# Multi-source Pivot-based Neural Machine Translation: a survey

**Pranav Gaikwad and Pushpak Bhattacharyya**

CFILT, Indian Institute of Technology Bombay, India

{pranavgaikwad,pb}@cse.iitb.ac.in

## Abstract

Machine translation between linguistically dissimilar languages poses a significant challenge, particularly due to the scarcity of parallel corpora. Neural machine translation (NMT) models represent the current state-of-the-art in this field. However, NMT models demand vast amounts of parallel training data to achieve optimal performance. For many languages, such extensive parallel corpora are unavailable, leading to what is known as the low-resource problem. In this context, we describe techniques in pivot-based machine translation, where a third language, called pivot language, is employed to enhance translation quality between the source and target languages. We investigate the use of a pivot language as one of the source languages in a multi-source translation framework. This paper presents a comprehensive survey of various pivoting techniques, with a primary focus on "Multi-source Pivot-based Neural Machine Translation." We extensively explore state-of-the-art machine translation models and various datasets for English-Indic language translation. We aim to provide a comprehensive guide for low-resource Machine Translation.

## 1 Introduction

India is a land with large linguistic diversity, with 22 languages recognized as scheduled languages in its Constitution, alongside numerous local languages and dialects. Providing information in various Indian languages, for domains such as healthcare, law, and tourism, is essential. While educational materials are mainly in English, many people only understand their mother tongue or regional language. Manual translation is time-consuming, expensive, and requires domain knowledge, especially in fields like healthcare and education. Machine translation offers a promising solution, providing fast, cost-effective translations across various domains.

The performance of Neural Machine Translation (NMT) models has significantly improved with the Transformer architecture (Vaswani et al., 2017). However, they require large amounts of data, which is often scarce for many language pairs (Arivazhagan et al., 2019). In such cases, *Pivoting* is useful, where a source language is translated to an intermediate pivot language, which is then translated into the target language (De Gispert and Marino, 2006; Utiyama and Isahara, 2007). Toral et al. (2019); Yeshpanov et al. (2024); Escolano et al. (2019); Salloum and Habash (2013) suggests that pivoting through a related high-resource language (HRL) helps the translation into a low-resource language (LRL).

Consider the case of language families such as Indic, Turkic, Finnic, etc. where there will be a few high-resource languages (HRL) and the rest of them being low-resource (LRL). In the case of Indic languages, Hindi and Marathi are HRL and LRL, respectively, and in the case of Turkic languages Turkish and Turkmen are HRL and LRL, respectively. In this case, English to HRL translation is of high quality, but English to LRL translation will fare poorly. However, given the high similarity of the HRLs and LRLs, and therefore the relative ease of translation between them, it should be possible to leverage the HRL as a pivot language when translating from English to LRL. However, our preliminary exploration of English to Indic LRL translation using Indic HRLs as pivots yielded poorer results compared to state-of-the-art systems, likely due to discarding the source language sentence, prompting deeper exploration of pivoting using both source and pivots jointly.

## 2 Background

Deep Neural Networks are powerful models for machine learning that have demonstrated good performance on various problems. However, they

face a challenge when used for machine translation, which is a sequence-to-sequence task where the input and output sequences can have different lengths. This is because deep neural networks require fixed-size vectors for both the input and output. To overcome this challenge, an encoder-decoder architecture was introduced for neural machine translation. This architecture consists of an encoder, which is a neural network that encodes the input sequence into a fixed-length vector known as the context vector. The context vector contains all the information of the input sequence. The decoder, which is another neural network, generates the output sequence tokens based on the context vector. By using this encoder-decoder architecture, deep neural networks can be used for the sequence-to-sequence task of machine translation, overcoming the challenge of dealing with variable-length input and output sequences.

### 2.1 Recurrent Neural Networks

Feed-forward neural networks do not consider the sequential order of text data. On the other hand, Recurrent Neural Networks (RNNs) process the data step by step in a sequential manner. This characteristic of RNNs enables (Sutskever et al., 2014) them to effectively capture the sequential information present in the data. A notable paper by (Sutskever et al., 2014) highlighted the effectiveness of RNNs in processing sequential data. This is suitable for a task involving sequential data like text.

The computation of Recurrent Neural Networks (RNNs) involves calculating the hidden state for each time step by considering both the input at the current time step and the hidden state that was generated at the previous time step. This enables RNNs to incorporate information from past time steps and use it to inform their processing at the current time step.

$$h_t = g(W^{hx}x_t + W^{hh}h_{t-1}) \tag{1}$$

In Recurrent Neural Networks used for sequence-to-sequence tasks, such as neural machine translation, the weight matrices $W^{hx}$ and $W^{hh}$ are shared across each time step. Additionally, the initial hidden state for the first time step is set to a zero vector. Once the entire input sequence has been processed by the encoder, the final hidden state at the last time step is taken as the context vector. During the decoding phase, the context vector is given as the previous state to the first time step of the decoder, and the input to the decoder is a start of sentence token. For each subsequent time step, the hidden state of the decoder is calculated based on the previous output token and the previous hidden state. The output token is generated by passing the hidden state at each step through a linear layer with an output size equal to the vocabulary size. After applying a softmax function, a probability distribution over the vocabulary is obtained, and the word with the highest probability is chosen as the output.

$$y_t = W^{yh}h_t \tag{2}$$

### 2.2 Attention

Although Recurrent Neural Networks are capable of capturing the sequential information of data, a potential issue with RNNs is that they store the entire context of the input sequence in a single fixed-size vector. This can result in a bottleneck problem, particularly for long input sequences, as the fixed-size vector may not be able to fully contain all the necessary contextual information. To address this limitation, an attention mechanism was introduced. This mechanism, introduced in a paper by (Bahdanau et al., 2014) in 2014, allows the model to focus on specific parts of the input sequence at each decoding step, rather than relying on a single fixed-size vector to encode the entire input sequence.

The function of the encoder in a Recurrent Neural Network remains unchanged, and its role is still to encode the input sentence. At each time step, the hidden state of the encoder is computed based on the input at the current time step and the hidden state generated at the previous time step. This enables the encoder to take into account the sequential nature of the input sequence and incorporate information from previous time steps into the encoding of the current time step.

$$h_t = f(x_t, h_t - 1) \tag{3}$$

In a Recurrent Neural Network, the decoder produces an output token at each time step based on the decoder's hidden state, which is determined by the previous hidden state, the previous output token, and the context vector. Unlike in a traditional RNN where the context vector is a fixed-size vector computed by the encoder, in the case of the attention mechanism, the context vector changes for each time step during the decoding process. It is calculated based on the weighted sum of the encoder's hidden states, where the weights are determined

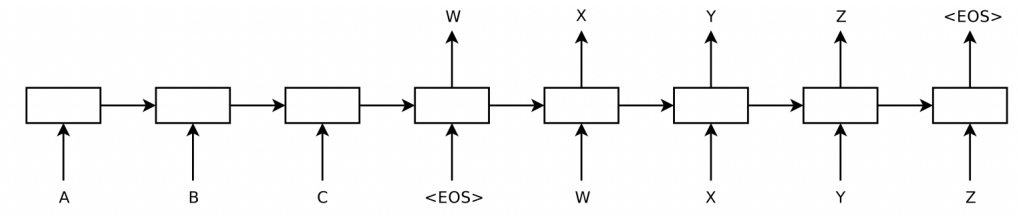


Figure 1: Working of a RNN

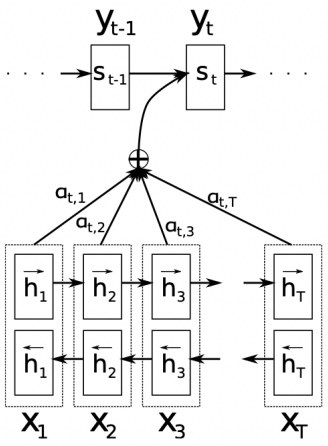


Figure 2: Working of a bi-directional RNN using attention mechanism

by a soft alignment score between the decoder's hidden state at the current time step and each of the encoder's hidden states. This allows the decoder to focus on different parts of the input sequence at each time step and produce more accurate output tokens.

$$s_i = f(s_{i-1}, y_{i-1}, c_i) y_i = g(y_{i-1}, s_i, c_i) \quad (4)$$

The context vector $c_i$ is computed as a weighted sum of the hidden vectors $h_i$ at each time step.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (5)$$

The attention weights $\alpha_{ij}$ are computed by applying the softmax operation over the alignment scores $e_{ij}$ between the *ith* output position and the *jth* input position.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \quad (6)$$

The attention scores, denoted as $e_{ij}$, are calculated based on the relationship between the decoder hidden state at the previous time step and each of the encoder hidden states at the current time step. For each time step of the decoder, a set of attention scores is computed that reflects the similarity between the decoder's hidden state and each of the encoder's hidden states. These attention scores indicate how relevant the input sequence is to the output sequence at each time step of the decoder. By weighting the encoder's hidden states according to these attention scores, the decoder can focus more on the most relevant parts of the input sequence and produce more accurate output tokens.

The attention mechanism enhances the decoder's ability to focus on key elements of the input sequence that are particularly relevant for generating the current output token. These relevant elements are assigned higher attention scores, causing their associated hidden states to have a greater influence on the context vector at the current decoding time step.

### 2.3 Transformers

Recurrent neural networks compute the hidden state at each time step sequentially, which means that the current hidden state depends on the previous hidden state. This sequential computation causes a delay, as the hidden state at time step *t* cannot be computed until the hidden states till time *(t-1)* are computed. This makes the encoding process slow and non-parallelizable. To address this limitation, the Transformer architecture was introduced. The Transformer architecture has significantly improved the performance of neural machine translation and has achieved state-of-the-art results on translation tasks for different language pairs.

The Transformer architecture(Vaswani et al., 2017) introduced a novel approach to encode input sequences using only the attention mechanism. In contrast to recurrent neural networks, the Transformer performs self-attention over the entire input sequence to compute the hidden state at the current time step, which allows for parallel computation of hidden states for all time steps. This eliminates

the need for sequential processing, making both training and inference faster than with recurrent neural networks. The Transformer has achieved state-of-the-art results on machine translation tasks for various language pairs.

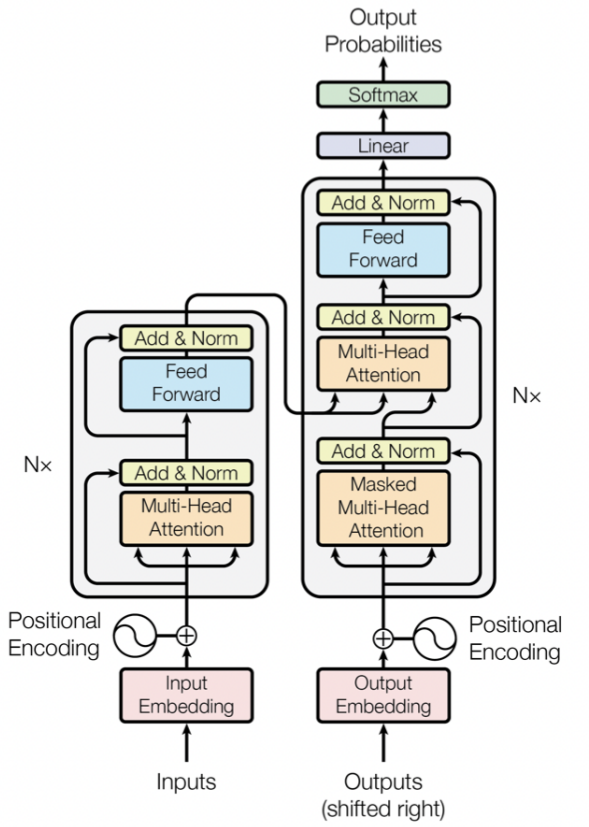


Figure 3: Transformer Architecture

### 2.3.1 Encoder

The transformer encoder is made up of multiple encoder layers, where the output of one layer serves as the input to the next layer. Each encoder layer has two sublayers - a multi-head self-attention layer and a feed-forward neural network layer. Residual connections are made around each encoder layer, and layer normalization is applied at the end.

### 2.3.2 Decoder

The transformer decoder is composed of stacked decoder layers. Each decoder layer contains three sublayers: multi-head self-attention, multi-head cross-attention, and a feed-forward neural network. Like the encoder, residual connections are established around each decoder layer, and then layer normalization is performed.

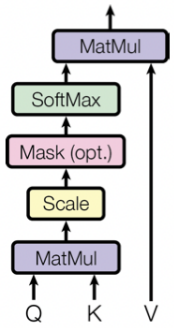


Figure 4: Scaled Dot Product Attention

### 2.3.3 Scaled Dot-Product Attention

The Transformer model uses two types of attention mechanisms, namely self-attention and cross-attention. Both these mechanisms employ scaled dot-product attention, which is based on three input vectors - Query, Key and Value - obtained by matrix multiplication with their respective weight matrices. The attention mechanism computes the compatibility of the query vector with all the key vectors, which generates the attention scores. These scores are then used to compute the weighted sum of all the value vectors, which represent the input positions. The attention computation is as follows,

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (7)$$

The attention mechanism in the Transformer architecture computes the compatibility between the query and key vectors by performing the dot product of the two vectors, and then scaling the value by the square root of the dimensionality of the key vectors, denoted as $\sqrt{d_k}$. This is why this attention mechanism is referred to as scaled dot product attention.

### 2.3.4 Self-Attention

The self-attention mechanism in the Transformer computes the query, key, and value vectors from the same input sequence, allowing it to attend to different positions in the input sequence and generate a representation for each position based on the information from other positions in the sequence. In other words, the self-attention mechanism attends to the input sequence itself to compute the output representation.

### 2.3.5 Cross-Attention

Cross attention mechanism involves computing the query vector from the decoder inputs, and key-

value vectors from the encoder outputs. This enables the decoder to attend over the encoder outputs while generating the output. Therefore, the decoder attends to the input sequence using a cross attention mechanism.

#### 2.3.6 Multi-head Attention

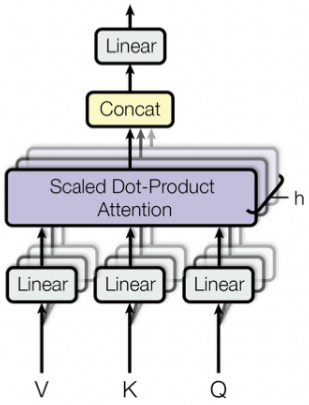


Figure 5: Multi-head Attention

In the Transformer architecture, the attention mechanism is not performed just once, but multiple times using different weight matrices for query, key, and value. Each set of weight matrices is called an attention head. Multiple attention heads allow the model to capture different patterns in the input sequence. The outputs of each attention head are concatenated and passed through a linear layer to get the final attention output.

### 2.4 Pivot language

In MT the language from the text is translated is called source and the language to the text is translated is called target, but in Pivot-based MT a third language called pivot is introduced. This language is usually relatively high-resource and linguistically closer to the low-resource language. This language acts as an intermediate assistant in translation and helps to improve the translation quality.

### 2.5 Parallel Corpus

A Parallel Corpus is a corpus in which there are pairs of sentences, the first sentence in the pair is a sentence in language L1 and the second sentence in the pair is the corresponding translation of the sentence in language L2.

### 2.6 Machine Translation Evaluation

There are two types of evaluation, Human evaluation and Automatic evaluation discussed as follows:

#### 2.6.1 Human Evaluation

An individual evaluator who is fluent in both the source and target languages reviews the machine-translation output. The human evaluator assigns a score to each translated output depending on a predetermined factor. The evaluation conducted might produce a single or several scores based on predetermined criteria. Human evaluation is an expensive and time-consuming process. Human evaluation necessitates the recruitment of human evaluators who are proficient in both the source and target languages. However, human judgment gives a high-quality assessment of machine-translation output. The prevalent scoring technique for machine translation output is based on adequacy and fluency.

**Adequacy** The adequacy of the translation is determined by how well the information or meaning in the source sentence is conveyed in the target sentence. Adequacy is an important criterion for evaluating machine-generated translations because the target sentence must accurately reflect the essence of the source sentence. An evaluator manually assigns an adequacy score depending on whether or not the source sentence's essence is correctly translated to the target sentence.

**Fluency** The fluency of a sentence is calculated by judging how well-structured the sentence is in that language. An extremely fluent sentence is one that a native speaker of the language would produce. A human evaluator scores the fluency of a translated sentence based solely on the quality of the target sentence. A sentence's fluency is determined by the words used the order in which they appear, and the naturalness of the sentence to the native speaker.

### 2.7 Automatic Evaluation

#### 2.7.1 BLEU

Bilingual Evaluation Understudy or BLEU (Papineni et al., 2002), is a metric for assessing the quality of translations produced by machine translation systems. It calculates a score by comparing the machine-generated hypothesis sentences with reference sentences created by human translators. This score reflects how closely the hypothesis matches the reference.

$$p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{n\text{-gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C' \in \text{Candidates}} \sum_{n\text{-gram}' \in C'} \text{Count}_{\text{clip}}(n\text{-gram}')} \quad (8)$$

BLEU employs modified n-gram precision, which involves capping the count of a word in the candidate translation to the word's count in the reference sentence. The BLEU score is determined by calculating the weighted sum of n-gram precision. To address the issue of overly long hypothesis sentences, a brevity penalty factor is applied.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \quad (9)$$

$$BLEU = BP.exp(\sum_{n=1}^{N} w_n log p_n) \quad (10)$$

#### 2.7.2 Subword BLEU

Subword BLEU is an adaptation of the BLEU score that evaluates sentences based on subwords instead of whole words. This method is particularly advantageous for assessing languages with complex morphology, such as those found in India. To calculate Subword BLEU, words in both hypothesis and reference sentences are first segmented into subwords using techniques like Byte Pair Encoding (BPE). The BLEU score is then determined using these subword sequences.

#### 2.7.3 chrF

(Popović, 2015) introduces the chrF score, short for Character level N-gram F-score, which is employed in the automatic evaluation of Machine Translation (MT) systems. The score ranges from 0 to 100 and is computed using the following formula:

$$\text{CHRF}\beta = (1 + \beta^2) \frac{\text{CHRP} \cdot \text{CHRR}}{\beta^2 \cdot \text{CHRP} + \text{CHRR}} \quad (11)$$

## 3 Dataset

The Bharat Parallel Corpus Collection (BPCC) is a comprehensive and publicly available parallel corpus that includes both human-labeled data and automatically mined data, totaling approximately 230 million bitext pairs. This collection covers all 22 scheduled Indic languages and is divided into two main parts: BPCC-Mined and BPCC-Human.

BPCC-Mined contains around 228 million pairs, with nearly 126 million pairs newly added as part of the current work. BPCC-Human comprises 2.2 million gold standard English-Indic pairs, including an additional 644K bitext pairs from English Wikipedia sentences (forming the BPCC-H-Wiki subset) and 139K sentences covering everyday use cases (forming the BPCC-H-Daily subset). Notably, BPCC provides the first available datasets for seven languages and significantly increases the available data for all covered languages.

The contribution from different sources is detailed in the table 1

Additionally, the BPCC includes augmented back-translation data generated by intermediate IndicTrans2 models for training purposes. Detailed information on the selection of sample proportions and sources can be found Gala et al. (2023).

## 4 IndicTransV2

IndicTrans2 (IT2) is the first translation model to support all 22 scheduled Indian languages, trained on the BPCC dataset. The advancements in translation quality achieved in this work, using existing open models, are illustrated in Figure 6

## 5 Pivot-based Neural Machine Translation

In this section, we discuss works related to pivoting and multi-source pivot-based NMT.

Low-resource scenarios lack a direct parallel corpus between source and target languages, meaning there are limited linguistic resources for these language pairs, rendering traditional translation models less effective. This challenge is common in machine translation for many Indian languages spoken by small communities. These languages often share similarities in vocabulary and grammar with more widely spoken Indian languages and can be local variations of high-resource languages. However, they lack good translation tools due to insufficient matching text data for training models. To address this, pivot-based methods can be employed. In these methods, a high-resource language acts as a bridge, or "pivot," between the low-resource language and the target language, leveraging its parallel corpora with both languages to enhance the translation process.

### 5.1 Cascade-based approaches

In the paper titled “A Comparison of Pivot-Based Methods for Machine Translation" authored by

| | En-Indic | | | | | | | Indic-En | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Language* | IT1 | M100 | N1.2 | N54 | IT2 | Goog | Az | IT1 | M100 | N1.2 | N54 | IT2 | Goog | Az |
| *asm_Beng* | 9.9 | - | 13.9 | 15.4 | 19.4 | 16.9 | 16.2 | 32.5 | - | 40.4 | **44.6** | 43.1 | 42 | 35.4 |
| *ben_Beng* | 18.1 | 11.3 | 16.6 | 18.3 | 20.8 | 18.3 | 18 | 33.4 | 26.3 | 36.1 | 39.3 | 39 | **39.8** | 34.9 |
| *brx_Deva* | - | - | - | - | **16.9** | - | - | - | - | - | - | **40.2** | - | - |
| *doi_Deva* | - | - | - | - | **33.5** | 22.2 | - | - | - | - | - | **53.5** | 45.1 | - |
| *gom_Deva* | - | - | - | - | **18.8** | 11.6 | 11.5 | - | - | - | - | **35.3** | 33 | 25.8 |
| *guj_Gujr* | 17.9 | 3.9 | 18.7 | 20.3 | **25.7** | 23.3 | 21.2 | 36.3 | 0.4 | 40.2 | **43.4** | **43.7** | 43 | 37.1 |
| *hin_Deva* | 28.3 | 22.1 | 27.6 | 28.9 | **33.5** | 30.2 | 29.2 | 36.1 | 27.1 | 37.4 | 41 | **42.5** | 39.8 | 36.1 |
| *kan_Knda* | 13.4 | 1 | 13.4 | 14.9 | **18** | 14.2 | 15.2 | 34.8 | 0.1 | 39 | **42.7** | 40.8 | 41 | 35 |
| *kas_Arab* | - | - | 9.9 | 10.5 | **14.4** | - | - | - | - | 31.5 | 35 | **38.3** | - | - |
| *mai_Deva* | - | - | 15.5 | 15.1 | **19.3** | 9.3 | 14.5 | - | - | 37.9 | **41.8** | 40.8 | 39.8 | 36.2 |
| *mal_Mlym* | 13.9 | 4.4 | 11.9 | 13.1 | **16.4** | 13.7 | 13.6 | 31.4 | 17.5 | 34.8 | 38.6 | **41.4** | 37.9 | 33.3 |
| *mar_Deva* | 13.9 | 7 | 14.5 | 15.6 | **21.7** | 16.2 | 17.5 | 33.5 | 20 | 37.2 | **40.8** | **40.2** | **40.6** | 35.1 |
| *mni_Mtei* | - | - | - | - | **17.5** | 10.8 | - | - | - | - | - | **35.1** | 27.5 | - |
| *npi_Deva* | - | 2.6 | 14.4 | 14.8 | 16.8 | 13.8 | 14.4 | - | 12.8 | 42.2 | 46 | 45.1 | **46.8** | 39.9 |
| *ory_Orya* | 10.2 | 0.1 | 12.2 | 11.8 | **14.5** | 10.7 | **14.1** | 36.7 | 0 | 40.7 | **44.7** | 43.8 | 40.4 | 34.7 |
| *pan_Guru* | 23.5 | 7.2 | 23.9 | 25.3 | 25.5 | **29.9** | 25.2 | 33.5 | 10.4 | 37.4 | 40.6 | **41.4** | 39.6 | 34.7 |
| *san_Deva* | - | - | 3.7 | 4.3 | **11.1** | 5.5 | - | - | - | 24.2 | 27.2 | **29.8** | 28.6 | - |
| *sat_Olck* | - | - | 0.0 † | 3.8 | **5.5** | - | - | - | - | 12.3 | 18.7 | **21.8** | - | - |
| *snd_Deva* | - | - | - | - | **14** | - | - | - | - | - | - | **35** | - | - |
| *tam_Taml* | 11.9 | 1.4 | 12.6 | 13 | **14.4** | **14** | **14.5** | 28.9 | 4.9 | 32.5 | 35 | **35.9** | 34.9 | 29.4 |
| *tel_Telu* | 15.5 | - | 15.1 | 17.1 | **19.4** | 17.7 | 17.7 | 33.5 | - | 37.6 | 41.5 | **42.3** | 41.3 | 35.7 |
| *urd_Arab* | - | 23.1 | 42 | 43.8 | 49.7 | 44.1 | **51.4** | - | 26.5 | 46.5 | 50.5 | **53.7** | 50.9 | 46.3 |
| *Avg.* | 16 | 7.6 | 15.6 | 16.8 | 20.3 | 17.9 | 19.6 | 33.7 | 13.3 | 35.8 | 39.5 | 40.1 | 39.6 | 35.3 |
| Δ | 7.3 | 15.8 | 4.8 | 1.7 | - | 4.4 | 2.7 | 8.6 | 29.2 | 4.4 | 0.9 | - | 2.3 | 6.6 |

Figure 6: BLEU scores of all the systems on the IN22-Gen Evaluation set in the En-Indic and Indic-En direction. The best-performing system is bolded, while underlined results indicate significant performance differences where IT2 outperforms the system. Avg means the average score of all the languages that system X supports. Δ represents the difference between the average scores of IT2 and the average scores of system X for the subset of languages that both X and IT2 support. A positive value for Δ indicates IT2 is better than X and vice-versa. † indicates completely off-target translations

| | | | |
|---|---|---|---|
| BPCC-Mined | Existing | Samanantar | 19.4M |
| | | NLLB | 85M |
| | Newly Added | Samanantar++ | 121.6M |
| | | Comparable | 4.3M |
| BPCC-Human | Existing | NLLB | 18.5K |
| | | ICLI | 1.3M |
| | Newly Added | Massive | 115K |
| | | Wiki | 644K |
| | | Daily | 139K |

Table 1: Data for BPCC-Mined and BPCC-Human

Utiyama and Isahara (2007), two distinct pivot-based methods for machine translation are explored, and their performance is evaluated through a comparative analysis with direct translation methods.

**Phrase translation** To create the source-to-target phrase table, the researchers rely on phrase translation probabilities in both directions, along with lexical translation probabilities in both directions for source-to-pivot and pivot-to-target translations. The phrase translation probability is determined using the formula detailed in the paper.

$$\varphi(\bar{f} \mid \bar{e}) = \frac{\text{count}(\bar{f}, \bar{e})}{\sum_{\bar{f}'} \text{count}(\bar{f}', \bar{e})} \tag{12}$$

The calculation of the lexical translation probability is carried out using the following formula:

$$w(f \mid e) = \frac{\text{count}(f, e)}{\sum_{f'} \text{count}(f', e)} \tag{13}$$

$$\text{Ew}(f_i \mid \bar{e}, a) = \frac{1}{|\{j \mid (i,j) \in a\}|} \sum_{\forall (i,j) \in a} w(f_i \mid e_j) \tag{14}$$

$$\text{pw}(\bar{f} \mid \bar{e}, a) = \prod_{i=1}^{n} \text{Ew}(f_i \mid \bar{e}, a) \tag{15}$$

$$\text{pw}(\bar{f} \mid \bar{e}) = \max_{a} \text{pw}(\bar{f} \mid \bar{e}, a) \tag{16}$$

The process for determining the lexical translation probability involves several steps. First, the maximum likelihood of an English word translating to a French word is computed based on word counts using a function denoted as 'w.' Following this, the expectation of a specific word alignment, represented as 'a,' between English and French phrase pairs is calculated. Finally, the probability of the lexical translation is determined by utilizing the alignment that yields the maximum probability.

The calculation of the forward phrase translation probability from the source language to the target language involves utilizing both the forward phrase translation probability from the source language to the pivot language and the forward phrase translation probability from the pivot language to the target language. This combination of probabilities is employed to estimate the likelihood of a given phrase being translated effectively from the source language to the target language via the pivot language, as explained in the study.

$$\theta(f|e) = \sum_{p_i} \theta(f|p_i) * \theta(p_i|e) \tag{17}$$

To calculate the reverse phrase translation probability from the source language to the target language, the study employs the reverse phrase translation probability obtained from the source language to the pivot language and the reverse phrase translation probability from the pivot language to the target language. These probabilities are used in tandem to estimate the likelihood of a phrase being translated in the reverse direction, from the source language to the target language, by way of the pivot language. This approach provides insight into the bidirectional translation dynamics, as outlined in the research.

$$\theta(e|f) = \sum_{p_i} \theta(e|p_i) * \theta(p_i|f) \tag{18}$$

The forward lexical translation probability from the source language to the target language is determined by combining the forward lexical translation probability from the source language to the

pivot language with the forward lexical translation probability from the pivot language to the target language. This combination of probabilities allows for the estimation of the likelihood of a word or phrase being effectively translated from the source language to the target language via the pivot language, as described in the research.

$$\mathrm{p_w}(\bar{f} \mid \bar{g}) = \sum_{\bar{e} \in \mathrm{T_F} \cap \mathrm{T_{EG}}} \mathrm{pw}(\bar{f} \mid \bar{e})\, \mathrm{pw}(\bar{e} \mid \bar{g}) \tag{19}$$

To calculate the reverse lexical translation probability from the source language to the target language, the study employs both the reverse lexical translation probability from the source language to the pivot language and the reverse lexical translation probability from the pivot language to the target language. This combination of probabilities is used to estimate the likelihood of a word or phrase being effectively translated from the source language to the target language in the reverse direction, through the intermediary of the pivot language, as outlined in the research.

$$\mathrm{p_w}(\bar{g} \mid \bar{f}) = \sum_{\bar{e} \in \mathrm{T_F} \cap \mathrm{T_{EG}}} \mathrm{pw}(\bar{g} \mid \bar{e})\, \mathrm{pw}(\bar{e} \mid \bar{f}) \tag{20}$$

**Sentence translation** In the sentence translation method, the entire source sentence is initially translated into the pivot language, which serves as an intermediary. This pivot language translation is then employed to generate the final translation in the target language. This technique necessitates the use of two distinct statistical machine translation systems: one for source-to-pivot translation and another for pivot-to-target translation.

To translate a source sentence into a target sentence, it's first rendered into 'n' pivot language sentences using a source-to-pivot model trained on the source-to-pivot corpus. Here, 'n' is a hyperparameter, and it determines the number of alternative translations produced. Each of these translations is evaluated and assigned scores based on various features, which include forward and reverse phrase translation probabilities, forward and reverse lexical translation probabilities, trigram language model probability, word penalty, phrase penalty, and linear reordering penalty.

The word penalty reflects the cost associated with generating an incorrect word in the translation, while the sentence penalty accounts for the penalty incurred for generating the wrong sentence. The word reordering penalty is applied when the translation places words in an incorrect order. These features collectively contribute to assessing the quality and accuracy of the translation produced by the sentence translation method.

$$f \rightarrow e_i(h_{i1}, h_{i2}, \ldots, h_{i8}) \tag{21}$$

$$\rightarrow g_{ij}(h_{ij1}, h_{ij2}, \ldots, h_{ij8}) \tag{22}$$

$$S(g_{ij}) = \sum_{m=1}^{8} (\lambda_{e,m} h_{i,m}^{(e)} + \lambda_{g,m} h_{ij,m}^{(g)}) \tag{23}$$

$$g = \arg\max_{g_{ij}} S(g_{ij}) \tag{24}$$

In the sentence translation method, a comprehensive function that takes into account all the specified features is computed separately for both the source-to-pivot translation and the pivot-to-target translation. Subsequently, these functions are scaled using weights that are trained. This process results in the selection of the highest-scoring translation for the source-to-target language pair. By weighing the various features and considering both the source-to-pivot and pivot-to-target phases, the method aims to generate the most accurate and contextually relevant translation from the source language to the target language.

**Multiple pivot-based Machine Translation** The paper titled "Leveraging Small Multilingual Corpora for SMT Using Many Pivot Languages" by Dabre et al. (2015) introduces an innovative approach to pivot-based machine translation. This technique leverages multilingual corpora to facilitate translation through the use of multiple pivot languages. The key idea is to address the common problem of insufficient or unavailable parallel corpora between the source and pivot languages by tapping into multilingual data, which is often more accessible in smaller quantities. By doing so, the authors argue that it's possible to create a more robust parallel corpus compared to using a single pivot language, and they provide evidence to support this assertion through their implementation and results.

The study's focus centers on the task of translating from Hindi to Japanese. To achieve this, they employ a three-step method. First, they align

the multilingual corpora using phrase table triangulation. Then, they construct a phrase table, and finally, they build the actual translation system.

In the phrase table triangulation step, the paper discusses the calculation of forward and reverse probabilities. Forward probabilities represent the likelihood of a source phrase being translated into a specific target phrase, while reverse probabilities capture the probability of the target phrase translating back into the source phrase.

Crucially, the paper explains that the forward phrase translation probability from source-to-target is computed by utilizing the forward phrase translation probability of source-to-pivot and the forward phrase translation probability of pivot-to-target. This approach offers an intriguing glimpse into the complexities of their methodology and the way they tackle the challenges associated with multilingual pivot-based translation.

$$\theta(f|e) = \sum_{p_i} \theta(f|p_i) * \theta(p_i|e) \tag{25}$$

$$P_w(f|e,a) = \sum_{p_i} P_w(f|p_i,a_1) * P_w(p_i|e,a_2) \tag{26}$$

The calculation of reverse phrase translation probability, which pertains to translating from the source language to the target language, involves utilizing the reverse phrase translation probability derived from source-to-pivot and the reverse phrase translation probability obtained from pivot-to-target. This process underlines the intricate interplay between the source language, the pivot language, and the target language in the translation model, as explained in the paper.

$$\theta(e|f) = \sum_{p_i} \theta(e|p_i) * \theta(p_i|f) \tag{27}$$

$$P_w(e|f,a) = \sum_{p_i} P_w(e|p_i,a_2) * P_w(p_i|f,a_1) \tag{28}$$

In their work, the process is iterated for each individual pivot language, resulting in the generation of a substantial parallel corpus that encompasses a wide array of potential translations for every phrase. Simultaneously, the researchers compute the forward and reverse probabilities for these translations. The paper discusses several techniques for fusing these distinct phrase tables, including linear interpolation, fillup interpolation, and the incorporation of multiple decoding paths.

Within the context of linear interpolation, the researchers adopt an approach where they multiply the probabilities obtained from direct translation with those stemming from pivot-based translation, effectively scaling the probabilities. This method enables them to assign different weights to the probabilities derived from direct translation and the various pivot languages. Notably, the research places a high emphasis on the direct translation probability, while allocating relatively lower weightage to the pivot translation probabilities. This strategic choice aims to strike a balance that optimally combines the strengths of both direct and pivot-based translations, as elucidated in the paper.

$$\theta(f|e) = \alpha_0 * \theta_d irect(f|e) + \sum_{l_i} \alpha_{l_i} * \theta_{l_i}(f|e) \tag{29}$$

The fillup interpolation technique operates with a distinct focus. Instead of altering the probabilities associated with phrase translations already existing in the direct translation table, it concentrates on populating the vacant slots within the table. Essentially, it introduces phrase pairs that are missing in the direct phrase table and assigns them probabilities calculated using data from pivot languages. This approach aims to comprehensively augment the direct translation table by incorporating phrase translations that would otherwise be absent.

In contrast, the multiple decoding path technique takes a different route. It sidesteps the process of merging tables and, during the decoding stage, refers to all available decoding paths stemming from both the direct and pivot-based phrase tables. By doing so, it strives to yield the most optimal results by considering multiple avenues for translation, thus potentially enhancing the overall translation quality.

### 5.2 Transfer learning

Zoph et al. (2016) introduced a new paradigm in pivoting, called “Pivoting via Transfer learning." The basic approach involves training the model for HRL and then fine-tuning it for LRL.

In the paper titled “Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages" authored by Kim et al. (2019), a novel approach is introduced, centering around transfer learning, to enhance the performance of neural machine translation (NMT) systems. Transfer learning is a technique that revolves around leveraging knowledge acquired while performing

one task and applying it to a related but distinct task.

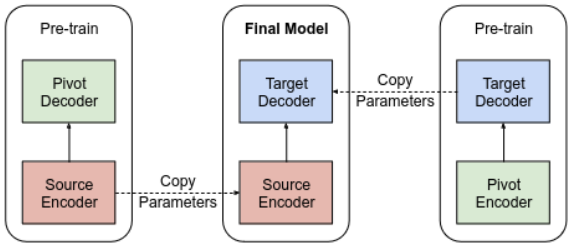


Figure 7: Plain transfer learning

The proposed method is structured into two key stages: pretraining and fine-tuning. In the pretraining stage, the source-to-pivot encoder-decoder is trained using the source-to-pivot parallel corpus. Simultaneously, the pivot-to-target encoder-decoder is also trained using the pivot-to-target parallel corpus.

Moving on to the fine-tuning stage, the source-to-target encoder-decoder model is initialized. This initialization is accomplished by using two main components: the source-to-pivot encoder-decoder and the pivot-to-target encoder-decoder. This step of initializing the source-to-target model can be achieved through three distinct approaches: step-wise pretraining, pivot adapter, and cross-lingual encoder. These techniques play a pivotal role in adapting the model for effective translation between non-English languages, as described in the paper.

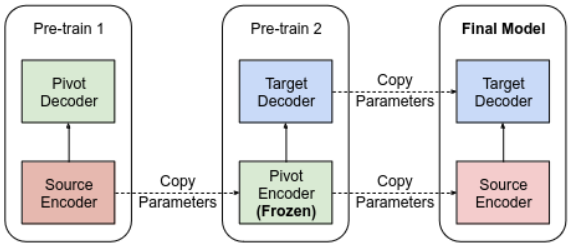


Figure 8: Step-wise pretraining

**Step-wise pretraining** In the step-wise pretraining approach, the process unfolds in a sequential manner. First, the source-to-pivot encoder-decoder is trained using the source-to-pivot parallel data. Subsequently, the pivot-to-target encoder-decoder is trained using the pivot-to-target parallel data. In this method, the model is initialized using the parameters of the source-to-pivot encoder-decoder model.

During the training phase of the pivot-to-target encoder-decoder, a particular strategy is employed. Only the decoder parameters are updated, while the encoder parameters are held constant (frozen). By implementing this, the result is an encoder-decoder model wherein the encoder is adept at encoding the source language, and the decoder excels at translating into the target language. Following this training, the encoder parameters from this model are transferred to the encoder of the source-to-target model, and the decoder parameters are copied to the source-to-target decoder. Consequently, the source-to-target model is fine-tuned using a smaller dataset of source-to-target parallel data, aiming to enhance its performance for translation between non-English languages.

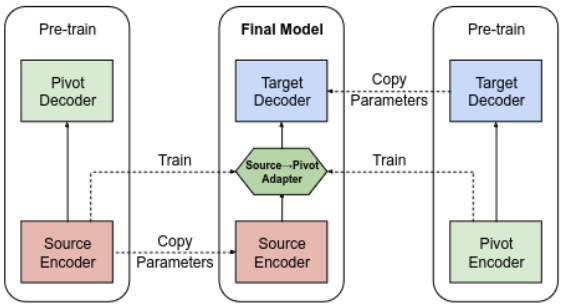


Figure 9: Adapter-based method

**Pivot adapter** In the pivot adapter method, the process follows a specific sequence. Initially, the source-to-pivot encoder-decoder is trained using the source-to-pivot parallel data. Subsequently, the pivot-to-target encoder-decoder is trained using the pivot-to-target parallel data.

Following these training phases, the adaptation process takes place. Specifically, the encoder parameters from the source-to-pivot model are copied into the encoder of the source-to-target model. Simultaneously, the decoder parameters from the pivot-to-target decoder are transferred to the source-to-target decoder.

However, the unique aspect of the pivot adapter method lies in the subsequent adaptation using a source-pivot adapter. This adapter is designed to fine-tune the encoded embeddings of a sentence from the source encoder and the corresponding embeddings from the pivot encoder. Its goal is to scale the source encoder embeddings within the pivot encoder embedding space and minimize the distance between these embeddings.

Once this adaptation process is complete, the model is prepared to perform source-to-target translation, having been adjusted to effectively leverage information from the pivot language in the transla-

tion process.

$$M = \underset{M}{\operatorname{argmin}} \sum_{s,p} \|Ms - p\|_2 \qquad (30)$$

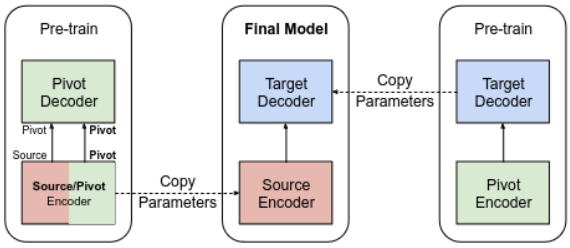


Figure 10: Cross-lingual encoder method

**Cross-lingual encoder** In the cross-lingual encoder approach, the process begins with the training of the source-to-pivot encoder-decoder using the source-to-pivot parallel data. Subsequently, the model undergoes an additional training phase, wherein it learns to encode and decode monolingual pivot data by attempting to decode the same sentences. Following this, the pivot-to-target encoder-decoder is trained using the pivot-to-target parallel data.

Once these training steps are complete, the encoder parameters from the source-to-pivot model are transferred to the encoder of the source-to-target model. Similarly, the decoder parameters from the pivot-to-target decoder are copied to the source-to-target decoder. Notably, the results indicate that the different methods employed to perform translation without the availability of parallel data yield comparable performance to models trained on parallel data.

Among the methods tested, the models that combine step-wise pretraining and cross-lingual encoder exhibit the most promising results, performing exceptionally well. Models trained using the pivot adapter method also deliver competitive results and closely follow the top-performing models. These findings suggest that the proposed approach, which involves combining various techniques and leveraging transfer learning, can be highly effective in improving the performance of neural machine translation systems between non-English languages, as demonstrated in the experiments.

## 5.3 Multi-source approaches

Multi-source NMT is a paradigm where multiple semantically equivalent source sentences from different languages are used to produce translation in target language. We discuss multiple Multi-sourcing techniques through lens of pivoting.

### 5.3.1 Multi-Source Neural Translation

Zoph and Knight (2016); Firat et al. (2016) introduced the multi-source technique in NMT, which exploits multiple source languages to improve translation accuracy in the target language. This method involves using multiple parallel source sentences to produce the target sentence. They employed an encoder-decoder framework with multiple encoders, each dedicated to one of the source languages. The combined representation of these encoders is fed to the decoder to generate the target sentence.

They experimented with three different methods for combining the encoder representations of multiple sources. The first method involves concatenating the encoder representations and feeding them to the decoder. The second approach adds a hidden layer before the decoder, allowing the model to learn weights for this layer, thus giving weightage to each source. The third approach includes an attention mechanism, where hidden states from the top decoder layer can look back at the top hidden states from the encoder. The top decoder hidden state is then combined with a weighted sum of the encoder hidden states to create a better-hidden state vector.

These methods resulted in a significant improvement over the one-to-one translation baseline, demonstrating the effectiveness of the multi-source approach.

### 5.3.2 Multi-Source Neural Machine Translation with Missing Data

Nishimura et al. (2018b) leverage an incomplete multilingual corpus to enhance translation quality using multi-source neural machine translation (NMT). In practical scenarios, it's common to have parallel data in multiple languages, while some data is only available in a subset of these languages. They use this incomplete multiway parallel data with the Multi-encoder NMT method proposed by Zoph and Knight (2016) and the mixture of NMT expert techniques from Garmash and Monz (2016) to outperform conventional one-to-one NMT systems. By incorporating a <NULL> token for missing sources during training, their approach effectively utilizes partially available multiway parallel data, resulting in significant improvements in

machine-translation output.

#### 5.3.3 Input Combination Strategies for Multi-Source approach

Libovický and Helcl (2017) investigate various attention mechanisms in the context of multi-source NMT. They present three strategies: serial, parallel, and hierarchical. In the serial strategy, encoder-decoder attention is computed sequentially for each input encoder. The query set for each cross-attention is derived from the preceding self-attention's context vectors. In the parallel combination strategy, each encoder is attended to independently, with the resulting context vectors summed up. All encoders are attended using the same set of queries from the self-attention sub-layer. The hierarchical combination involves computing attention independently for each input, treating the resulting contexts as states for another input, and then computing attention again over these states.

Their evaluation of multi-source MT demonstrates significant enhancements over single-source baselines. However, adversarial evaluation reveals heavy reliance on the English input, with additional source languages primarily influencing minor output modifications.

#### 5.3.4 Transfer Learning for Multi-source approach

Huang et al. (2020) introduced a novel approach to leverage pre-trained models for multi-source Neural Machine Translation (NMT). They noted that directly employing pre-trained models and fine-tuning them for multi-sourcing might lead to catastrophic forgetting. To address this, they proposed a two-stage pre-training method.

In the first stage, the model is trained on monolingual corpora to learn the sequence generation task. Subsequently, in the second stage, the model is trained using parallel data for translation tasks. Finally, fine-tuning is conducted for multi-sourcing. This gradual training process ensures better performance compared to baseline multi-source models.

Their approach demonstrates that this phased pre-training significantly enhances the model's ability to handle multi-source translation tasks effectively.

#### 5.3.5 Multi-Source Neural Machine Translation with Data Augmentation

(Nishimura et al., 2018a) highlight the challenge of training multi-source Neural Machine Translation (NMT) systems due to the scarcity of complete parallel corpora in multiple source languages and the target language. To address this issue, they propose a data augmentation technique wherein they iteratively train two multi-source NMT systems. Initially, they trained the first system to generate synthetic data for missing source languages. Then, they utilize this synthetic data to train a second multi-source NMT system. Subsequently, the second system generates additional synthetic data, which is used to further train the first system. This iterative process continues until both systems' performance converges.

Their approach demonstrates superior performance compared to both traditional multi-source systems trained solely on complete data and one-to-one translation systems.

## 6 Summary and Conclusion

In this paper, we explore pivot-based machine translation for English to low-resource Indic languages using multi-source techniques. We begin by tracing advancements in Neural Machine Translation and discussing the challenges in this field. Next, we review the current state-of-the-art methods and available datasets for English to Indic translation. Focusing on the low-resource scenario, we delve into pivot-based approaches used to address this issue and provide a detailed discussion on multi-source pivoting. We aim for this paper to serve as a valuable guide for those embarking on research in low-resource Machine Translation.

## References


Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George F. Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *CoRR*, abs/1907.05019.

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

Raj Dabre, Fabien Cromieres, Sadao Kurohashi, and Pushpak Bhattacharyya. 2015. Leveraging small multilingual corpora for SMT using many pivot languages. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1192–1202, Denver, Colorado. Association for Computational Linguistics.

Adrià De Gispert and Jose B Marino. 2006. Catalan-english statistical machine translation without parallel corpus: bridging through spanish. In *Proc. of 5th International Conference on Language Resources and Evaluation (LREC)*, pages 65–68.

Carlos Escolano, Marta R. Costa-jussà, and José A. R. Fonollosa. 2019. From bilingual to multilingual neural machine translation by incremental training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 236–242, Florence, Italy. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Jay Gala, Pranjal A Chitale, A K Raghavan, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar M, Janki Atul Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M Khapra, Raj Dabre, and Anoop Kunchukuttan. 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Transactions on Machine Learning Research*.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan. The COLING 2016 Organizing Committee.

Po-Yao Huang, Junjie Hu, Xiaojun Chang, and Alexander Hauptmann. 2020. Unsupervised multimodal neural machine translation with pseudo visual pivoting. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8226–8237, Online. Association for Computational Linguistics.

Yunsu Kim, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, and Hermann Ney. 2019. Pivot-based transfer learning for neural machine translation between non-English languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 866–876, Hong Kong, China. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2017. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–202, Vancouver, Canada. Association for Computational Linguistics.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018a. Multi-source neural machine translation with data augmentation. In *Proceedings of the 15th International Conference on Spoken Language Translation*, pages 48–53, Brussels. International Conference on Spoken Language Translation.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018b. Multi-source neural machine translation with missing data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 92–99, Melbourne, Australia. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Wael Salloum and Nizar Habash. 2013. Dialectal Arabic to English machine translation: Pivoting through Modern Standard Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 348–358, Atlanta, Georgia. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Antonio Toral, Lukas Edman, Galiya Yeshmagambetova, and Jennifer Spenader. 2019. Neural machine translation for English–Kazakh with morphological segmentation and synthetic data. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 386–392, Florence, Italy. Association for Computational Linguistics.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Rustem Yeshpanov, Alina Polonskaya, and Huseyin Atakan Varol. 2024. KazParC: Kazakh parallel corpus for machine translation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9633–9644, Torino, Italia. ELRA and ICCL.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.