the estimate of the prior probability of the target sentences(by using Language Models in SMT). This paper does not use separate models on the monolingual data but exploits the fact that encoder-decoder neural networks already condition the probability distribution of the next target word on the previous target words.

To ensure that the output layer remains sensitive to the source context, and that good parameters are not unlearned from monolingual data, the monolingual training instances are paired with synthetic source sentences from which a context vector can be approximated. The synthetic source side sentences are obtained via back-translation, i.e. an automatic translation of the monolingual target text into the source language. This is generally achieved by training a target to source side model using the same parallel corpus and using it for backtranslating.

During training the synthetic parallel text is mixed into the original text with no distinction among them. Only the source side of these additional training examples is synthetic, and the target side comes from the monolingual corpus. The synthetic and original data is used in a 1:1 ratio (randomly shuffled). We cannot arbitrarily increase the ratio of monolingual training instances because different output layer parameters are optimal for the two tasks, and the network 'unlearns' its conditioning on the source context if the ratio of monolingual training instances is too high.

4.3 Combined Corpus

In this technique the knowledge from similar languages on the target side is exploited. As shown in Figure 6, at first a NMT model is trained using combined corpora from English-Marathi and English-Hindi (EnglishEnglish-HindiMarathi) language pairs. This model is then fine-tuned with the English-Marathi parallel corpora only, using the same vocabulary as that used while training. The intuition is that a model which at the start of training knows how to translate mixed languages is better than a model initialized with random weights.

This technique will be more effective if the languages at the target side are similar as this will potentially lead to a partial overlap in the target side vocabulary. Here Hindi and Marathi are the target languages which are similar as both belong

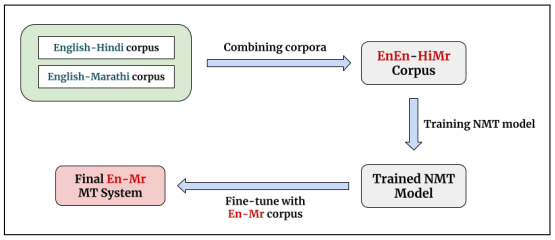

Figure 6: Combined Corpus

to the same language family (Indo-Aryan) and have an overlap in their alphabet set

4.4 Pivoting

Both Statistical MT and Neural MT rely on large quantities of parallel corpora to produce translations of relatively higher quality. Unfortunately, large quantities of parallel data are not readily available for some languages pairs, therefore limiting the potential use of current MT systems. To solve this problem some researchers use a third language called pivot language for which there exist large source-pivot and pivot-target bilingual corpora.

4.4.1 Pivot Methods for Phrase-based SMT

The paper (Wu and Wang, 2009) talks about three different approaches to Pivot based MT in context of SMT systems.

The three approaches are briefly described below:

- **Triangulation Method:** Here, the source-pivot and pivot-target translation models are first trained using the source-pivot and pivot-target corpora. Based on these models a source-target model is built in which two important elements need to be induced: phrase translation probability and lexical weight. These parameters are calculated by making independence assumptions and estimating the the co-occurring frequencies of word pairs directly from the induced phrase pairs.
- **Transfer Method:** This is a very simple technique in which the system first translates from the source language to the pivot language using a source-pivot model, and then from the pivot language to the target language using a pivot-target model. The problem here is that the errors get propagated through two models and add up to reduce performance of overall system.
- **Synthetic Method:** A source-target corpus can be built in two ways from the source-pivot and pivot-target corpora. One way is to obtain translations of source sentences in the source-pivot corpus. This can be achieved by translating the pivot sentences of source-pivot corpus using the pivot to target model. Similarly the pivot sentences of the pivot-target corpus can be translated by using a pivot to source model. We can combine these two source-target corpora to produced a final synthetic corpus.

4.4.2 Pivot Methods for NMT

The paper (Saha et al., 2016) used a correlation based joint encoder-decoder model instead of a two stage model. We see a two stage model in Figure 7 where X is the source language Y is the pivot language and Z is the target language. This model takes double time during decoding and errors also get accumulated over 2 independent models.

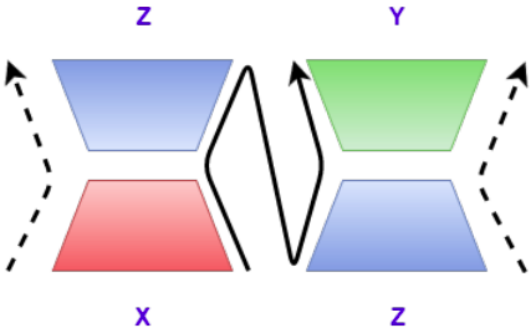

Figure 7: Two stage encoder-decoder model. Dashed lines denote how the model is used during training time and solid line denotes the test time usage. The two encoder and decoders are trained independently but used jointly during testing. ((Saha et al., 2016))

To overcome the issues with two stage model, a new model is proposed. Figure 8 shows the setup. Here the training corpora between X and Z(source and pivot) is used to train the two encoders marked X and Z in the diagram. They are trained such that the correlation between the source and pivot sentences is maximized. Before maximizing correlation the the hidden representation of X and Z is normalized(zero mean and unit variance) which helps in calculating correlation. Simultaneously the parallel corpora between Z and Y is used to train the Z-Y encoder decoder model to minimize the cross entropy loss. While training mini batches are picked from the X-Z and Z-Y corpora and correlation is maximized and cross-entropy loss is minimized simultaneously.

Another approach to use of pivot language in NMT is through Transfer Learning((Kim et al., 2019)).

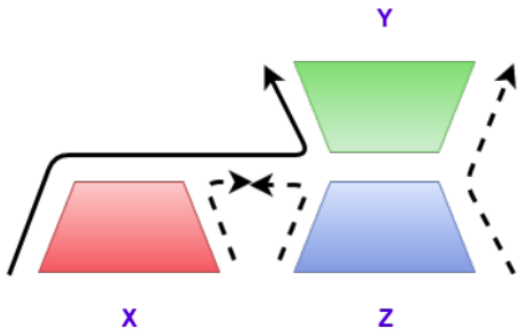

Figure 8: Correlated encoder-decoder model. Dashed lines denote how the model is used during training time and solid line denotes the test time usage. During training, both the encoders are trained to produce correlated representations and the decoder for Y is trained based on encoder Z. During test time only encoder for X and decoder for Y are used. ((Saha et al., 2016))

This can be used when no source-target parallel corpus is available as well as when insufficient source-target parallel corpus is available. The core steps of the transfer can be summarized as:

- Pre-train a source-¿pivot model with a source-pivot parallel corpus and a pivot-¿target model with a pivot-target parallel corpus.
- Initialize the source-¿target model with the source encoder from the pre-trained source-¿pivot model and the target decoder from the pre-trained pivot→target model.
- Continue the training with a source-¿target parallel corpus.

If the last step is skipped, when no source-target corpus is available then it corresponds to zero-shot translation. Figure 9 shows us this setup visually.

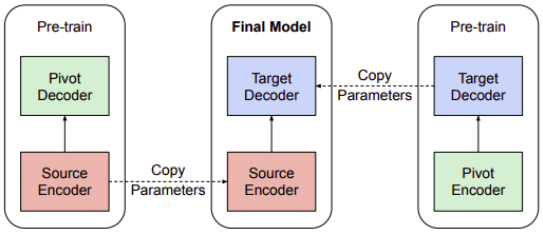

Figure 9: Pivot based Transfer Learning ((Kim et al., 2019))

4.5 Multi-lingual models

(Dabre et al., 2020) provides a very detailed survey of multilingual NMT and touches upon all aspects of multilingual NMT. Figure 10 shows us an overview of MNMT for low-resource language pairs.

4.5.1 Training

This section falls under transfer learning paradigm. The simplest approach is jointly training for both language pairs but the final model may not be optimally tuned for the child language pair. In case of multiple languages on the target side of the model a language flag is appended at the start of every source language sentence. Section 4.1.2 has talked about another fine-tuning approach. Meta-learning can be another approach where the parent model parameters are amenable to fast adaptation.

4.5.2 Lexical Transfer

These approaches tries to bridge the lexical divergence gap between parent and child languages. One way can be to map the pre-trained word embeddings of the parent and child languages to a common space, that is, using cross-lingual embeddings. This mapping is learned via the orthogonal Procrustes method using a bilingual dictionaries between the sources and the target language. A variation of this approach can be where the parent model is first trained and monolingual word-embeddings of the child source are mapped to the parent source's embeddings prior to fine-tuning.

4.5.3 Syntactic Transfer

Syntactic transfer tries to deal with the difference in syntax between the parent and child languages as they may not belong to the same language family always. Thus fine-tuning the child model from parent blindly may not always work. Not much work has been done on this area. One approach has been to reduce the word order divergence between source languages by pre-ordering the parent sentences to watch child word order is beneficial in extremely low-resource scenarios. Another approach has been to train the parent encoder with noisy data (insertion/deletion of words, changing word order etc) which ensures that the encoder is not over-optimized for the parent source language syntax.

4.5.4 Language Relatedness

A related parent language benefits the child language more than an unrelated parent. How to utilize language relatedness for improving accuracy is an important question. Language relatedness is typically exploited by using shared Byte Pair Encoding(BPE) vocabulary and BPE embeddings

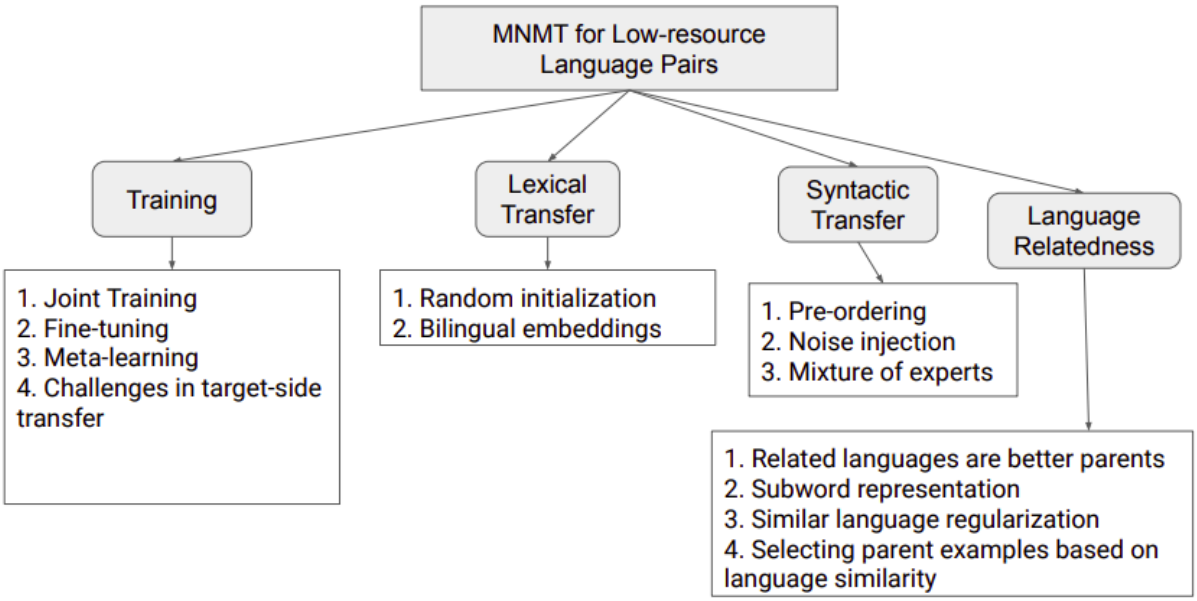

Figure 10: From left to right: single pair, semi-supervised, multilingual, and transfer learning strategies ((Dabre et al., 2020))

between the parent and child languages. We have seen BPE in detail in section 3.1. Other approaches could be by using a unified transliteration scheme at the character level, using "similar language regularization" to prevent overfitting and so on which gives significant gains.

5 Conclusion

In this literature survey paper we looked at Neural Machine Translation and how it can produce good results in low resource scenarios. We looked at word embeddings, LSTM, GRU's and the Transformer model which is state of the art now. We also looked at attention and its various types which gives a significant boost to any NMT model performance. We also looked at subword tokenization techniques with a focus on Byte Pair Encoding. Finally we looked at some of the methods which help us deal with low resource language pairs like back-translation, pivoting, multilingual models, phrase table injection and combined corpus. We also got a peek into statistical machine translation in PTI and pivoting techniques.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A comprehensive survey of multilingual neural machine translation. *CoRR*, abs/2001.01115.

Shubham Dewangan, Shreya Alva, Nitish Joshi, and Pushpak Bhattacharyya. 2021. Experience of neural machine translation between indian languages. *Machine Translation*, pages 1–29.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, and Hermann Ney. 2019. Pivot-based transfer learning for neural machine translation between non-english languages. *CoRR*, abs/1909.09524.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Amrita Saha, Mitesh M. Khapra, Sarath Chandar, Janarthanan Rajendran, and Kyunghyun Cho. 2016. A correlational encoder decoder architecture for pivot based sequence generation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 109–118, Osaka, Japan. The COLING 2016 Organizing Committee.

Sukanta Sen, Mohammed Hasanuzzaman, Asif Ekbal, Pushpak Bhattacharyya, and Andy Way. 2018. Neural machine translation of low-resource languages using smt phrase pair injection. *Natural Language Engineering*, pages 1–22.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Hua Wu and Haifeng Wang. 2009. Revisiting pivot language approach for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 154–162, Suntec, Singapore. Association for Computational Linguistics.