Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B
D-70569 Stuttgart

Machine Translation with Transformers

Truong Thinh Nguyen

Master Thesis

Prüfer:	Prof. Dr. Ngoc Thang Vu
Betreuer:	Pavel Denisov

Beginn der Arbeit:	01.02.2019
Ende der Arbeit:	01.09.2019

Abstract

The Transformer translation model (Vaswani et al., 2017), which relies on self-attention mechanisms, has achieved state-of-the-art performance in recent neural machine translation (NMT) tasks. Although the Recurrent Neural Network (RNN) is one of the most powerful and useful architectures for transforming one sequence into another one, the Transformer model does not employ any RNN. This work aims to investigate the performance of the Transformer model compared to different kinds of RNN model in a variety of difficulty levels of NMT problems.

Acknowledgements

I would like to express my deep gratitude to Professor Ngoc Thang Vu and Pavel Denisov - my supervisors - for their patient guidance, encouragement and valuable advice they have provided throughout my enrollment as a master student. I am particularly grateful for the assistance given by Pavel Denisov, who helped me a lot with installing the speech processing toolkit.

Special thanks to Hung Ngo, Vien Ngo, Khiem Nguyen, Kim Anh Nguyen for sharing your orientations and experiences in the lunchtimes we enjoyed together at Mensa.

I would also like to extend my thanks to Maximilian Schmidt, Dirk Väth, Tuan Pham and Ha Nguyen - my dear friends - for making my study complete in joy and warmth.

Last but not least, I wish to thank my family members for their unconditional love and support.

Contents

List of Figures

List of Tables

1 Introduction

With the power of deep learning, Neural Machine Translation (NMT) has been established as the most powerful algorithm to perform machine translation (MT). There has been a significant change in the state-of-the-art techniques for NMT in recent years. Most former NMT models are Sequence-to-Sequence (Seq2Seq) and highly rely on the architecture composed of two recurrent neural networks (RNNs) which are Encoder and Decoder. However, RNNs handle sequences word-by-word sequentially, which prohibits parallelization and creates a problem with learning long-term dependencies within the input and output sequences from memory (Kolen and Kremer, 2001). Hence, the Transformer model replaces RNNs with self-attention layers to get rid of them.

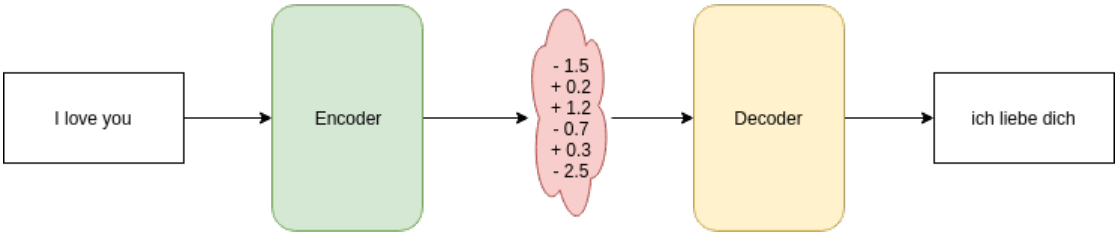

Figure 1: Encoder - Decoder structure translating the English sentence "I love you" to the German sentence "ich liebe dich"

Vaswani et al. (2017) shows that the Transformer model outperforms both Convolutional Neural Network (CNN) and RNN on WMT14 [1] English-to-German and English-to-French translation tasks. To determine whether the Transformer can handle other machine translation problems, this work performed several MT experiments on an increasingly complex scale.

The following three Machine Translation tasks are carried out in this work:

1. Named entity transliteration from 13 different languages to English.

[1] `https://www.statmt.org/wmt14/translation-task.html`

2. Text translation from English to German.

3. Speech translation from English voice to the German text.

In each of these above tasks, there are comparisons between the effectiveness of the Transformer (attention-based) and other RNN architectures such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and bidirectional LSTM (BLSTM - Graves and Schmidhuber (2005)). We observe that the Transformer usually outperforms Recurrent LSTM-based variants on performing MT tasks.

2 Background

2.1 Neural Machine Translation

The Machine Translation of text or speech from one language to another is one of the most popular and challenging goals for computers. Conventional machine translation systems often use rules, which are usually created by linguists at the semantic, syntactic or morphological level. However, the key weakness of rule-based translation systems is that it requires large sets of rules, which is difficult to deal with rule interactions in ambiguity or idiomatic expressions. With the raising concern of deep learning, Neural Machine Translation, which was proposed by (Kalchbrenner and Blunsom, 2013), (Cho et al., 2014) and (Sutskever et al., 2014), has the potential to overcome many limitations of the classical rule-based machine translation approach. Neural machine translation is the art of using artificial neural networks (ANN) models to learn a statistical model for machine translation. The phrase-based translation systems (e.g. Marcu and Wong (2002), Koehn et al. (2003), Setiawan et al. (2005)) require the pipeline of specialized components such as language model, translation model, and reordering model. The structure of the NMT models is simpler than phrase-based models. NMT aims at building and training a single and large ANN that can be tuned to perform language translation effectively (Bahdanau et al., 2014). NMT learns directly the mapping from an input source language to its associated output target language in an end-to-end fashion (Wu et al., 2016).

The architecture of NMT models often consists of an encoder and a decoder (Figure 1). Firstly, each word in the input sentence is fed separately into the encoder to encode the source sentence into an internal fixed-length representation called the context vector. This context vector contains the meaning of the sentence. Secondly, the decoder decodes the fixed-length context vector and then predicts the output sequence. While some types of Encoder-Decoder model used LSTM-based approach (e.g. Sutskever et al. (2014), Luong et al. (2015b)), the others (e.g. Luong et al. (2015a), Vaswani et al. (2017), Galassi et al. (2019)) explored the use

of attention-based architectures for neural machine translation.

2.2 RNN, LSTM and BLSTM

In MT tasks, if we want to predict the next word in a sentence, it is a good idea to know which words come before it. Recurrent Neural Network (RNN) is designed to make use of sequential information. The output y_t at time step t depends not only on its present input x_t but also the entire history of inputs x_0, x_1,..., x_{t-1} from the previous moments. In order to remember the past, RNN introduces hidden states to act as the memory of the network. The hidden state h_t at time step t captures information from all of previous time steps. It is calculated based on the input at the current time step x_t and the previously hidden state h_{t-1}, through a single *tanh* layer as the activation function.

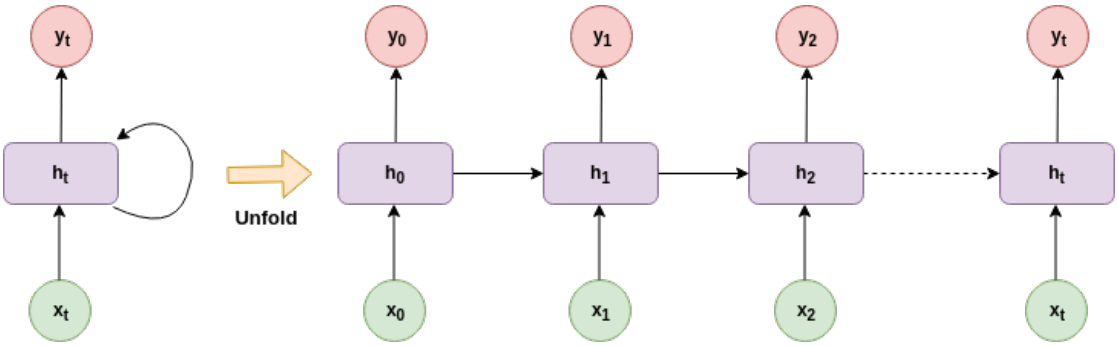

Figure 2: Recurrent Neural Networks with loop (Left) and its unfolded representation (Right)

However, RNN is difficult to solve long-term dependence in practice, which means the information slowly disappears as the number of layers in the neural network increases. One of the most popular solutions for this drawback is to use Long Short-Term Memory (LSTM) network. LSTM is a special kind of RNN using a gating mechanism that controls the work of adding or deleting the memory of the network.

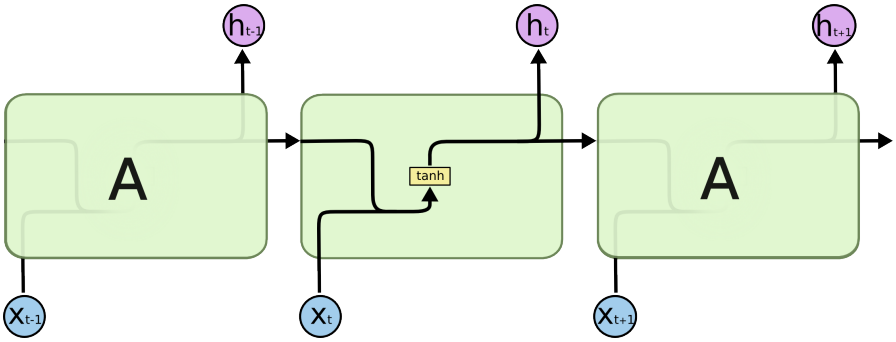

Figure 3: The control flow of a standard RNN.

Figure 3 [2] and Figure 4 [3] show that the control flow of an LSTM network is a chain-like structure which is almost identical with a standard RNN. In addition to the hidden state, each LSTM unit has a cell state to store memory. The gate in LSTM is a component that selectively passes information to an LSTM cell state. A gate consists of a neural network layer with Sigmoid as the activation function and an element-wise multiplication operation. A standard LSTM has three gates learning what information should be added or deleted during training: forget gate, input gate, and output gate. In short, the forget gate controls the amount of information which should be kept from prior steps. The input gate determines how much new relevant information is added from the current step. Lastly, the output gate decides which part of the current cell makes the next hidden state and cell state.

Although LSTM is capable of learning long-term dependencies, it still has limitations since its output is mostly based on previous states. The bi-directional Long Short-Term Memory (BLSTM) (Graves and Schmidhuber, 2005) can process the input sequence in both forward and backward directions. Therefore, input information from the past and future of a point in a given sequence can be combined

[2]Image Credit: Christopher Olah
`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`
[3]Image Credit: Purnasai Gudikandula
`https://mc.ai/recurrent-neural-networks-and-lstm-explained/`

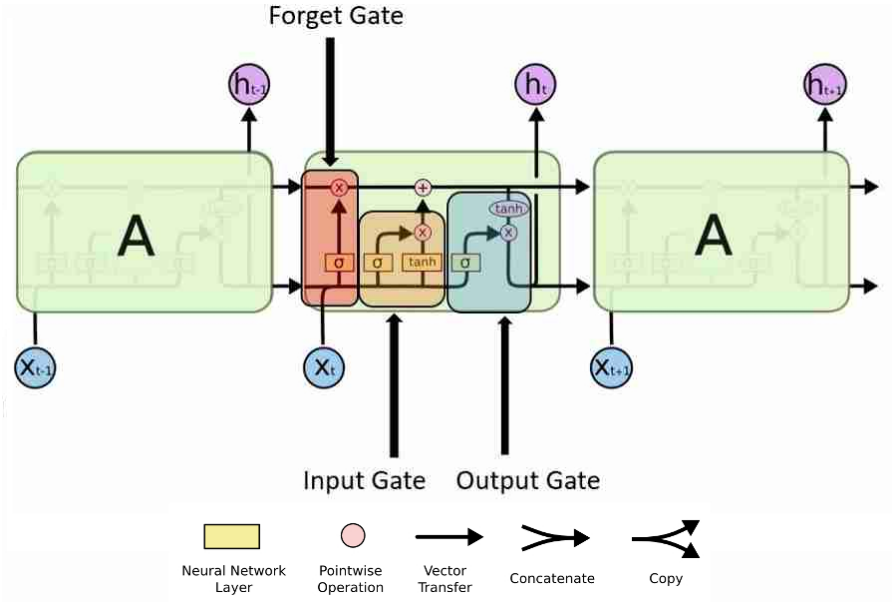

Figure 4: The control flow of an LSTM.

to compute the output sequence.

2.3 Transformer Model

Before Vaswani et al. (2017) introduced the Transformer, RNNs used to be the most popular and powerful architecture for the Encoder-Decoder structure to solve NMT problems. Nowadays, modern and fast computing devices such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) rely on parallel computing. However, the sequential nature of RNNs forces the computation to process sentences word by word, which makes RNNs suitable for parallelization. In order to overcome this shortcoming of RNNs, the Transformer employs a self-attention mechanism that allows the encoder and decoder to account every word of the entire input sequence. Transformer proposes to encode each position, apply self-attention in both decoder and encoder, enhance the idea of self-attention by calculating multi-head attention.

2.3.1 Self-Attention Mechanism

Self-attention in NMT (intra-attention) is an attention mechanism which has the ability to represent relationships between words in a sentence.

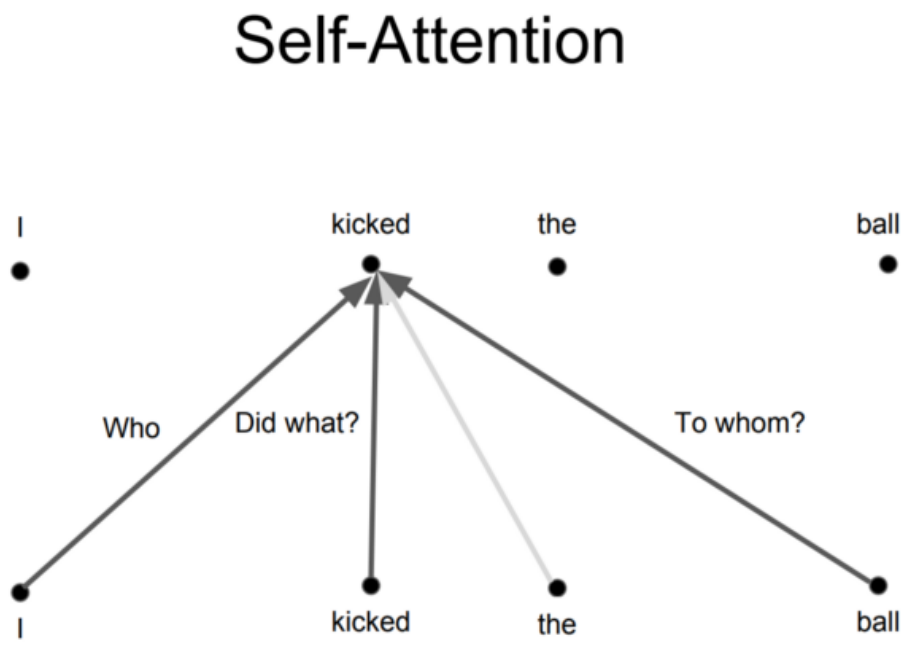

Figure 5: Idea of Self-attention.

Figure 5 [4] shows an example of self-attention mechanism. The model is able to look at the other words in the input sequence - "I", "kicked", "ball" - to get a better understanding of the certain word "kicked" by answering three questions - "Who", "Did what", "To Whom" - respectively. The attention mechanism used by Vaswani et al. (2017) is dot-product attention, which can be described by the following equation:

[4] Image Credit: Ashish Vaswani and Anna Huang
`http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture14-transformers.pdf`

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where Q is a matrix that contains the set of queries packed together, K and V are keys and values matrices. In terms of encoder-decoder, the query and key are usually the hidden state of the decoder and encoder, respectively. Inputs are used as keys and values. The value is a normalized weight representing how much attention that a corresponding key gets.

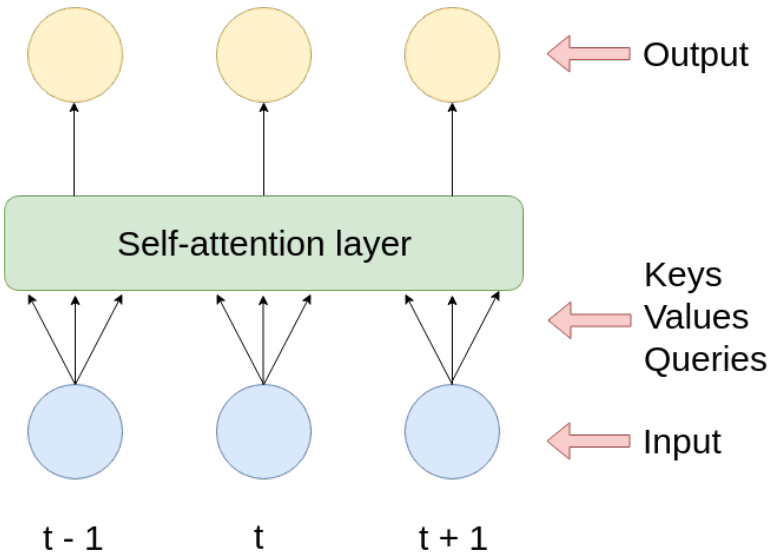

Figure 6: Self-attention layer.

On the encoder side, a better representation of input x_t at time step t is generated by self-attention using all other inputs x_1, x_2,..., x_n, where n is the sequence length. Because this work can be done for all input steps in parallel, the Transformer is more suitable than RNNs for parallelization. Besides, self-attention layer connects all positions with the complexity of O(1) number of sequential operations, cheaper than O(n) of RNNs .

On the decoder side, at time step t, x_t is the current input as the query. All past queries x_1, x_2,...,x_{t-1} are combined to be keys and values of the self-attention

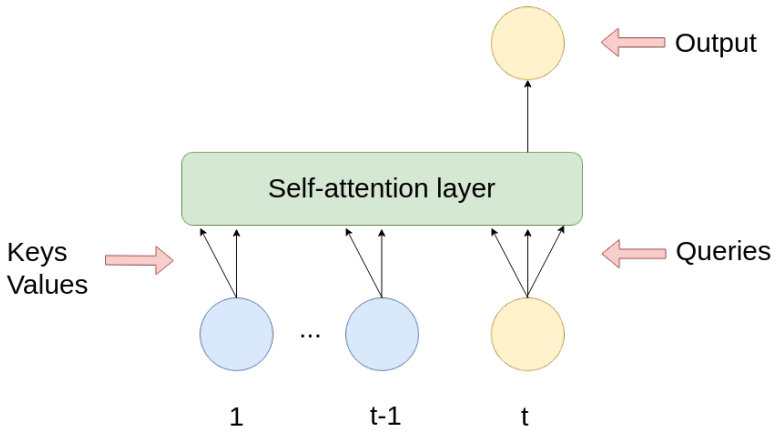

Figure 7: A self-attention layer predicts at time step t

layer.

2.3.2 Multi-Head Attention

Instead of a single attention weighted sum of the values, the Multi-Head Attention computes multiple attention weighted sums to capture various aspects of the input. Through simple splicing of multiple independent attentions, we can obtain information on different sub-spaces.

To learn diverse representations, each head is a unique linear transformation of the input representation as query, key, and value. Then the Scaled-Dot Attention is calculated h times in parallel, making it so-called Multi-Headed. Outputs are then concatenated. Finally, one single linear transformation is applied, as showed in Figure 8. Multi-Head Self Attention can learn related information from different representation sub-spaces because each of these query, key, value sets is randomly initialized.

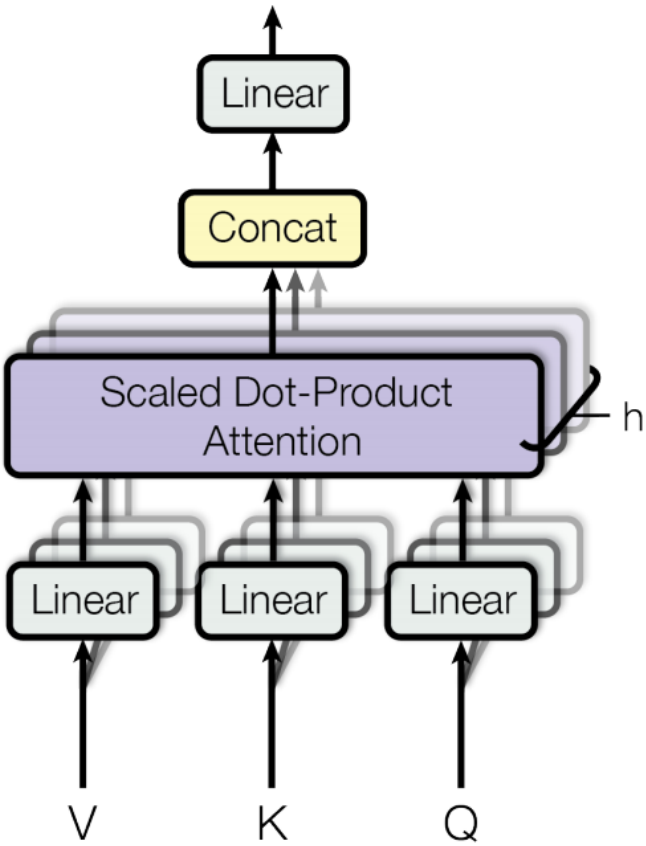

Figure 8: Multi-Head Attention consists of h attention layers running in parallel. (Vaswani et al., 2017)

2.3.3 Positional Encoding

In the case of RNNs, we feed the words sequentially to the model, each token is aware of how it was ordered. However, multi-head attention computes the output of each item in the sequence independently with no notion of word order. It is inefficient to model the sequence information without any special order or position. To account for the order of the words in the input sequence, the Transformer model adds a vector to each input embedding called Positional Encoding. Positional Encoding from the Transformer model is computed by sine and cosine functions of different frequencies as:

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2}$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{3}$$

where i represents the vector index we are looking at, pos represents the position, d_{model} represents the dimension of the input embeddings. Each dimension of the positional encoding forms a sinusoid, as illustrated in Figure 9, allows the model to generalize to longer sequence lengths. PE_{pos+k} can be computed by a linear function of PE_{pos} with an offset k, so the relative position between different embeddings can be inferred at a cheap cost.

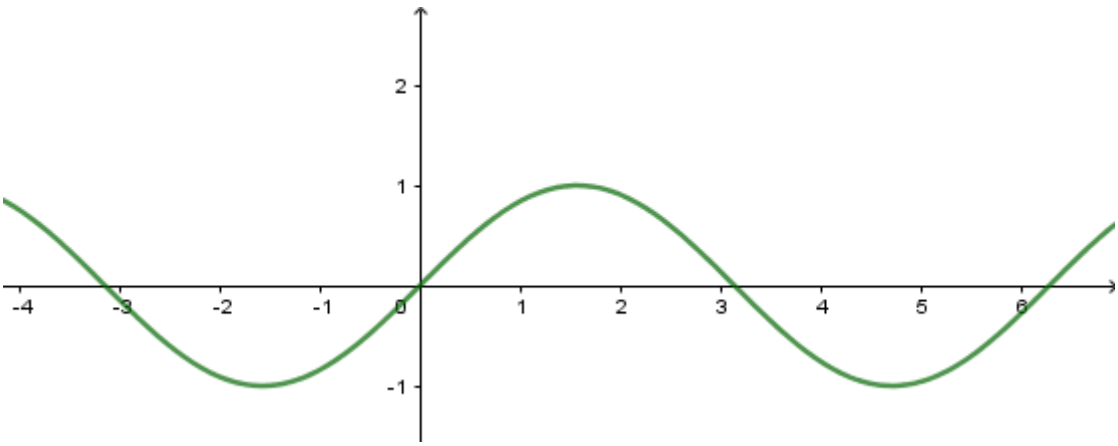

Figure 9: Waveform representation of a sinusoid

Figure 10 describes an example of positional encoding for the input sentence "I love you". Each input word is first turned into a vector using an embedding algorithm of size 4. Positional encoding is then applied to get the corresponding output vector of the same size.

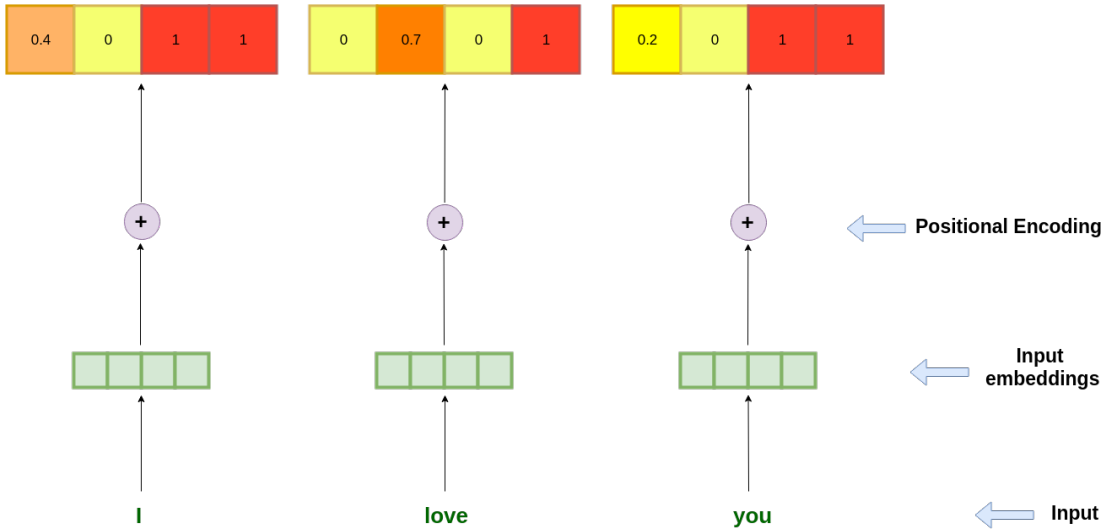

Figure 10: Positional encoding with an embedding size of 4

2.3.4 The Overall Model Architecture

The Transformer model with its encoder and decoder components is illustrated in Figure 11. Both Encoder and Decoder are composed of multiple identical encoders and decoders that can be stacked on top of each other Nx times. The encoder stack and the decoder stack share the same number of Nx.

Encoder: The encoder block is a stack of Nx identical layers. Each layer has a multi-head self-attention mechanism sub-layer followed by a position-wise fully connected feed-forward network sub-layer.

Decoder: The decoder block is also a stack of Nx identical layers. In addition to the two sub-layers in each encoder layer, the decoder has an extra Masked Multi-Head Attention sub-layer to avoid this attention sub-layer looking into the future.

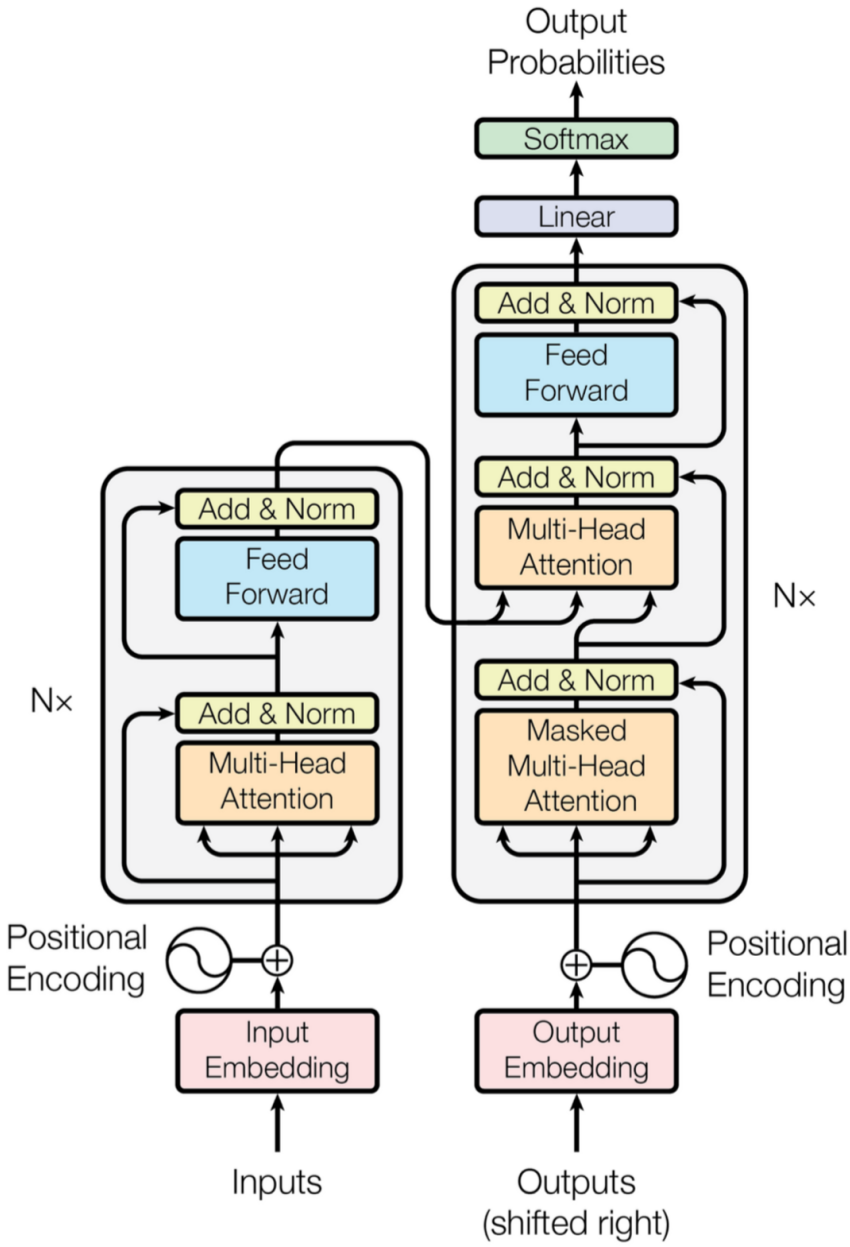

Figure 11: Transformer model architecture with a single layer of Encoder (left) and Decoder (right) (Vaswani et al., 2017)

2.4 Automatic Speech Recognition

Automatic speech recognition (ASR) is a standalone, machine-based process decoding of phonetic transcription (Lai and Yankelovich, 2003). Automatic speech recognition has been a field of research since the 1950s. Recently, we have witnessed a progressive improvement of ASR technologies (Yu and Deng, 2015), (Ravanelli, 2017), especially with the participation of deep learning technology (Hinton et al., 2012). To build a recognition system, several potential choices of open-source toolkits are available: HTK written in C by Young et al. (2002), Sphinx-4 written in Java by Walker et al. (2004), Julius written in C by Lee et al. (2001), RASR written in C++ by Rybach et al. (2011). The most popular ASR tool-kits are built on end-to-end deep learning such as Deep Speech 2 Amodei et al. (2015), ESPnet Watanabe et al. (2018), Kaldi Povey et al. (2011).

This thesis uses ESPnet for the speech recognition task. ESPnet is developed based on Chainer (Tokui and Oono, 2015) and PyTorch (Paszke et al., 2017). For data processing, feature extraction/format, ESPnet also follows the style of Kaldi ASR toolkit, making it convenient to use existing Kaldi recipes.

The standard recipe of ESPnet does not consist of complicated tasks such as lexicon preparation, finite state transducer compilation, alignment, Gaussian mixture modeling, and lattice generation. In total, there are 6 stages in an ESPnet standard recipe, as illustrated in Figure 12:

- **Data preparation**: Using the Kaldi data preparation script.
- **Feature extraction**: Using the Kaldi feature extraction. This stage extracts log Mel feature with 80 dimensions combined with the pitch feature.
- **Data preparation for ESPnet**: Converting all the information about transcriptions, speaker and language IDs, and input and output lengths into one JSON file.
- **Language model training** (*optional*): Training a character-based RNN Language Model.

- **End-to-end ASR training**: Training a hybrid CTC (Graves et al., 2006) and attention-based encoder-decoder model (Watanabe et al., 2017).
- **Recognition and scoring**: Accomplishing speech recognition using previously trained RNN Language Model and end-to-end ASR model.

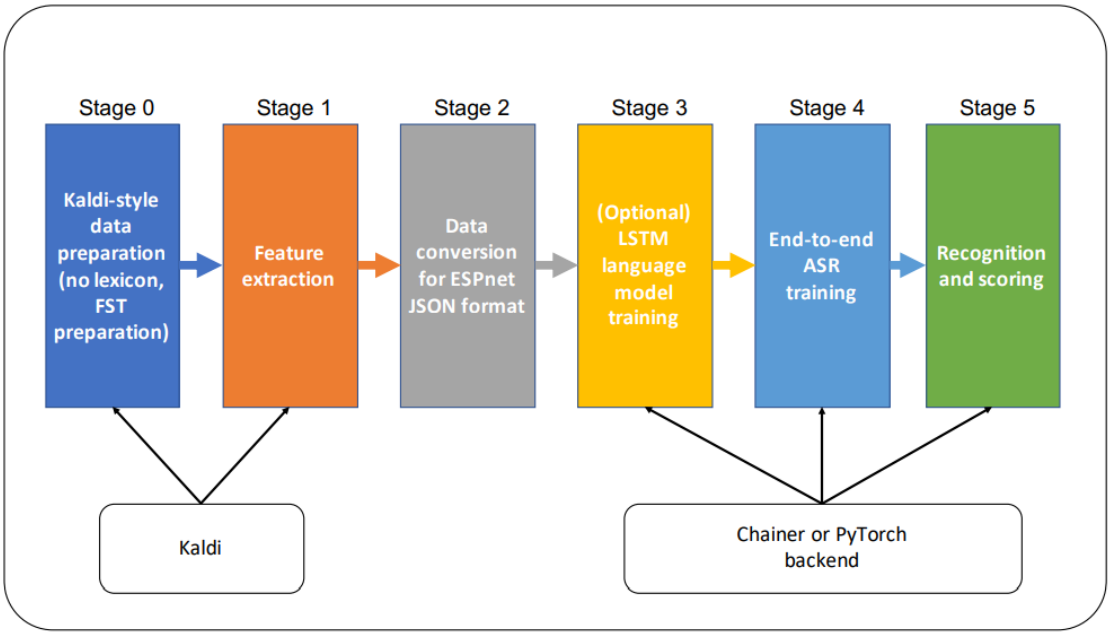

Figure 12: Flow of standard recipes in ESPnet (Watanabe et al., 2018)

3 Multilingual Named Entity Transliteration

Named entity transliteration (NET) is an important sub-task in machine translation (MT). It is the process of taking as input a named entity from one source language then generates a named entity in another language while maintaining their pronunciation (Knight and Graehl (1997), Karimi et al. (2011b)). For example, transliteration of the Greek name $M\epsilon\lambda\beta o\upsilon\rho\nu\eta$ to English is Melbourne.

Source and Tatget words	Letter Correspondence				Description
English to Persian					
John /dʒɒn/	J	o	h	n	*h* is a silent letter (no sound
	|	|		|	is associated to the letter)
جان /dʒɒn/	ج	ا		ن	and is not transliterated
Arabic to English					
نجيب /nædʒiːb/	ن	ج	ی	ب	short vowel /æ/ on N is
	|	|	|	|	normally not written in
Najib /nædʒiːb/	Na	j	i	b	Arabic script
English to Japanese					
Bill /biːl/	B	i	l	l	each syllable in Japanese is a
	\	/	\	/	consonant-vowel sequence
ビル [bi-ru]	ビ		ル		
English to Hindi					
Adam /ˈædəm/	A	d	a	m	the second "a" is not
	|	|		|	transliterated in Hindi
अदम /ˈædəm/	अ	द		म	

Figure 13: Transliteration examples in four language pairs. (Karimi et al., 2011a)

Work in transliteration can be classified into two categories: generative transliteration and transliteration discovery. Transliteration discovery aims at selecting the best candidate for a query name, by discovering already transliterated pairs of words in different languages (Al-Onaizan and Knight (2002), Klementiev and Roth (2006), Kuo et al. (2009)). This thesis focuses on the generative transliteration where the task is to directly map symbols of a source word to a target word (Li et al. (2004), Jiampojamarn et al. (2009), Finch et al. (2015), Upadhyay et al. (2018)).

Machine transliteration has emerged for around two decades and most generative transliteration systems are data-driven (Irvine et al. (2010),Ekbal et al. (2006), Sajjad et al. (2011), Shao and Nivre (2016)). Recently, neural learning systems have become good alternatives to traditional data-driven approaches. Since the LSTM models and the Transformer model have achieved remarkable success in a wide range of natural language processing tasks, this thesis aims to compare the neural encoder-decoder method with LSTM (Sutskever et al., 2014) to the novel Transformer model on the Named Entity Transliteration task. In this work, we use OpenNMT-py (Klein et al., 2017), which is an open-source NMT system, to train an LSTM model and several Transformer models.

3.1 Experiment Setups

3.1.1 Training Data

Name pairs based on Wikipedia have been widely used in transliteration works (e.g., Rosca and Breuel (2016), Pasternack and Roth (2009)). Most Wikipedia pages contain multilingual links among them, which is easy to collect a set of labels that describe the same name entity in different languages. This thesis uses multilingual dataset mined from Wikipedia by Irvine et al. (2010). This dataset consists of 100 languages with overlapping name pages with English. For all experiments, 13 languages of interest are chosen in the same way as Irvine et al. (2010) to transliterate into English. For each language, its dataset is separated into 80% for training, 10% for validation and 10% for testing.

Language	Number of name-pairs	Training	Validation	Test
Russian (ru)	229680	183744	22968	22968
Farsi (fa)	92295	73837	9229	9229
Ukranian (uk)	75054	60044	7505	7505
Arabic (ar)	59356	47486	5935	5935
Korean (ko)	56865	45493	5686	5686
Hebrew (he)	51887	41511	5188	5188
Bulgarian (bg)	41767	33415	4176	4176
Serbian (sr)	28198	22560	2819	2819
Greek (el)	24755	19805	2475	2475
Belarusian (be)	19807	15847	1980	1980
Georgian (ka)	16466	13174	1646	1646
Macedonian (mk)	10227	8183	1022	1022
Old-Belarusian (be-x-old)	9752	7802	975	975

Table 1: Languages of interest and the number of person names paired with English.

3.1.2 LSTM

Every language is trained with the default LSTM model from OpenNMT-py for 100,000 steps. To be specific, some important hyper-parameters in this default model are:

- Number of layers in the encoder = 2
- Number of layers in the decoder = 2
- Number of hidden units in the encoder = 500
- Number of hidden units in the decoder = 500
- Dropout = 0.3

- Learning rate = 1.0
- Decay learning rate = 0.5
- Learning rate warm-up steps = 4000
- Maximum prediction length = 100
- Word embedding size = 500
- Batch size = 64

3.1.3 Transformer

OpenNMT-py provided a set of hyper-parameters for their default Transformer model. Those hyper-parameters are also used in our baseline setting:

- Number of layers in the encoder/decoder $N_x = 6$
- Word embedding size $d_{model} = 512$
- Size of hidden Transformer feed-forward $d_{ff} = 2048$
- Number of heads for Transformer self-attention $h = 8$
- Number of training steps = 100,000
- Maximum batches of words in a sequence to run the generator in parallel = 2
- Dropout = 0.1
- Batch size = 4096
- Adam optimizer:
  - $\beta_1 = 0.9$
  - $\beta_2 = 0.998$

- Learning rate = 2.0
  - Decay method = *noam*
  - Learning rate warm-up steps = 8000
- Maximum prediction length = 100
- Label smoothing value $\epsilon = 0.1$

Due to the difference in the number of name pairs between English and other source languages, different numbers of batch size are assigned while training:

Language	**Batch Size**
Russian (ru)	4096
Farsi (fa)	1024
Ukranian (uk)	1024
Arabic (ar)	1024
Korean (ko)	1024
Hebrew (he)	1024
Bulgarian (bg)	512
Serbian (sr)	512
Greek (el)	512
Belarusian (be)	256
Georgian (ka)	256
Macedonian (mk)	128
Old-Belarusian (be-x-old)	128

Table 2: Batch size for each source language in Named Entity Transliteration task

The Transformer model is highly sensitive to hyper-parameters as described by Popel and Bojar (2018). Therefore, several Transformer models are experimented, including the default OpenNMT-py Transformer model (baseline) and the others called A, B, C, D as described in Table 5. Hyper-parameters such as number of layers in decoder/encoder, word embedding size, feed-forward hidden size, number of attention heads... are tuned in the setting of each model.

3.2 Evaluation Metric

For evaluation, the **Normalized Edit Distance** metric, which is a normalized version of Levenshtein Edit Distance, is used to compute the similarity between a pair of names. Levenshtein Edit Distance is the minimum number of single-character edit operations required to change one word into another. The editing operations include insertions, deletions, and substitutions.

Given two strings *a,b* of length $|a|$ and $|b|$ respectively. The Levenshtein Edit Distance $lev_{a,b}$ between a and b is defined by:

(4)

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where $lev_{a,b}(i,j)$ indicates is the distance between the first *i, j* characters of a and b respectively.

Example: The Levenshtein Edit Distance between "Mannhaton" and "Manhattan" is 3, as we need three editing operations to transform the first one into the second one, and there is no way to do it with fewer than three edits

- "Mannhaton" → "Manhaton"

  1 operation of deletion of first "n".

- "Manhaton" → "Manhatton"

  1 operation of insertion of "t" at the end of sub-string "Manha".

- "Manhatton" → "Manhattan"

  1 operation of substitution of "a" for "o".

Normalized edit distance: To compute Normalized edit distance, we first normalize Levenshtein Edit Distance by the length of the reference string, then multiply the result by 100.

3.3 Experimental Results

Language	Average Edit Distance	Average Normalized Edit Distance
Russian (ru)	1.83	9.93
Farsi (fa)	0.78	4.89
Ukranian (uk)	1.08	6.43
Arabic (ar)	1.37	8.62
Korean (ko)	1.54	10.24
Hebrew (he)	0.75	4.79
Bulgarian (bg)	1.1	7.43
Serbian (sr)	0.92	5.33
Greek (el)	1.6	8.86
Belarusian (be)	1.61	9.95
Georgian (ka)	1.09	7.5
Macedonian (mk)	2.14	12.92
Old-Belarusian (be-x-old)	1.9	11.92

Table 3: Average Normalized Edit Distance of LSTM models in Named Entity Transliteration task

Models	Transformer hyper-parameters				Source languages												
	N_x	d_{model}	d_{ff}	h	***ru***	***fa***	***uk***	***ar***	***ko***	***he***	***bg***	***sr***	***el***	***be***	***ka***	***mk***	***be-x-old***
baseline	6	512	2048	8	**4.92**	7.33	**6.42**	8.76	12.32	4.65	10.36	8.15	13.01	50.34	16.02	87.71	163.24
(A)	6	256	2048	8	11.47	**4.48**	8.31	8.6	**9.48**	**4.38**	7.22	8.2	8.01	**8.86**	**7.85**	20.3	20.53
(B)	6	256	1024	8	10.44	4.53	8.05	**8.26**	11.67	4.49	**6.81**	7.9	**7.13**	8.92	16.36	8.62	**9.91**
(C)	6	256	1024	4	6.27	24.98	8.24	12.19	11.47	4.48	7.14	**7.63**	11.51	19.14	15.11	**8.47**	10.21
(D)	6	32	1024	8	28.02	42.16	31.89	21.75	21.98	45.67	30.45	25.12	16.37	39.27	41.48	31.45	32.47
(E)	4	128	512	4	22.28	46.58	24.54	42.71	21.47	41.44	20.77	15.52	22.39	26.1	20.51	18.18	19.85

Table 4: Average Normalized Edit Distance of Transformer models in Named Entity Transliteration task

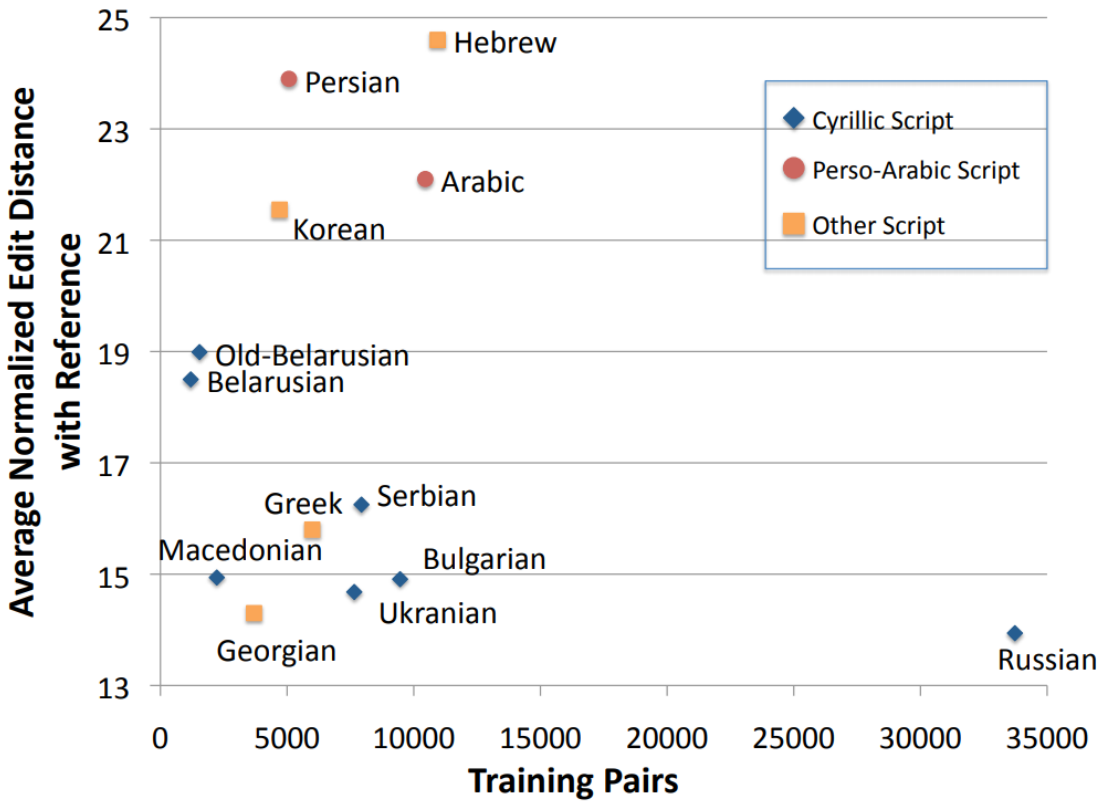

Figure 14: Normalized Edit Distance reported by Irvine et al. (2010) in Transliterating from all languages paper.

Table 3 reports the average Normalized Edit Distances of LSTM models for 13 source languages transliterating into English. The average Normalized Edit Distances of Transformer models are also showed in Table 4. This work uses the same dataset with Irvine et al. (2010). In order to evaluate the quality of the LSTM models and Transformer models in Named Entity Transliteration task, the results from Transliterating from all languages (TFAL) (Irvine et al., 2010) paper is also presented in Figure 14 for later comparison. In TFAL, they trained a statistical transliteration model based on the log-linear formulation of statistical machine translation. Finally, Table 5 shows a comparison between the results from TFAL , the LSTM models and Transformer models.

Language	**TFAL**	**LSTM**	**Transformer**
Russian (ru)	13.8	9.93	**4.92**
Farsi (fa)	24	4.89	**4.48**
Ukranian (uk)	14.8	6.43	**6.42**
Arabic (ar)	22	8.62	**8.26**
Korean (ko)	21.5	10.24	**9.48**
Hebrew (he)	24.7	4.79	**4.38**
Bulgarian (bg)	14.9	7.43	**6.81**
Serbian(sr)	16.2	**5.33**	7.63
Greek (el)	15.8	8.86	**7.13**
Belarusian (be)	18.5	9.95	**8.86**
Georgian (ka)	14.2	**7.5**	7.85
Macedonian (mk)	14.9	12.92	**8.47**
Old-Belarusian (be-x-old)	19	11.92	**9.91**

Table 5: Comparison between the results from TFAL (Irvine et al., 2010), the LSTM models and Transformer models in Named Entity Transliteration task

4 Neural Machine Translation with Seq2Seq

4.1 Experiment Setups

4.1.1 Training Data

To evaluate the effectiveness of the LSTM model and Transformer on the translation tasks, we conduct experiments on the widely adopted benchmark dataset WMT14 English → German translation. We use 4.5M parallel sentence pairs for training set. newstest2014 (2737 sentences) is used as the test set. Validation is done on newstest2013 (3000 sentences) for each of the experiment setups. Tokenization tool [5] from OpenNMT Torch version (Klein et al., 2017) is used to convert raw sentences into sequences of tokens. Being different from Vaswani et al. (2017), all sentences are not encoded using byte pair encoding (BPE) (Britz et al., 2017).

4.1.2 LSTM and BLSTM

Both of the LSTM and BLSTM models share the same hyper-parameters for English → German translation task:

- Number of layers in the encoder = 2
- Number of layers in the decoder = 2
- Number of hidden units in the encoder = 500
- Number of hidden units in the decoder = 500
- Dropout = 0.3
- Learning rate = 1.0
- Decay learning rate = 0.5

[5] `http://opennmt.net/OpenNMT/tools/tokenization/`

- Learning rate warm-up steps = 4000
- Maximum prediction length = 100
- Word embedding size = 500
- Batch size = 64
- Number of training steps = 200,000

4.1.3 Transformer

This thesis experiments the Transformer model using the hyper-parameters suggested by OpenNMT. Those hyper-parameters have been confirmed by Klein et al. (2017) that they have the ability to replicate WMT14 results:

- Number of layers in the encoder/decoder = 6
- Word embedding size = 512
- Size of hidden transformer feed-forward = 2048
- Number of heads for transformer self-attention = 8
- Number of training steps = 200,000
- Maximum batches of words in a sequence to run the generator in parallel = 2
- Dropout = 0.1
- Batch size = 4096
- Adam optimizer:
  - $\beta_1 = 0.9$
  - $\beta_2 = 0.998$

- Learning rate = 2.0
  - Decay method = *noam*
  - Learning rate warm-up steps = 8000
- Maximum prediction length = 100
- Label smoothing value $\epsilon = 0.1$

This model is very similar to base Transformer by Vaswani et al. (2017), such that all projection and multi-head attention layers consist of 512 units followed by a feed-forward layer provided with 2048 units.

4.2 Evaluation Metric

We evaluate all Seq2Seq models using tokenized and BLEU scores (Papineni et al., 2002). Experimental results have proved that BLEU is highly correlated with human judgments (Bojar et al., 2010). Basically, BLEU is the averaged percentage of n-gram matches (typically up to n-grams of length 4). The BLEU score between a reference (*ref*) sentence and a hypothesis (*hyp*) candidate is calculated by:

- Modified *n-gram precision* on corpus:

$$p_n = \frac{\sum_{C \in \{hyp\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{hyp\}} \sum_{n-gram' \in C'} Count(n-gram')} \tag{5}$$

  where $Count_{clip} = min(Count, Max_Ref_Count)$.

- *brevity penalty*:

$$BP = \begin{cases} 1 & if\ |ref| > |hyp| \\ exp(1 - \frac{|ref|}{|hyp|}) & otherwise \end{cases} \tag{6}$$

  where $|ref|$ and $|hyp|$ are length of reference and hypothesis sequences respectively.

- BLEU score:

$$BLEU = BP.\exp(\frac{1}{N}\sum_{n=1}^{N}\log p_n) \tag{7}$$

All reported BLEU scores are computed by the *score.lua* script from OpenNMT Scorer tool [6] with N=4.

[6] `http://opennmt.net/OpenNMT/tools/scorer/`

4.3 Experimental Results

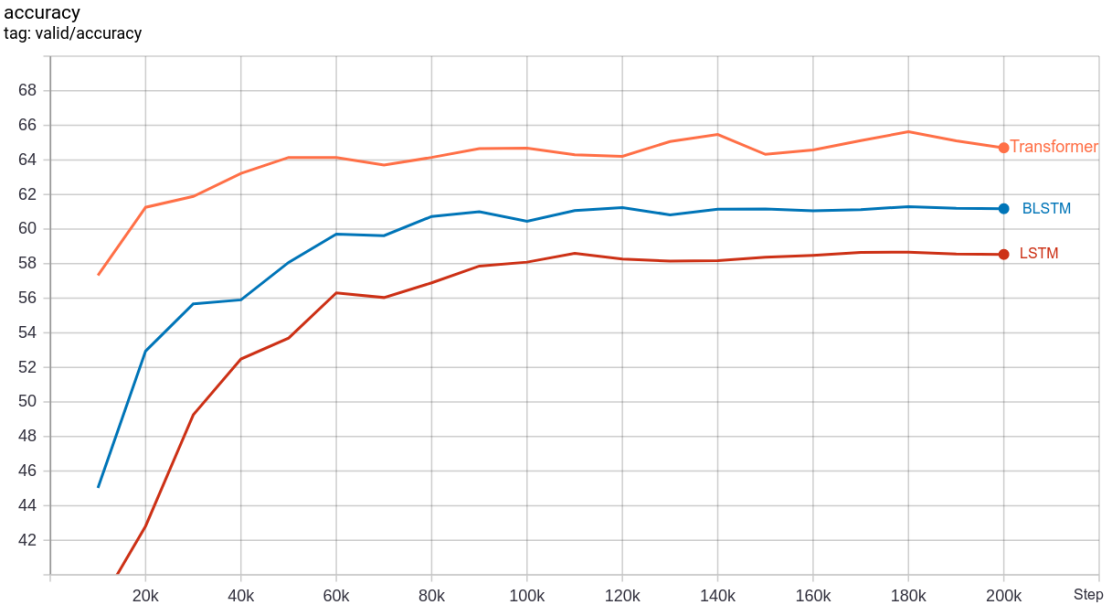

Figure 15: Validation accuracy of LSTM, BLSTM, Transformer models during training

Model	Test BLEU
LSTM	22.21
BLSTM	23.37
Transformer	**25.17**

Table 6: BLEU scores of LSTM, BLSTM, Transformer models on test set *newstest2014* of MT task

On the WMT 2014 English-to-German translation task, the Transformer model outperforms both the LSTM model and the BLSTM model. Table 6 summarizes the translation quality of three architectures: Transformer, LSTM and BLSTM on *newstest2014* test set.

5 Speech Translation

Speech Translation or Speech-to-Text Translation is a process that takes the conversational speech phrase in one language as an input, then outputs translated text phrases in another language.

Traditionally, the cascaded approach has a pipeline consisting of two components connected in a sequential order: an automatic speech recognition (ASR) system and a machine translation (MT) system (Post et al., 2013),(Kumar et al., 2014),(Kumar et al., 2015), (Sulubacak et al., 2018). The ASR is responsible for converting the spoken voices of the source language to the text transcriptions in a similar language. After that, the MT is followed to translates the text in source language the text in the target language.

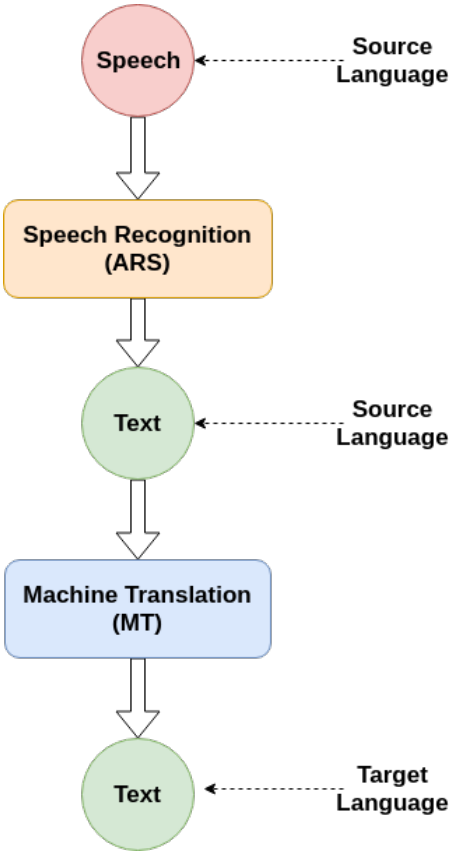

Figure 16: Traditional pipeline of Speech Translation

Recently, end-to-end neural models, seq2seq for example, have showed that they are powerful enough to solve speech translation task (e.g. Berard et al. (2016), Weiss et al. (2017), Duong et al. (2016), Liu et al. (2019)). End-to-end models offer advantages at low resource settings when the voices are in one language while their transcript is in another.

This part of the thesis addresses the translation of English audio into German text by following the traditional pipeline. This approach not only gives benefit from large quantities of text-only corpus WMT14 training data but also makes it easy to evaluate the performance of the Transformer on speech translation task. For the ASR module, ESPnet (Watanabe et al., 2018) - an end-to-end speech processing toolkit is used to perform the speech recognition task on the provided TED-LIUM speech recognition corpus version 2 (Rousseau et al., 2012). Next, several seq2seq text machine translation models are trained for accomplishing the pipeline.

5.1 Experiment Setups

5.1.1 Training Data

For the speech recognition part, TEDLIUM corpus version 2 has been used for training. This data consists of more than 200 hours from TED talks [7] (Cettolo et al., 2013), which allows me to build an ASR system with high performance.

For the machine translation task, WMT14 is used once again together with text-only corpus provides from IWSLT 2018 campaign [8]. These datasets are in regular language with a large amount of punctuation and special characters. Hence, there is a mismatch between these data and the ASR output in the speech translation pipeline. To solve this problem, standard text-based MT sentences are normalized using NLTK toolkit (Loper and Bird, 2002) and transformed to all lowercase letters to reflect ASR output.

[7] `https://www.ted.com/talks`
[8] `https://workshop2018.iwslt.org/`

For the speech translation module, the test sets from the tasks between 2013 and 2015 ("*tst2013*" and *"tst2015"*) provided by the IWSLT organizers are used for testing the performance of the cascade model.

5.1.2 ASR

An automatic speech recognition system is trained by ESPnet with VGG-BLSTMP encoder-decoder, combined with joint connectionist temporal classification (CTC) decoding and RNN language model(LM). The following parameters are used during training:

1. Network architecture

    - etype = vggblstmp *# Encoder architecture type*
    - elayers=6 *# Number of encoder layers*
    - eunits=320 *# Number of encoder hidden units*
    - eprojs=320 *# Number of encoder projection units*
    - subsample= 1_2_2_1_1 *# Encoder subsampling*
    - dlayers=1 *# Number of decoder layers*
    - dunits=300 *# Number of decoder hidden units*
    - atype=dot *# Type of attention*
    - adim=320 *# Number of attention dimensions*
    - aconv_chans=10 *# Number of attention convolution channels*
    - aconv_filts=100 *# Number of attention convolution filters*
    - mtlalpha=0.5 *# Multitask learning coefficient*
    - batchsize=30 *# Batch size*
    - maxlen_in=800 *# Maximum input length for reducing batch size*
    - maxlen_out=150 *# Maximum output length for reducing batch size*

- sortagrad=0 *# Feed samples type*
- opt=adadelta *# Optimizer*
- epochs=15 *# Epochs Number*
- patience=3 *# Patience for optimization*

2. RNN language model

- lm_layers=2 *# Number of LM layers*
- lm_units=650 *# Number of LM hidden units*
- lm_opt=sgd *# LM optimizer*
- lm_sortagrad=0 *# LM Feed samples type*
- lm_batchsize=1024 *# LM batch size*
- lm_epochs=20 *# LM Epochs Number*
- lm_patience=3 *# LM Patience for optimization*
- lm_maxlen=150 *# LM Maximum length for reducing lm_batchsize*

3. Decoding parameter

- lm_weight=1.0 *# Language model weight*
- beam_size=20 *# Beam size*
- penalty=0.0 *# Penalty*
- maxlenratio=0.0 *# Maximum length ratio*
- minlenratio=0.0 *# Minimum length ratio*
- ctc_weight=0.3 *# CTC weight*
- recog_model=model.acc.best *# set a model for decoding*

5.1.3 BLSTM

Hyper-parameters for BLSTM model in English → German text translation task:

- Number of layers in the encoder = 2
- Number of layers in the decoder = 2
- Number of hidden units in the encoder = 500
- Number of hidden units in the decoder = 500
- Dropout = 0.3
- Learning rate = 1.0
- Decay learning rate = 0.5
- Learning rate warm-up steps = 4000
- Maximum prediction length = 100
- Word embedding size = 500
- Batch size = 64
- Number of training steps = 200,000

5.1.4 Transformer

Hyper-parameters for Transformer model in English → German text translation task:

- Number of layers in the encoder/decoder = 6
- Word embedding size = 512
- Size of hidden transformer feed-forward = 2048
- Number of heads for transformer self-attention = 8
- Number of training steps = 200,000

- Maximum batches of words in a sequence to run the generator in parallel = 2
- Dropout = 0.1
- Batch size = 4096
- Adam optimizer:
  - $\beta_1 = 0.9$
  - $\beta_2 = 0.998$
- Learning rate = 2.0
  - Decay method = *noam*
  - Learning rate warm-up steps = 8000
- Maximum prediction length = 100
- Label smoothing value $\epsilon = 0.1$

5.2 Evaluation Metric

To evaluate the performance of an automatic speech recognition system, word error rate (WER) (Popović and Ney, 2007) is the most widely used metric. Basically, the WER (working at the word level) is extended from the Levenshtein distance (working at the phoneme level). The WER between the hypothesis (*hyp*) and the reference (*ref*) sentences is calculated as:

$$WER_{hyp,ref} = \frac{\mathbf{S} + \mathbf{I} + \mathbf{D}}{\mathbf{N}} \tag{8}$$

Where $\boldsymbol{I}$ stands for the number of editing operation insertions (inserting a word), $\boldsymbol{D}$ is the number of deletions, $\boldsymbol{S}$ means the number of substitutions and $\boldsymbol{N}$ is the number of words in the reference.

This thesis uses WER and BLEU score for evaluating the effectiveness of the ASR system and speech translation system, respectively.

5.3 Experimental Results

Utterance	System	Transcription
AimeeMullins 2009P-0002881-0004026	**Our ASR system**	i had already finished editing the piece and i realized that i have never *ones* in my life looked up the words disabled to see what i'd find *when we read* the entry
	IMS-Speech	i had already finished editing the piece and i realized that i had never once in my life looked up the word disabled to see what i'd find let me read you the entry
	Ground truth	i'd already finished editing the piece and i realized that i had never once in my life looked up the word disabled to see what i'd find let me read you the entry

Table 7: Example of our ASR system transcriptions and IMS-Speech transcriptions

Firstly, the performance of the ASR systems is evaluated before conducting any machine translation. In Table 8 we report results on the TED-LIUM 2 dataset for our ASR system, which is built with ESPnet on the TED-LIUM development and test sets, and a web-based speech transcription tool for English and German languages called IMS-Speech (Denisov and Vu, 2019). The speech recognition component of IMS-Speech is implemented with ESPnet toolkit with PyTorch backend and trained on multiple speech databases summed up to 2277 hours of English transcribed speech training data. Due to a large amount of training data, IMS-Speech can compete confidently with other ASR systems on various tasks and conditions. As expected, IMS-Speech achieves better results than our ASR system

on TED-LIUM 2 test set.

set	**Our ASR system**	**IMS-Speech**
dev	19.4	-
test	18.2	**7.3**

Table 8: Word error rates (WER) of our ASR system and IMS-Speech on the TED-LIUM 2 dataset

Secondly, the efficiency of several NMT models on translating ASR-like English to German is shown in Table 9. Table 10 also gives an example of a translation from ASR-like English to German. Once again, the Transformer model outperforms the BLSTM model on this task.

set	**BLSTM**	**Transformer**
dev *(newstest2013)*	17.00	**20.24**
test *(newstest2014)*	16.66	**20.40**

Table 9: BLEU scores of the NMT systems including the Transformer model and the BLSTM model

	Input sentence (English)	orlando bloom and miranda kerr still love each other
Output sentence (German)	BLSTM	orlando bloom und miranda kerr lieben immer noch
	Transformer	orlando bloom und miranda kerr lieben sich noch immer
	Reference	orlando bloom und miranda kerr lieben sich noch immer

Table 10: Example of our NMT models on an ASR-like version of *newtest2014*

The third and last evaluation is about the performance of speech translation systems using different ASR and NMT components. The IWSLT organizers provide a baseline model [9] (Zenkel et al., 2018) of the spoken language translation system. This baseline model includes two different speech recognition systems, a CTC-based system and an attention-based system. For machine translation, it uses an

[9] `https://github.com/isl-mt/SLT.KIT`

Testset	baseline	Our ASR system		IMS-Speech	
	LSTM	BLSTM	Transformer	BLSTM	Transformer
tst2013	13.73	12.52	15.17	16.02	**19.19**
tst2015	-	12.80	15.16	15.91	**19.42**

Table 11: BLEU scores of Speech Translation systems

LSTM model trained with OpenNMT-py. To compare the performance of speech translation systems, we use the test sets used for the IWSLT conference: *tst2013* and *tst2015*. Clearly, Table 11 states that the cascaded speech translation systems which use the Transformer model achieved higher performance than the others which use the BLSTM model as their MT component.

6 Analysis

6.1 Multilingual Named Entity Transliteration

Since transliteration is a simple sub-task of Machine Translation, both the LSTM model and the Transformer model demonstrated the power of deep neural networks when they outperform the word-based n-gram model from Irvine et al. (2010) easily.

	Input (Russian)	Алиев, Рахат Мухтарович
Target (English)	LSTM	Rahat Aliyev
	Transformer	Rakhat Aliyev
	Reference	Rakhat Aliyev

Table 12: Example of transliterating from a Russian name entity to English

In general, we see that the Transformer model can achieve higher (and sometimes competitive) scores than the LSTM approach at the work of transliterating named entities from a source language into English. Serbian (sr) and Georgian (ka) are the only cases in which the LSTM model (slightly) beats the Transformers approach. Due to the fact that both Serbian and Georgian are in the group of Slavic languages and contain small amounts of data, we hypothesize that the Transformer models achieve a high quality of transliterating Slavic languages into English when huge datasets are given. For example, in cases of large datasets such as Russian and Ukrainian, which are also Slavic languages, the Transformer models significantly surpass the performance of the LSTM model (an example is provided in Table 12) and the baseline model. Another explanation is that the Transformer model is extremely responsive to hyper-parameters, there is a possibility that chosen hyper-parameters for Serbian and Georgian are not really fitting. The Transformer models seem to outperform the LSTM models if the hyper-parameters are

carefully tuned in the transliteration task.

6.2 Neural Machine Translation with Seq2Seq

Looking at the BLEU scores given in Table 6, it is evident that the Transformer performs better in all the two RNN-based model variants. The main factor that makes the Transformer model more efficient than both the LSTM model and the BLSTM model is its multi-head self-attention mechanism. Although LSTMs were designed to handle the long-term dependency problem, it still performs not well when the length of sentences is too long. When the distance between words at the current step and previously step increases, the probability of keeping the context decreases exponentially with that distance. Since the self-attention mechanism applies attention weights to all tokens in the input sequences, the Transformer model is able to easily capture long-term dependencies. Besides, the multi-head attention allows the model to capture diverse representations of the input.

	Input sentence (English)	Orlando Bloom and Miranda Kerr still love each other
Output sentence (German)	LSTM	Orlando Bloom und Miranda Kerr immer noch gegenseitig.
	BLSTM	Orlando Bloom und Miranda Kerr lieben immer noch jede andere.
	Transformer	Orlando Bloom und Miranda Kerr lieben einander noch immer.
	Reference	Orlando Bloom und Miranda Kerr lieben sich noch immer

Table 13: Example of our NMT models on *newtest2014* test set

Despite the fact that our Transformer model uses identical settings with the Transformer base model from Vaswani et al. (2017), the BLEU score for it is slightly lower than the one reported in Vaswani et al. (2017). The reason for this is because this thesis intends to determine whether the Transformer model performs MT problems better than other models. Hence, we did not carry out checkpoint averaging, Unicode normalization or tokenization using Moses (Koehn et al., 2007) and byte pair encoding.

6.3 Speech Translation

On the grounds that the cascaded speech translation systems in this thesis are composed of two separate components: an ASR system and a seq2seq NMT model, the performances of the Transformer model and the BLSTM model can be compared obviously. The results in Table 11 demonstrate that the speech translation systems using the Transformer model as their MT component outperform the speech translation systems using the BLSTM model. The reason for these results is the cascaded speech translation system in which the output of a speech recognizer is given and translated using the seq2seq Transformer model as its MT component naturally inherits the advantages of the seq2seq Transformer model as discussed in Chapter 5.

Step		**Text**
Reference *EN*		a mobile phone can change your life
Our ASR system *EN*		mobile phone can change your life
Machine Translation *DE*	BLSTM	handy kann ihr leben ändern
	Transformer	mobiltelefon kann ihr leben verändern
Reference *DE*		ein handy kann ein leben verändern

Table 14: References of an example English speech source and German text target sentence and the output of the different steps of our cascaded speech translation system

Despite the fact that our speech translation system using the Transformer model performs better than the others, its performance is still low. Since the speech translation architecture consists of an ASR component followed by an NMT component, its performance consequently may not be high when the ASR system gets poor quality. Hence, the performance of the speech translation system can be significantly improved by increasing the efficiency of the ASR component. Table 11 also provides systematic evidence in which the cascaded speech translation system using a better ASR system such as IMS-Speech outperforms our speech translation system.

7 Conclusion and Future works

In this work, we demonstrate how the recently introduced Transformer architecture performs in several Machine Translation problems range from easiness to difficultness: transliteration, text translation and speech translation. Our analysis compared the Transformer model with other popular RNN-based models such as the LSTM and the bidirectional-LSTM. To conclude, the Transformer architecture can perform different tasks in Neural Machine Translation with great efficiency, compared to RNN-based models.

A more in-depth analysis of these NMT tasks is going to be carried out in future work. In addition, the performance of these problems using the Transformer architecture need to be improved by applying novel powerful techniques.

Bibliography

Yaser Al-Onaizan and Kevin Knight. Translating named entities using monolingual and bilingual resources. pages 400–408, 01 2002.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015. URL `http://arxiv.org/abs/1512.02595`.

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

Alexandre Berard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *CoRR*, abs/1612.01744, 2016. URL `http://arxiv.org/abs/1612.01744`.

Ondřej Bojar, Kamil Kos, and David Mareček. Tackling sparse data issue in machine translation evaluation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 86–91, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P10-2016`.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017. URL `http://arxiv.org/abs/1703.03906`.

Mauro Cettolo, Jan Niehues, Sebastian Stker, Luisa Bentivogli, and Marcello Federico. Report on the 10th iwslt evaluation campaign. pages 29–38, 01 2013.

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

Pavel Denisov and Ngoc Thang Vu. Ims-speech: A speech to text tool. In Peter Birkholz and Simon Stone, editors, *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2019*, pages 170–177. TUDpress, Dresden, 2019. ISBN 978-3-959081-57-3.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1109. URL `https://www.aclweb.org/anthology/N16-1109`.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 191–198, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P06-2025`.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. Neural network transduction models in transliteration generation. In *Proceedings of the Fifth Named Entity Workshop*, pages 61–66, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3909. URL `https://www.aclweb.org/anthology/W15-3909`.

Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention, please! a critical review of neural attention models in natural language processing. *ArXiv*, abs/1902.02181, 2019.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference*

on Neural Networks, 2005., volume 4, pages 2047–2052 vol. 4, July 2005. doi: 10.1109/IJCNN.2005.1556215.

Alex Graves and Jrgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18:602–10, 07 2005. doi: 10.1016/j.neunet.2005.06.042.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143891. URL `http://doi.acm.org/10.1145/1143844.1143891`.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2205597.

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

Ann Irvine, Chris Callison-burch, and Alexandre Klementiev. Transliterating from all languages. In *In AMTA*, 2010.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. DirecTL: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W09-3504`.

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

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. Machine transliteration survey. *ACM Comput. Surv.*, 43:17, 04 2011a. doi: 10.1145/1922649.1922654.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. Machine transliteration survey. *ACM Comput. Surv.*, 43(3):17:1–17:46, April 2011b. ISSN 0360-0300. doi: 10.1145/1922649.1922654. URL `http://doi.acm.org/10.1145/1922649.1922654`.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL `https://doi.org/10.18653/v1/P17-4012`.

Alexandre Klementiev and Dan Roth. Named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 82–88, New York City, USA, June 2006. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N06-1011`.

Kevin Knight and Jonathan Graehl. Machine transliteration. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 128–135, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. doi: 10.3115/979617.979634. URL `https://doi.org/10.3115/979617.979634`.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557769.1557821`.

J. F. Kolen and S. C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*. IEEE, 2001. ISBN 9780470544037. doi: 10.1109/9780470544037.ch14. URL `https://ieeexplore.ieee.org/document/5264952`.

G. Kumar, M. Post, D. Povey, and S. Khudanpur. Some insights from translating conversational telephone speech. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3231–3235, May 2014. doi: 10.1109/ICASSP.2014.6854197.

Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. A coarse-grained model for optimal coupling of ASR and SMT systems for speech translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1902–1907, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1218. URL `https://www.aclweb.org/anthology/D15-1218`.

Jin-Shea Kuo, Haizhou Li, and Chih-Lung Lin. Harvesting regional transliteration variants with guided search. In Wenjie Li and Diego Mollá-Aliod, editors, *Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, pages 133–144, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-00831-3.

Jennifer Lai and Nicole Yankelovich. The human-computer interaction handbook. chapter Conversational Speech Interfaces, pages 698–713. L. Erlbaum Associates

Inc., Hillsdale, NJ, USA, 2003. ISBN 0-8058-3838-4. URL http://dl.acm.org/citation.cfm?id=772072.772116.

Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Juliusan open source real-time large vocabulary recognition engine. volume 3, pages 1691–1694, 01 2001.

Haizhou Li, Min Zhang, and Jian Su. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 159–166, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1218976. URL https://www.aclweb.org/anthology/P04-1021.

Yuchen Liu, Hao Xiong, Zhongjun He, Jiajun Zhang, Hua Wu, Haifeng Wang, and Chengqing Zong. End-to-end speech translation with knowledge distillation. *CoRR*, abs/1904.08075, 2019. URL http://arxiv.org/abs/1904.08075.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117. URL https://doi.org/10.3115/1118108.1118117.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015a. URL http://arxiv.org/abs/1508.04025.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015b. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL https://www.aclweb.org/anthology/P15-1002.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 133–139, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118711. URL `https://doi.org/10.3115/1118693.1118711`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://doi.org/10.3115/1073083.1073135`.

Jeff Pasternack and Dan Roth. Learning better transliterations. pages 177–186, 01 2009. doi: 10.1145/1645953.1645978.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Martin Popel and Ondrej Bojar. Training tips for the transformer model. *CoRR*, abs/1804.00247, 2018. URL `http://arxiv.org/abs/1804.00247`.

Maja Popović and Hermann Ney. Word error rates: Decomposition over pos classes and applications for error analysis. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 48–55, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626355.1626362`.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. Improved speech-to-text translation with the fisher and callhome spanishenglish speech translation corpus. In *International Workshop on Spoken Language Translation (IWSLT 2013)*, 2013.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. ISBN 978-1-4673-0366-8. IEEE Catalog No.: CFP11SRW-USB.

Mirco Ravanelli. Deep learning for distant speech recognition. *CoRR*, abs/1712.06086, 2017. URL `http://arxiv.org/abs/1712.06086`.

Mihaela Rosca and Thomas Breuel. Sequence-to-sequence neural network models for transliteration. *CoRR*, abs/1610.09565, 2016. URL `http://arxiv.org/abs/1610.09565`.

Anthony Rousseau, Paul Delglise, and Yannick Estve. Ted-lium: an automatic speech recognition dedicated corpus. pages 125–129, 05 2012.

David Rybach, Stefan Hahn, Patrick Lehnen, David Nolden, Martin Sundermeyer, Zoltn Tske, Simon Wiesler, Ralf Schlter, and Hermann Ney. Rasr - the rwth aachen university open source speech recognition toolkit. 12 2011.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. An algorithm for unsupervised transliteration mining with an application to word alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 430–439, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P11-1044`.

Hendra Setiawan, Haizhou Li, Min Zhang, and Beng Chin Ooi. Phrase-based statistical machine translation: A level of detail approach. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Yee Kwong, editors, *Natural Language Processing – IJCNLP 2005*, pages 576–587, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31724-1.

Yan Shao and Joakim Nivre. Applying neural networks to English-Chinese named entity transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 73–77, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2710. URL `https://www.aclweb.org/anthology/W16-2710`.

Umut Sulubacak, Jörg Tiedemann, Aku Rouhe, Stig-Arne Grönroos, and Mikko Kurimo. The memad submission to the IWSLT 2018 speech translation task. *CoRR*, abs/1810.10320, 2018. URL `http://arxiv.org/abs/1810.10320`.

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

Seiya Tokui and Kenta Oono. Chainer : a next-generation open source framework for deep learning. 2015.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. Bootstrapping transliteration with constrained discovery for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 501–511, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1046. URL `https://www.aclweb.org/anthology/D18-1046`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`.

Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Wlfel. Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems*, 12 2004.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition.

IEEE Journal on Selected Topics in Signal Processing, 11(8):1240–1253, 12 2017. ISSN 1932-4553. doi: 10.1109/JSTSP.2017.2763455.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. Espnet: End-to-end speech processing toolkit. In *Interspeech*, pages 2207–2211, 2018. doi: 10.21437/Interspeech.2018-1456. URL `http://dx.doi.org/10.21437/Interspeech.2018-1456`.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. Sequence-to-sequence models can directly transcribe foreign speech. *CoRR*, abs/1703.08581, 2017. URL `http://arxiv.org/abs/1703.08581`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

Steve Young, G Evermann, M.J.F. Gales, Thomas Hain, D Kershaw, Xunying Liu, G Moore, James Odell, D Ollason, Daniel Povey, V Valtchev, and Philip Woodland. *The HTK book*. 01 2002.

Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer, London, 2015. ISBN 978-1-4471-5778-6. doi: 10.1007/978-1-4471-5779-3.

Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Mller, Quan Pham, Sebastian Stker, and Alex Waibel. Open source toolkit for speech to text translation.

The Prague Bulletin of Mathematical Linguistics, 111:125–135, 10 2018. doi: 10.2478/pralin-2018-0011.