CONNECTIONIST MULTI-SEQUENCE MODELLING AND APPLICATIONS TO MULTILINGUAL NEURAL MACHINE TRANSLATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORHAN FIRAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JULY 2017

Approval of the thesis:

CONNECTIONIST MULTI-SEQUENCE MODELLING AND APPLICATIONS TO MULTILINGUAL NEURAL MACHINE TRANSLATION

submitted by **ORHAN FIRAT** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** ________________

Prof. Dr. Fatos T. Yarman Vural
Supervisor, **Computer Engineering Department, METU** ________________

Assist. Prof. Dr. Kyunghyun Cho
Co-supervisor, **Computer Science Dept., New York University** ________________

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan
Computer Science Department, METU ________________

Prof. Dr. Fatos T. Yarman Vural
Computer Engineering Department, METU ________________

Assoc. Prof. Dr. Selim Aksoy
Computer Engineering Department, Bilkent ________________

Assist. Prof. Dr. Aykut Erdem
Computer Engineering Department, Hacettepe University ________________

Assist. Prof. Dr. Emre Akbaş
Computer Engineering Department, METU ________________

Date: ________________

ABSTRACT

CONNECTIONIST MULTI-SEQUENCE MODELLING AND APPLICATIONS TO MULTILINGUAL NEURAL MACHINE TRANSLATION

Firat, Orhan
Ph.D., Department of Computer Engineering
Supervisor : Prof. Dr. Fatos T. Yarman Vural
Co-Supervisor : Assist. Prof. Dr. Kyunghyun Cho

July 2017, 157 pages

Deep (recurrent) neural networks has been shown to successfully learn complex mappings between arbitrary length input and output sequences, called sequence to sequence learning, within the effective framework of encoder-decoder networks. This thesis investigates the extensions of sequence to sequence models, to handle multiple sequences at the same time within a single parametric model, and proposes the first large scale connectionist multi-sequence modeling approach. The proposed multi-sequence modeling architecture learns to map a set of input sequences into a set of output sequences thanks to the explicit and shared parametrization of a shared medium, interlingua.

Proposed multi-sequence modeling architecture is applied to machine translation tasks, tackling the problem of multi-lingual neural machine translation (MLNMT). We explore applicability and the benefits of MLNMT, (1) on large scale machine translation tasks, between ten pairs of languages within the same model, (2) low-resource language transfer problems, where the data between any given pair is scarce, and measuring the transfer learning capabilities, (3) multi-source translation tasks where we have multi-way parallel data available, leveraging complementary information between input sequences while mapping them into a single output sequence and finally (4) Zero-resource translation task, where we don't have any available aligned data between a pair of source-target sequences.

Keywords: Sequence to Sequence, Deep Learning, Neural Machine Translation

ÖZ

BAĞLANTICI ÇOKLU DİZİ MODELLEME VE ÇOKDİLLİ NÖRAL MAKİNA ÇEVİRİSİ UYGULAMARI

Firat, Orhan
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Fatos T. Yarman Vural
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Kyunghyun Cho

Temmuz 2017, 157 sayfa

Derin (yineleyen) yapay sinir ağları, karmaşık ve farklı uzunluktaki girdi - çıktı dizileri arasındaki ilişkiyi modellemede son dönemde etkin bir yöntem olarak öne çıkmışlardır. Bu modelleme, girdi dizisini kodlama, ve kodlanan dizinin çözümlemesi adımlarını izleyen kodlama-çözümleme ağları ile mümkün hale gelmiştir. Bu tez, kodlama-çözümleme ağları mimarisini bir ileri adıma taşıyarak, aynı anda birden fazla girdi - çıktı dizisi arasındaki ilişkiyi modelleyebilen, çok girdi - çoklu çıktı yinelenen yapay sinir ağları modelini önermektedir. Önerilen çoklu girdi - çoklu çıktı yapay sinir ağı mimarisi, tek bir parametrik fonksiyon ile, farklı uzunluktaki birden fazla girdi dizisini, yine farklı uzunluktaki birden fazla çıktı dizisine eşlemeyi etkin bir şekilde öğrenebilmektedir. Bu karmaşık eşleme fonksiyonu, yine bu tez tarafından önerilen, ortak paylaşım alanı (interlingua) sayesinde gerçeklenmekte olup, ortak paylaşım alanı olan dikkat yapay sinir ağı da, bütün girdi-çıktı dizileri arasında paylaşılan parametrik bir fonksiyon olarak sunulmaktadır. Çoklu girdi - çoklu çıktı dizi eşleme mimarisi, bu tez kapsamındaki uygulama alanı olarak, çok-dilli makina çevirisi alanına uygulanmıştır.

Bu kapsamda, önerilen mimari, (1) büyük ölçekli makina çevirisi probleminde, aynı anda on girdi-çıktı çiftini modelleyebilmekte, (2) yetersiz-veri rejiminde, transfer öğrenme kabiliyetine sahip, (3) aynı anda birden fazla girdi dizisini, tek bir çıktı dizisine eşleyebilen, ve bu maksatla girdi dizileri arasındaki tümleyici bilgiyi kullanabilmekte, ve (4) hiç-veri rejiminde, aralarında hiç veri bulunmayan bir girdi ve çıktı dizisi arasında da eşleme yapabilme kabiliyetine sahip bir model olarak önerilmektedir.

Anahtar Kelimeler: Diziden Diziye Eşleme, Derin Öğrenme, Nöral Makina Çevirisi

To *curiosity*, *chili peppers* and *rogue cells*

ACKNOWLEDGMENTS

This thesis had been one of the longest, courageous, selfish, tiring, educating, humbling, devastating and enlightening journey of my life. And over the years I've worked for this thesis, I realized (slowly) how lucky and fortunate I am, always surrounded by loving, supporting and teaching people. It is perhaps true that I've lost some peaces on the way, good or bad, but one thing for sure, I definitely am a different person now as I write these paragraphs, compared to myself when I started this journey in a cold winter day of Ankara. Throughout this journey, I have been dismantled into pieces over and over again, and miraculously, put back together into a more consistent and functioning form on the foundation. Now, it is my turn to thank to the architects.

I would like to thank first to my family and supervisors. To my mother, sister and father; thank you for believing in me. There were times that was the only thing that I had, and the only thing that I needed. I am grateful! I also express my gratitude to my supervisor, Dr. Vural, not only being supportive and guiding but also bearing with me and my troubles, being a second mother to me. I will never forget the moment that I first entered her room as a master's student, and the moment I received my Ph.D. pen from her. Without her support, I wouldn't be writing these lines. Then, of course, my co-supervisor, Dr. Cho. There are no words for me to express my gratitude, literally. His guidance, encouragement, support, vision, spirit and most importantly, friendship walked me towards the lines of this thesis, made me witness the history! I can never exchange the joy of having a drink with him, so from the bottom of my heart, cheers! I also want to thank Dr. Bengio for putting me back in track when I was lost in my research. I learned a lot from him, precious lessons, especially, having the courage to question, question everything. I would also like to thank all the examining committee members for their feedback and support.

Being a lucky person necessitates having the best friends possible, and to be honest, when it comes to having good friends, I am the luckiest and richest person in the world. The list is almost infinite but there are few I have to name here, because they contributed to this thesis at least as the author of this thesis did. Çağlar Gülçehre, Francesco Visin, İsmet Özöztürk, Çağrı Aslum, Cem Eken, Li Yao, Okan Tarhan Tursun, Kelvin Xu, Sebastien Jean, Marcin Moczulski, Yusuf Gür, Ke Tran, Uğur Göçen and many others, thank you for being there with me or for me when I needed, for extending your hands. Also I want to thank Gülcan Can and Esin Kutar for making me a better person. As I was trying to endure the slings and arrows of life, I also realized that I broke so many hearts, here I want to take the chance to apologize, to all the people that I hurt through this journey, please forgive me.

There is one more name I have to say a few more words. Utku, my dear friend, brother. Wherever I was geographically in this world, and that used to change quite frequently for some time, you were the first door that I was knocking, the moment things were going south or the other way. Thank you for being my eyes, scream and tear. I would have been lost without your support.

I would also like to thank all the members of METU-Image Lab, Montreal Institute for Learning Algorithms (MILA), IBM Machine Translation Team, Facebook AI Research Group, NYU CILVR Group and Google Research, thank you for your support.

I finally want to name my dear friends, whom I've lost during this very journey. I cherish the precious memories of you; Mehmet Düzenli, Yaşar Karaağaç, Ömer Bozkurt, Özgür Ekizoğlu, Emre Acar, Taner Erdoğan, Aykut Köroğlu, Serhat Sığnak, Nuri Şener, Uğur Taşçı, Murat Üçöz, Ozan Şarlak... Vode An!

This research was supported in part by the Scientific and Technological Research Council of Turkey with TUBITAK 2214/A Grant Program: 1059B141400301.

Orhan Firat
May 2017
Half Moon Bay, CA

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

NN	Neural Networks
RNN	Recurrent Neural Networks
cRNN	Conditional Recurrent Neural Networks
BP	Back-Propagation
BPTT	Back-Propagation Through Time
GRU	Gated Recurrent Units
cGRU	Conditional Gated Recurrent Units
LM	Language Modelling
RNN-LM	Recurrent Neural Networks Language Models
cRNN-LM	Conditional Recurrent Language Models
seq2seq	Sequence to Sequence Mapping
multi-seq2seq	Multi-Sequence to Sequence Mapping
MT	Machine Translation
NMT	Neural Machine Translation
SGD	Stochastic Gradient Descent
ADAM	Adaptive Moment Matching
BLEU	Bilingual Evaluation Understudy
TER	Translation Error Rate
TB	TER minus BLEU Score

CHAPTER 1

INTRODUCTION

We are not the only smart agents around, far from it. This is what i have defined as the fourth revolution in our self-understanding. We are not at the center of the universe (Copernicus), of the biological kingdom (Darwin), or of the realm of rationality (Freud). After Turing, we are no longer at the center of the world of information and smart agency either. We share info-sphere with digital technologies. These are not the children of some sci-fi super-intelligence, but ordinary artifacts that outperform us in ever more tasks, despite being no cleverer than a toaster. Their abilities are humbling and make us reevaluate our intelligence, which remains unique. We thought we were smart because we could play chess. Now a phone plays better than a chess master. We thought we were free because we could buy whatever we wished. Now our spending patterns are predicted, sometimes even anticipated by devices as thick as a plank.

—LUCIANO FLORIDI

Making sense of sensory information is the main necessity for intelligent agents that learn from data. The sensory information is almost all the time sequentially ordered making the learning process more complicated, since it requires a model for the longitudinal extend of information. Sequence models that are learned from sequential data is then enables agents to conduct decision making, planning and acting accordingly. But the real-world problems involve not only one sequence of data stream, rather *multiple streams of sequences* streaming together. Intelligent agents need to make sense of these sensory stream of multiple sequences, which means at the end, modelling the relations between multiple sequences.

In this thesis, we aim to answer the question "how can we make sense of multiple sequences and their relations when the amount and structure of the data varies".

From the basic description, in a multi-sequence environment, two sequences are involved. An input stream, and the corresponding output stream. This problem can be cast as many applications or sub-problems that intelligent agents tackle, mapping an input sequence into an output sequence. The most common example can be taught as the translation problem when we are given an input sequence and we are asked to map it into an output sequence by preserving the semantics between input and output.

Recently, it has been shown that a deep (recurrent) neural network can successfully learn a complex mapping between variable-length input and output sequences on its own. Some of the earlier successes in this task have, for instance, been handwriting recognition [10, 51] and speech recognition [50, 27]. More recently, a general framework of encoder-decoder networks has been found to be effective at learning this kind of sequence-to-sequence mapping by using two recurrent neural networks [24, 118]. Applications on machine translation, which is a natural sequence to sequence mapping problem, revealed that, simple encoder-decoder architecture suffers from a *bottle-neck* problem, when input and output sequences are long.

In [3], a remedy to this issue was proposed by incorporating an *attention mechanism* to the basic encoder-decoder network. The attention mechanism in the encoder-decoder network frees the network from having to map a sequence of arbitrary length to a single, fixed-dimensional vector. Since this attention mechanism was introduced to the encoder-decoder network for machine translation, neural machine translation, which is purely based on neural networks to perform full end-to-end translation, has become the state of the art, making existing phrase-based statistical machine translation systems obsolete [65, 52, 86].

But the general problem of multi-sequence modelling is much more complex and sophisticated than mapping a single input sequence to a single output sequence since one modality, or one source of information is never enough for the intelligent agents that have the processing capability both in terms of compute and complexity.

Being able to model multiple sequences and their mutual relations in within a single model, is defined as *the multi-sequence modeling problem* in this thesis.

Test bed to benchmark the proposed multi-sequence modelling architectures is chosen to be machine translation because of mainly the availability of the data. This allows us to experiment and control the amount, complexity and complementary nature of the data.

1.1 Main Contributions

This thesis proposes the first large scale multi-sequence modelling architecture that uses a single parametric function to learn the mapping structures between multiple input sequences and multiple output sequences. This single function is parametrized with a single neural network, trained by using data that is semantically aligned between source-target sequences. We propose the notion of the *shared medium*, implemented as a neural network, that learns the commonalities between any number of input sequences and any number of output sequences. We show that, the proposed *shared medium* can be actualized in machine translation as an *interlingua*, and display the benefits of having an *interlingua* representation or machinery. We first show that, the proposed *shared medium* enables a single neural network model to translate from multiple input languages into multiple output languages, one source-target mapping at a time. We then illustrate the transfer learning capabilities of the proposed architecture, with the experiments on source-target mappings where the amount of available training data is low, showcasing the positive language transfer in the proposed architecture. Next, we show that the proposed architecture has an elegant and simple way of mapping semantically similar input sequences into an output sequence, using multiple sources during test time only. Last, we show that, first time in the literature, a neural architecture that has the capability of modelling *interlingua* can do zero-resource translation, where the model can translate between a source and target pair, without seeing any source-target pair data during training.

1.2 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 introduces the basic concepts, building blocks, the structure of the data and problem definitions.
- Chapter 3 introduces the proposed Multi-way Multilingual Sequence to Sequence Mapping architecture along with the Notion of *Shared Medium*.
- Chapter 4 extends the proposed architecture to be able to do Low-Resource Translations.
- Chapter 5 introduces different test time processing strategies to make use of multiple input sequences at the same time, enabling Multi-Source Translation.
- Chapter 6 introduces a simple fine-tuning strategy that enables Zero-Resource Translation.
- Chapter 7 summarizes the main contributions of the research conducted in this thesis, it's impact and proposes several future directions that can be built on top of this work.

CHAPTER 2

BACKGROUND: FROM CONNECTIONIST SEQUENCE MODELLING TO INTERLINGUA

"... the theory to be presented here takes the empiricist, or connectionist position and has been developed for a hypothetical nervous system, or machine, called a perceptron. The perceptrop is designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in the special, and frequently unknown, conditions which hold for particular biological organisms. The analogy between the perceptron and biological systems should be readily apparent to the reader."

FRANK ROSENBLATT - THE PERCEPTRON, 1958

In this chapter, we overview the fundamental building-blocks of connectionist (or neural)[1] multi- sequence modelling, in a bottom-up fashion, from pieces that can be used to represent individual sequences, into a final architecture that can model multiple sequences simultaneously.

First, we look at connectionist sequence models, in particular Recurrent Neural Networks (RNN), that form the basis of Sequence to Sequence Models (seq2seq). We investigate the basic properties of RNN, covering the gradient-based training technique called back-propagation through time (BPTT), it's deficiencies to propagate error signals back in time and Gated Recurrent Units (GRU) which mitigates some of the deficiencies. We show the use of RNN for Language Modelling (LM), the task of

[1] Note that, throughout this thesis, we use the term *neural* to refer *connectionist* (and vice-versa), as neural network models are the most common forms of connectionist approaches.

predicting the next token given a context of previous sequence of tokens, under the framework of RNN Language Modelling (RLM-LM) and conditional RNN Language Modelling (cRNN-LM).

Second, we describe how to combine an RLM with a cRLM, for two sequences that are semantically similar, a critical approach which is known as Sequence to Sequence mapping (seq2seq). We explore two architectures for seq2seq, simple encoder-decoder models and encoder-decoder models with attention, with a special emphasis on attention mechanism.

Next, we provide background knowledge on Neural Machine Translation (NMT), which is chosen to be our test-bed in this thesis. We focus on NMT training using input and output sequence pairs (called parallel corpora), testing methods with Beam-Search decoding, evaluation metrics to assess goodness of translations BLEU, TER and TB.

Finally, we give motivation and background to multi-sequence modelling, the end goal of this thesis. We discuss the possible routes to be taken in order to extend seq2seq models to multi-task seq2seq models first. We enumerate and analyze the representation and scalability problems on the path to multi- sequence modelling. At last, we remind Warren Weaver's Memorandum for Machine Translation (MT), the description and premise of *interlingua* along with benefits and caveats for Multi-lingual Sequence Modelling.

2.1 Connectionist Sequence Modelling

What is connectionism? Connectionism is a wide range of approaches, that try to model cognition using connectionist networks, also known as neural networks [88]. In a connectionist system, simple processing units (neurons) are connected together into large complex networks. The processing is then, characterized by the activation patterns across neurons, and knowledge is stored over the strength of the connections between neurons (weights), hence the name *connectionist* [55]. From a statistical learning perspective, learning itself is then, about operating on the connection weights and adjusting them according to the task at hand. Since learning complex systems is a challenging problem, connectionist approaches put special emphasis on learning internal (intermediate) representations [5]. Learning multiple levels of intermediate representations with increasing abstraction and complexity, is now being called *deep learning* [46, 108, 78] and constitutes the core methodology we use in this thesis.

From our perspective, connectionist approaches are end-to-end computational models that focus on using neural networks for function approximation [59] and we use connectionist approaches to discover statistical regularities over (temporal) sequences. Although connectionism is historically proposed to understand the computations carried out by networks of neurons for mental processes, here in this thesis, we do not investigate or speculate about the biological plausibility of connectionist models and draw conclusions about natural neural networks (eg. human brain).

What is sequence modelling? Most of the human behavior revolve around temporal sequences, due to an artifact of 4th dimension in the universe,[2] such as language, visual understanding, action planning, communication, reasoning and many more. Having a good statistical model over the events occurring in sequences is then becomes crucial for intelligent agents, either natural or artificial.

Let us consider a sequence of events $\mathbf{x} = x_1, x_2, \ldots, x_T$ occurring in discrete time steps of length T. Then, having a statistical sequence model over $\mathbf{x}$, is about estimating how plausible (statistically likely) the sequence is [21], or equivalently, estimating $p(\mathbf{x})$. Two problems immediately arise, (1) how to factorize $p(\mathbf{x})$ to account for all the previous context, and (2) how to parametrize such joint probability distribution as sequence length extends to far in the past, such that $T \to \infty$?

What is connectionist sequence modelling? Using (recurrent) neural networks in order to parametrize $p(\mathbf{x})$ is what we refer as connectionist sequence modelling. When we are concerned about *Finding Structure in Time* [38] in an end-to-end fashion, without limiting the model to have limited context (or pre-determined markov property), connectionist sequence models become the weapon of choice for the following reasons. In a connectionist sequence model, the underlying primitives governing the temporal behavior of the model, are context- sensitive representations of observed events x_i, which are also known as hidden states or activations. This enables the se-

[2] A vague claim, assuming unified four-dimensional space-time of general relativity.

quences to be represented by the sequential trajectories through the activation space. Being able to form trajectories, allows the model to build simpler internal representations that are useful to unpack the structure of more complex sequences (encoding).

After our motivational and verbose introduction on connectionist sequence modelling, let us provide formal description on recurrent neural networks and their linguistic extension, recurrent neural language models.

2.1.1 Recurrent Neural Networks

Let us consider a discrete sequence of observations $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ where T is the sequence length and each observation x_i is a vector representation of input with dimensionality d, ie. $x_i \in \mathbb{R}^d$. To have a probabilistic model of the sequence $\mathbf{x}$, we need to specify a probability distribution over the sequence, in the form of joint probability of its elements x_i as,

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_T). \tag{2.1}$$

We can further rewrite this joint distribution as a product of conditionals, since the sequences we are interested are in sequential order,

$$p(x_1, x_2, \ldots, x_T) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_T|x_{<T}) = \prod_{t=1}^{T} p(x_t|x_{<t}), \tag{2.2}$$

where $p(x_t|x_{<t})$ is the conditional probability of the observation x_t given all the previous (preceding) observations $x_{<t}$ in the sequence, as we call *context*. The decomposition in Eq. (2.2) exhibits two major difficulties.

First, how are we going to parametrize each individual conditional, $p(x_t|x_{<t})$, as sequence length T grows arbitrarily large. In other words, how to take into account the *context*?[3] A simple relaxation, which connectionist models do not have and therefore

[3] Note that, we only focus on parametric models and non-parametric approaches are beyond the scope of this thesis.

not used in this thesis, is resorting to an n-th order Markovian assumption, where dependency beyond a predetermined context-window n is ignored, formally:

$$\prod_{t=1}^{T} p(x_t|x_{<t}) \approx \prod_{t=1}^{T} p(x_t|x_{t-n}, \ldots, x_{t-1}). \tag{2.3}$$

From our perspective, sequence modelling is about capturing the long-term dependencies in the sequential data [8]. Considering only a pre-determined subset of input observations (as in an n-th order Markovian model) invalidates the premise of capturing the long-term dependencies in the data, and therefore we approach the problem as a credit assignment problem [91], and let the model *learn* the extent of these dependencies by making use of algorithms such as *backpropagation* [106].

The second problem is about modelling arbitrary length sequences. How can we generalize to the sequences which are in different lengths, in other words how to handle varying sequence lengths of different examples when we do not know the sequence lengths in advance? Let $T^{(j)}$ and $T^{(k)}$ be the sequence lengths of sequences $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$, where $|\mathbf{x}^{(j)}| = T^{(j)}$ and $|\mathbf{x}^{(k)}| = T^{(k)}$, as $|\cdot|$ is the cardinality operator. Problem emerges with the question, how to formulate Eq. (2.2) with varying T such that $T^{(j)} \neq T^{(k)}$ when $j \neq k$?

Since we care about modelling arbitrary length sequences, without having any assumptions about the structure of the data, and without any Markovian restrictions, we need to address both problems mentioned above.

Recurrent Neural Networks [106, 38] allow representing arbitrary length sequences, while paying attention to the structured properties of the elements in each sequence without any Markov property (or can be stated as infinite Markov property)[21]. RNN directly models the original conditional probabilities in Eq (2.2) and work for any number of *context* inputs, naturally addressing both problems.

Given an input sequence $\mathbf{x} = (x_1, \ldots, x_T)$, an RNN consists of (1) an internal memory $\mathbf{h}_t$, which evolves over time, (2) a recursive function f which operates on the input sequence one by one and the internal memory $\mathbf{h}_t$ and finally (3) an (optional) output function g which predicts the conditional probabilities given the state or inter-

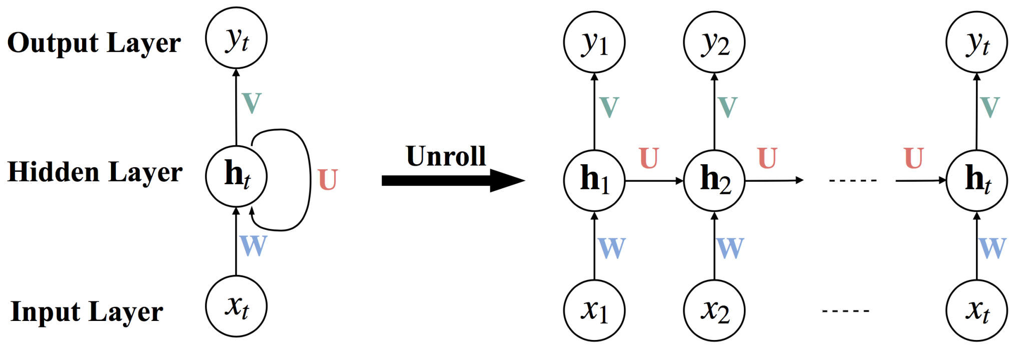


Figure 2.1: Abstract view of an RNN (left) and the same RNN, unrolled over time (right).

nal memory at time t. Let us now elaborate the details of each component.

The internal memory, (hidden) representation or (hidden) state $\mathbf{h}_t$, is usually kept as a real valued vector and contains a trace of recent events (how things are at the moment given the entire context). In other words, $\mathbf{h}_t$ summarizes the history from x_1 up to x_t. The initial state (or condition) $\mathbf{h}_0$ depends on the particular application, but in this Section, we assume the initial memory is set to zero.[4]

Before giving the details about the internals of RNN, let us define the the *computational graph* abstraction we employ in this thesis to represent mathematical expressions, and neural networks in particular. A computational graph is a directed acyclic graph, where mathematical operations, inputs and variables are represented by the nodes, and edges represent the flow of intermediate values and weights of the graph. The directed nature of a computational graph enforces the order of computation, and in neural network context, has two modes. The forward mode, (forward propagation or forward pass) computes the output of the computational graph give all it's inputs. The backward mode (back-propagation or backward pass), executes the flow of error signals computed at the output of the computational graph. An example computational graph can be seen in Fig. 2.1 right panel.

Graphically, an abstract RNN can be expressed as a computational graph, with an input layer, a hidden layer and an output layer, see Fig. 2.1 left panel. The difference between a feed-forward neural network and an RNN is the additional connections

[4] Further in the sequence-to-sequence models we will revisit the initial conditions, and explain how to give/learn contextual biases from other sequences.

between hidden layers at the same level (hidden-to-hidden connections). In Fig. 2.1 left, a compact representation of an RNN is given. In order to execute the mathematical expression represented by this RNN, we first *unroll* the computational graph over time, where all the input, hidden and output nodes are replicated to span all the inputs x_i to compute all the outputs y_i. An unrolled RNN is illustrated in Fig. 2.1 right panel.

In the core of an RNN, resides the parametric recursive function f. Given an initial state $\mathbf{h}_0$ and a sequence of input vectors $\mathbf{x} = (x_1, x_2, \dots, x_T)$, and the parameters θ, the recurrence relation of f can be expressed as follows:

$$\begin{aligned} \mathbf{h}_t &= f(x_t, \mathbf{h}_{t-1}; \theta) \\ &= f(x_t, f(x_{t-1}, \mathbf{h}_{t-2}); \theta) \\ &\dots \\ &= f(x_t, x_{t-1}, \dots, x_1, \mathbf{h}_0; \theta). \end{aligned} \tag{2.4}$$

As can be seen from the recurrence relation, the information is contained in the current memory (or hidden state) and each new input is processed in the context of the full history of the previous inputs. This simple connectionist network, trained properly, can learn statistical regularities over temporal sequences, in other words can *find structure in time* [38].

As mentioned above, learning in connectionist models is about adjusting the connection weights, namely, the parameters θ in our definition. The simplest parametrization of function f can be done with transformations of input x_t and hidden state $\mathbf{h}_{t-1}$ with parameters $\theta = \{\mathbf{W}, \mathbf{U}\}$ and an additional element-wise non-linear function σ, such as logistic sigmoid $\frac{1}{1+\mathrm{e}^{-z}}$ or hyperbolic-tangent $\frac{\mathrm{e}^{z}-\mathrm{e}^{-z}}{\mathrm{e}^{z}+\mathrm{e}^{-z}}$, as z being the input of the transfer function:

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}; \theta) = \sigma(\mathbf{W}x_t + \mathbf{U}\mathbf{h}_{t-1}), \tag{2.5}$$

where $\mathbf{W} \in \mathbb{R}^{d_h \times d}$ and $\mathbf{U} \in \mathbb{R}^{d_h \times d_h}$, as d_h is the dimensionality of the internal memory (hidden state) and d is dimensionality of the input. This parametrization is known as Elman Network [38] in the literature and usually called as vanilla-RNN. Please see Fig. 2.2 for a graphical depiction.

After having defined the internal memory $\mathbf{h}_t$ and the recursive function f, an RNN can also be equipped with an optional output function g, to predict the conditional probabilities in Eq. (2.2), parametrized with the weight matrix $\mathbf{V} \in \mathbb{R}^{d_h \times d_o}$:

$$y_t = p(x_t | x_{<t}) = g(\mathbf{h}_t; \mathbf{V}), \tag{2.6}$$

where d_o is the dimensionality of the output vector y_t. Note that, here we abuse the notation for the sake of generality and use y_t to refer $p(x_t|x_{<t})$, since the output y_t, as we will use shortly is not always a proper probability distribution. In order to make output y_t a proper distribution, function g is usually taken as a softmax activation function [11], after the transformation with $\mathbf{V}$ to predict the output probabilities y_t. We will revisit and detail the output function g in Language Modelling Section.

The most commonly used RNN achitectures are transducers (Elman networks) and encoders [45], which we also use extensively throughout this thesis. A transducer is a type of RNN, which produces an output y_i for each input x_i it reads, see Fig. 2.2. Transducers land in naturally to be used for language modelling [89], but also extensively employed for many other sequence labelling problems [48]. An encoder on the other hand, does not predict an output for each input x_i it reads, but processes the entire sequence one by one using Eq. (2.5) until the end of the sequence x_t, and the hidden state at position t, namely $\mathbf{h}_t$ is used as a summarization/encoding of the whole sequence $\mathbf{x}$. Encoding the temporal information of an arbitrary length sequence into a fixed-sized vector $\mathbf{h}_t$ is quite practical for abstractive summarization [107] and also being used as the encoders for multi-sequence modelling in this thesis.

Before going into the details of how we train an RNN using *backpropagation*, it is worth mentioning the computational powers of RNNs. We have the motivation to build generic multi-sequence modelling architectures and we want to be sure about the expressive power of underlying individual sequence models. RNNs are abstract machines, meaning that, for any computable function by a Turing machine, there

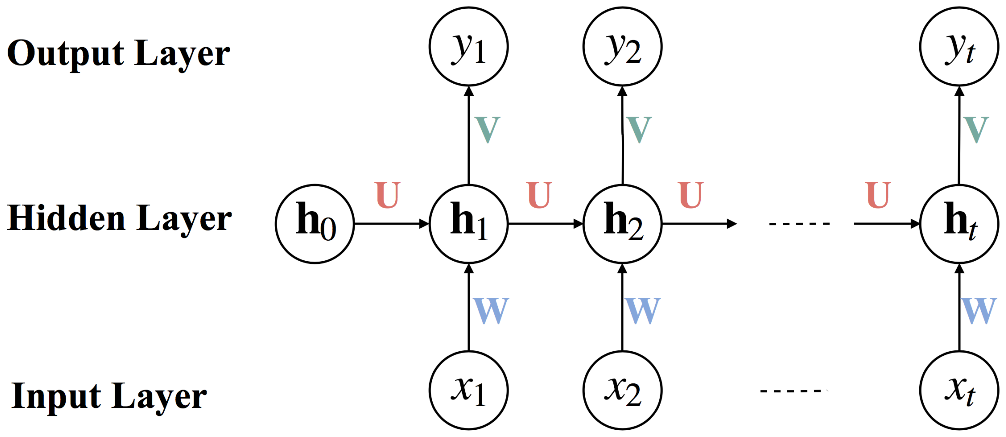


Figure 2.2: A Simple Elman-Recurrent Neural Network consists of three layers. From bottom-up, (1) Input layer, where discrete sequence of observations x_t are fed into the network with transformation matrix $\mathbf{W}$, one token per time step. (2) Hidden layer, where the function f recurses over each transformed input x_t and previous hidden state $\mathbf{h}_{t-1}$ to compute the hidden state $\mathbf{h}_t$ at time step t with the associated recurrent weight matrix $\mathbf{U}$. (3) Output layer, a sequence of outputs y_t, emitted one per each input observation x_t, with the transformation of hidden state h_t using output layer weight matrix $\mathbf{V}$. Notice that, the sequence of hidden states from $\mathbf{h}_0$ up to $\mathbf{h}_t$ traverses a trajectory in an intrinsic state space of RNN (also known as the *phase space* for dynamical systems [121]), and each hidden state $\mathbf{h}_i$ summarizes the previous context up to i.

exists a finite sized RNN that can compute it, making RNNs Turing complete. [115, 116, 114] Why do we care about the Turing completeness of RNNs that we use? Because we view each generated (naturally or artificially) sequence as a function and sequence modelling as a function approximation [21]. If we cannot approximate the underlying function of the sequence, the road to multi-sequence modelling will be ill conditioned. We should also remind a very important point that Expressivity $\neq$ Trainability, stated by Edward Grefesente. [5]

[5] `http://videolectures.net/deeplearning2016_grefenstette_augmented_rnn/`

Back Propagation Through Time

The common approach for training *connectionist* models is using stochastic gradient descent algorithm (SGD) [79] which relies on estimating the gradients of the loss function $\mathcal{L}$ with respect to the model parameters θ, formally $\nabla_\theta \mathcal{L}$. A videly used version of SGD, "mini-batch" SGD follows the steps below.[6]

Algorithm 1 "Mini-Batch" Stochastic Gradient Descent

```
**Given** starting point of $\theta$, loss function $\mathcal{L}$, learning rate $\eta$, dataset $\mathcal{D}$ and mini-batch size $m$
**repeat** until convergence or stopping criterion is satisfied
    1: *Sample.* Randomly choose $m$ examples from $\mathcal{D}$.
    2: *Compute.* Steepest descent direction $\nabla_\theta \mathcal{L}^{(i)}$, for each chosen example $i$.
    3: *Update.* $\theta_{new} := \theta - \eta \sum_{i=1}^{m} \nabla_\theta \mathcal{L}^{(i)}/m$
```

In this section, we focus on estimating the gradient of the loss function with respect to the model parameters $\nabla_\theta \mathcal{L}$, efficiently using back propagation through time (BPTT) algorithm [92, 104, 128], namely the Step 2 in Algorithm 1.

Let us first describe our loss function to be optimized. Given an input sequence $\mathbf{x} = (x_1, \ldots, x_T)$ where each x_i is a real valued vector, such that $x_i \in \mathbb{R}^d$ and a true label (ground truth) output sequence $\mathbf{y} = (y_1, \ldots, y_T)$ we want to train a transducer RNN, practically, optimize the model parameters $\theta = \{\mathbf{W}, \mathbf{U}, \mathbf{V}\}$ that minimizes the empirical loss $\mathcal{L}$ over the dataset $\mathcal{D} = \{\mathbf{x}^{(m)}, \mathbf{y}^{(m)}\}_{n=1}^{M}$ with N examples.[7]

As we can observe from Eq. (2.5) and Eq. (2.6), an RNN emits an output $\hat{y}_i$ at each time step i. We then use *local* prediction $\hat{y}_i$ and true label y_i to define local loss $\mathcal{L}_{local}(\hat{y}_i, y_i)$. Finally our total loss $\mathcal{L}$ is the sum of all local losses $\mathcal{L}_{local}$.[8] Note that, for a classification task, $\mathcal{L}_{local}$ can be defined as cross entropy loss or for regression $\mathcal{L}_{local}$ can be squared error. Let us pick the cross entropy loss [9] and equip function g in Eq. (2.6) with a softmax activation. [10] Putting it all together:

[6] Here we do not describe mini-batch (or vanilla)-SGD because of its commonality, but we will describe a particular variant of SGD which uses a slightly more sophisticated step rule, in the following section.

[7] Notice that, we threat a full sequence as one training example.

[8] We use $\mathcal{L}_t$ to refer $\mathcal{L}_{local}(\hat{y}_t, y_t)$ for brevity.

[9] Cross entropy loss between y_i and $\hat{y}_i$ is defined as $-y_i \log(\hat{y}_i)$

[10] A softmax activation is a normalized exponential function which can be interpreted as a proper probability distribution [9].

$$h_t = \sigma(\mathbf{W}x_t + \mathbf{U}h_{t-1}) \tag{2.7}$$

$$\hat{y}_t = \text{softmax}(\mathbf{V}h_t) \tag{2.8}$$

$$\mathcal{L}_{local}(\hat{y}_t, y_t) = -y_t \log(\hat{y}_t) \tag{2.9}$$

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_t \mathcal{L}_{local}(\hat{y}_t, y_t) = -\sum_t y_t \log(\hat{y}_t) \tag{2.10}$$

Given the total loss, $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ our goal is to calculate the gradients of this loss with respect to the model parameters θ, formally, $\nabla_\theta \mathcal{L} = \{\frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \frac{\partial \mathcal{L}}{\partial \mathbf{U}}, \frac{\partial \mathcal{L}}{\partial \mathbf{V}}\}$. We sum up the local losses at each time step t, we also sum up (accumulate) all the gradients at each time step for a training example, such that $\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = \sum_t \frac{\partial \mathcal{L}_t}{\partial \mathbf{V}}$.

Similar to thestandard backpropagation algorithm, BPTT also relies on chain rule and reverse mode differentiation, and in practice, BPTT is an extension of standard backpropagation applied to unrolled computational graphs, shown in Fig. 2.2. Starting from the local loss $\mathcal{L}_t$ at time step t, let us calculate all the gradients. For $\frac{\partial \mathcal{L}_t}{\partial \mathbf{V}}$, the gradient of the local loss function $\mathcal{L}_t$ with respect to output layer parameter matrix $\mathbf{V}$, we simply apply the chain rule:

$$\begin{aligned}\frac{\partial \mathcal{L}_t}{\partial \mathbf{V}} &= \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial \mathbf{V}} \\ &= \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial z_t}{\partial \mathbf{V}},\end{aligned}$$

where z_t is the softmax pre-activation in Eq. (2.6), namely, $z_t = \mathbf{V}\mathbf{h}_t$. It is easy to see that the gradient $\frac{\partial \mathcal{L}_t}{\partial \mathbf{V}}$ depends only on the current time step variables, $\hat{y}_t, y_t, z_t$, thus easy to calculate.

The calculation gets slightly more tedious for the gradients $\frac{\partial \mathcal{L}_t}{\partial \mathbf{U}}$ and $\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}}$, because of the recurrence. For the recurrent weight matrix $\mathbf{U}$, corresponding gradient is calculated as:

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{U}} = \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{U}}.$$

The crucial point in the above equation, the hidden state $\mathbf{h}_t$ depends on $\mathbf{h}_{t-1}$ and $\mathbf{U}$, because of the recurrence in Eq. (2.5). Since we use $\mathbf{U}$ at each time step up to t, we have to backpropagate gradients from t through the network all the way back to $t = 0$, which yields:

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{U}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{U}}. \tag{2.11}$$

In Eq. (2.11), it is worth mentioning that the term $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}$ itself expands with the chain rule, giving our final gradient in the form:

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{U}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial \mathbf{h}_t} \left(\prod_{j=k+1}^{t} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} \right) \frac{\partial \mathbf{h}_k}{\partial \mathbf{U}}. \tag{2.12}$$

The gradient $\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}}$ is very similar to Eq. (2.12) therefore we skip the derivations for $\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}}$. Luckily, all the modern neural network libraries [120, 1] provide efficient automatic differentiation support that prevents us to manually implement the above gradients for BPTT.

Gated Recurrent Units

Training recurrent neural networks using BPTT suffers from two major problems. *Vanishing* and *exploding* gradients [56, 8]. These problems can prevent the model to capture long-term dependencies severely, sometimes even halts the training completely. Let us take a look at the term $\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}$ in Eq. (2.12), which appears to be a Jacobian matrix.[11] In BPTT, we multiply this Jacobian matrix over and over again along the longest credit assignment path. It has been shown that [101], when the 2-norm of this Jacobian is greater than 1, gradients tend to explode and exponentially vanish when 2- norm of this Jacobian is smaller than 1. [12]

Dealing with the gradient vanishing problem is still an open research question, but here we mention one particular remedy, a specialized hidden unit that is designed to assist the gradient flow back in time, called Gated Recurrent Units (GRU) [24, 29].

[11] Computed by the derivative of a vector function with respect to a vector.
[12] 2-norm of a matrix $\|\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}\|$ corresponds to its largest eigenvalue.

The main idea behind GRU is creating backward shortcut connections across non-consecutive time steps (units that are not connected directly). During BPTT, these shortcut connections will help the error signals skip some connections if *necessary*, practically preserving gradients across time. Important question is that, how do we decide this *necessity*. GRU employs internal parametric functions to decide when to carry the information or when to stop. Additional parametric functions (gates) are trained along with the model itself using BPTT. Gating functions are implemented as simple feed-forward neural networks but their outputs are always maintained to be between 0-1 (hopefully close to either one of them), casting them as soft multiplicative gates.

GRU introduces an additive leaky integration function, and two internal gates, namely, reset (r) and update (u) gates, with and internal hidden state of dimensionality d_h, $r \in [0;1]^{d_h}$, $u \in [0;1]^{d_h}$. Formally, GRU replaces Eq. (2.5):

$$\mathbf{h}_t = u \odot \mathbf{h}_{t-1} + (1-u) \odot \tilde{\mathbf{h}}_t, \tag{2.13}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}x_t + r \odot (\mathbf{U}\mathbf{h}_{t-1})\right), \tag{2.14}$$

$$r = \sigma\left(\mathbf{W}_r x_t + \mathbf{U}_r \mathbf{h}_{t-1}\right), \tag{2.15}$$

$$u = \sigma\left(\mathbf{W}_u x_t + \mathbf{U}_u \mathbf{h}_{t-1}\right). \tag{2.16}$$

GRU first computes a proposal hidden state $\tilde{\mathbf{h}}_t$ for time step t, by looking at current input x_t and also previous hidden state $\mathbf{h}_{t-1}$, see Eq (2.14). However, the difference with the standard recurrent unit in Eq. (2.5) is that, GRU gates the information coming from the previous hidden state $\mathbf{h}_{t-1}$ with the reset gate r, see Eq. (2.15), practically applying an element-wise multiplication (indicated by $\odot$).[13]. The proposal hidden state $\tilde{h}_t$ is, then, linearly interpolated with the previous hidden state h_{t-1}. This interpolation is also gated with another gate, called the update gate u, effectively to control the ratio of a blend between proposal hidden state $\tilde{h}_t$ and previous hidden state h_{t-1} (see Eq. (2.13) and (2.16)). With the help of update and reset gates, GRU can learn to keep it's internal hidden state intact, effectively creating shortcuts through time.

Another well known gated recurrent unit is Long Short-Term Memory (LSTM) which highly influenced the design of GRU and also commonly used in the literature [57,

[13] Also known as an hadamard product.

49]. Here, we do not detail LSTM, since they share the basic principles with GRU, (ie. activations close to 1 for u).

The exploding gradient problem on the other hand, has a very simple and empirical solution. Whenever the norm of the gradient $\nabla \mathcal{L}_\theta$ exceeds a threshold τ, we renormalize the norm of the gradient to be τ:

$$\nabla_\theta \mathcal{L} = \begin{cases} \tau \frac{\nabla_\theta \mathcal{L}}{\|\nabla_\theta \mathcal{L}\|}, & \text{if } \|\nabla_\theta \mathcal{L}\| > \tau \\ \nabla_\theta \mathcal{L}, & \text{otherwise} \end{cases} \tag{2.17}$$

This technique is called *gradient clipping* [101] and extensively is used in the literature.

2.1.2 Neural (Connectionist) Language Modelling

A language model, is a probability distribution function over the sequences of words. On the other hand, language modelling is about learning a conditional probability distribution over the words given a *context* [4], as we have also framed in Eq. (2.1)-(2.2). In traditional non- parametric models, this conditional probability is approximated with count based statistics, such as (smoothed) N-gram models [71, 75, 47]. Counterpart parametric approaches include maximum entropy models [105], feed-forward neural networks [6] and recurrent neural networks [89]. As stated in Section 2.1.1, at this thesis, we are only interested in parametric approaches using RNN, since they surpass the previous approaches by large margins with the state-of-the-art results [68].

Formally speaking, let V be the *vocabulary* which contains all possible tokens (symbols, words, sub-words or characters) in a language, with a total number of unique symbols $|V|$,

$$V = \{w_1, w_2, \ldots, w_{|V|}\}. \tag{2.18}$$

Further, let each symbol w_i in V is represented with a one-hot encoding such that, $w_i \in \{0,1\}^{|V|}$ is a binary vector of dimensionality $|V|$, having only one non-zero

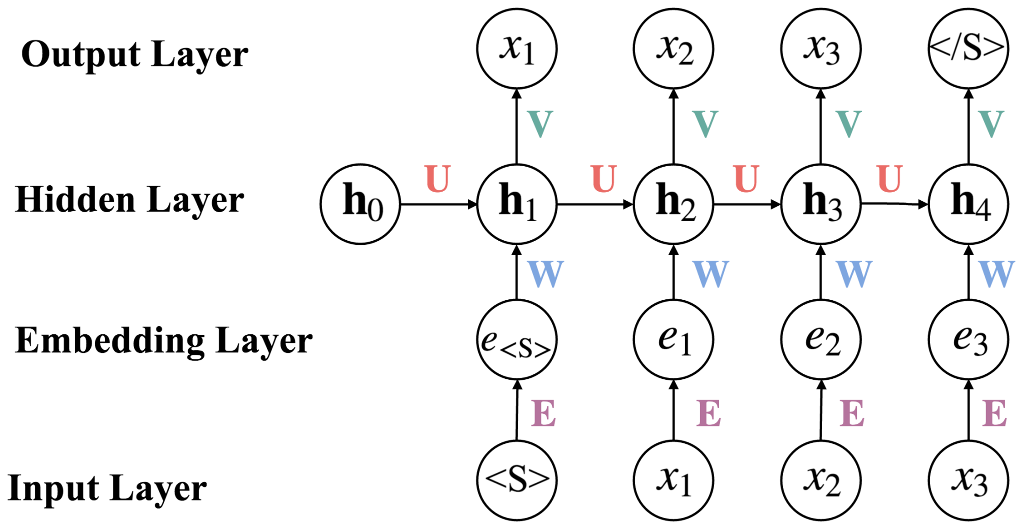


Figure 2.3: A Recurrent Neural Network Language Model (RNN-LM) over the sequence x_1, x_2, x_3. Similar to a Recurrent Neural Network, an RNN-LM consists of three regular layers and one additional embedding layer. From bottom to top, input layer represents the input sequence, embedding layer corresponds to the mapping (embedding) of a one-hot encoded input token to a continuous space, followed by a hidden layer and output layer. Number of hidden layers, can be more than one, simply by stacking more layers. As the task of a language model is to predict the next token, input sequence is presented to the RNN-LM with a special token <s> at the beginning, indicating the beginning of sequence. Similarly, the output sequence is appended with an end of sequence token </s>, enforcing the model to predict the end of sequence token when it sees the last input token x_3

.

element at index i. [14] One artifact of such encoding is the orthogonality of items in the vocabulary. As an example, any two symbols in the vocabulary will be linearly independent even if they are semantically close (eg. *king* and *queen*). Neural language models, introduce the concept of *continuous space word representations* [6], which maps (embeds) each one-hot encoded discrete symbol into a continuous space that preserves the original semantic relationships across symbols.

Continuous space representations make use of an embedding matrix $\mathbf{E} \in \mathbb{R}^{|V| \times d_{emb}}$ where d_{emb} is the dimensionality of embedding space. By slicing the rows of matrix

[14] Notice that, we change the notation from x_i to w_i just to emphasize the discrete nature of inputs and outputs. Since we do not want to break the generality of an input sequence (which can very well be a sequence of patches in an image) we can use w_i when the inputs of the model are discrete in place of x_i.

$\mathbf{E}$ with the corresponding symbol w_i, we obtain a continuous vector representation of symbol w_i which we call e_i, simply $e_i = \mathbf{E}[w_i]$. In neural language modelling, this representation is called word embedding and used to embed discrete input tokens into a continuous space [6, 90]. As this slicing operation can be tought as a linear operation and differentiable, we can train the parameters of the embedding matrix $\mathbf{E}$ along with all the other parameters of the model at use. In short, embedding matrices are learned together with all the other parameters using backpropagation. Please see Fig. 2.3 for an illustration of input and embedding layers. We refer the reader to the references [6, 33, 90] for further explanations of word embeddings, since word embeddings are not the main focus of this thesis. Next we detail the internals of RNN language model and it's variants.

Recurrent Language Model

In a RNN language model (RNN-LM), the goal is to predict the next token (symbol) given the entire preceding symbols (context). For an input sequence $\mathbf{x} = (x_1, x_2, x_3)$ an RNN-LM first constructs it's input and output sequences by making use of two special symbols, the beginning of sequence symbol <s>, and the end of sequence symbol </s>. Both symbols are necessarily added to the vocabulary V. Since the goal is to predict the next symbol, an input-output pair $\mathbf{x}^{in}, \mathbf{x}^{out}$ can easily be constructed by prepending <s> to the input sequence $\mathbf{x}^{in}$, and appending </s> to the output sequence $\mathbf{x}^{out}$, such that $\mathbf{x}^{in} = (\text{<s>}, x_1, x_2, x_3)$ and $\mathbf{x}^{out} = (x_1, x_2, x_3, \text{</s>})$. By doing this pre-ordering of input and output sequences, we practically shift the input sequence to right by one token, aligning the input-output mapping each time step to be consistent with predicting the next symbol in the sequence. Please see Fig. 2.3 for the graphical illustration of an RNN-LM. In an RNN-LM layout, the symbol emitted at time step t is given as an input to time step $t + 1$ as the model is generating an output sequence given an initial input token x_1 at test time which is also called as sampling. During training, as we know the whole input sequence and the corresponding output sequence (which simply is the shifted version of the input sequence) the true labels are given to the model for each time step which is known as teacher forcing in the literature [129].

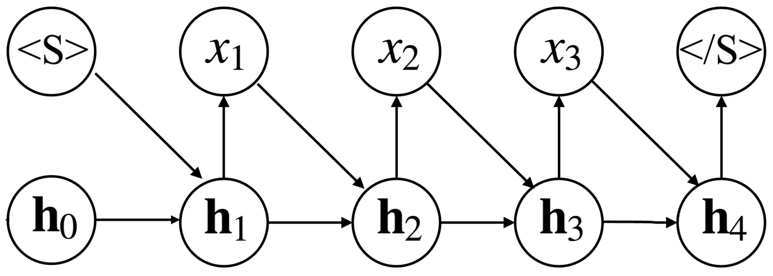


Figure 2.4: Simplified illustration of an RNN Language Model (RNN-LM). All the connection weights and embedding layer is omitted for clarity. Further input and output layers merged together to emphasize the computations done by the RNN-LM.

The next question is, how are the discrete symbols in the vocabulary V are being used in a RNN-LM in order to map an input sequence to an output sequence. Let, $\mathbf{E}\left[x_t\right]$ be the continuous space vector representation of the t-th symbol in the input sequence, $\Psi(\cdot)$ be the recurrent activation function (eg. tanh, GRU or LSTM unit) and g_j be the j-th element of the output function $g : \mathbb{R}^{d_h} \rightarrow \mathbb{R}^{|V|}$. An RNN-LM predicts the probability of an output symbol j at time step t, by following:

$$\mathbf{h}_t = \Psi(\mathbf{h}_{t-1}, \mathbf{E}\left[x_t\right]), \tag{2.19}$$

$$p(x_t = j | x_{<t}) = \frac{\exp(g_j(\mathbf{h}_t))}{\sum_{j'=1}^{|V|} \exp(g_{j'}(\mathbf{h}_t))}, \tag{2.20}$$

where the right hand side of Eq. (2.20) is called softmax. Considering the recurrent function Ψ being a vanilla RNN, such that $\Psi = f(x_t, \mathbf{h}_{t-1})$ from Eq. (2.5), the parameter set of an RNN-LM is $\theta = \{\mathbf{E}, \mathbf{W}, \mathbf{U}, \mathbf{V}\}$.

Training an RNN-LM is simply done by maximizing likelihood of the training set $\mathcal{D}$. Given a training set of M examples (sequences), such that $\mathcal{D} = \{(x_1^{(1)}, \dots, x_{T_1}^{(1)}), \dots, (x_1^{(M)}, \dots, x_{T_M}^{(M)}$ with the loss function $\mathcal{L}(\theta)$, training objective is to maximize the log-probability of $\mathcal{D}$. As it was done in Section 2.1.1, we can re- write the joint log-probability of a sequence as the sum of conditionals,

$$\log p(x_1, \dots, x_T) = \sum_{t=1}^{T} \log p(x_t | x_1, \dots, x_{i-1}). \tag{2.21}$$

Then, the loss function to optimize the parameters θ is:

$$\mathcal{L}(\mathcal{D};\theta) = \arg\max_{\theta} \frac{1}{M}\sum_{m=1}^{M} \log p(x_1^{(m)},\dots,x_{T_m}^{(m)}), \tag{2.22}$$

$$= \arg\min_{\theta} -\frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T_m} \log p(x_t^{(m)}|x_{<t}^{(m)}). \tag{2.23}$$

Learning is done, by mini-batch stochastic gradient descent (SGD). Note that, in figures and drawings, we omit the embedding layer, along with the weight matrices, for the sake of brevity. For a simplified graphical depiction of an RNN-LM, see Fig. 2.4.

Conditional Recurrent Language Model

A conditional RNN-LM (cRNN-LM) is a regular RNN-LM with auxiliary inputs, such that, the distribution over the sequence is conditioned on an additional source of information. The additional information, which we call $\mathcal{C}$, can be any other modality, such as a sequence in another language (machine translation) [24, 118, 3], a sequence of frames in video (video captioning)[134], patches in an image (image captioning) [133], a speech signal (speech translation)[25, 19] or sometimes multiple modalities, such as image and speech or image and text [30, 13]. In all of the above cases, cRNN-LM slightly changes the formulation of RNN-LM to account for the newly introduced modality.

Let $\mathcal{C}$ be the auxiliary information source, which is a set of T_c number of vectors, $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^{T_c}$, where each $\mathbf{c}_i$ is a partial representation of the auxiliary input modality. The cardinality of $\mathcal{C}$, $T_c = |\mathcal{C}|$ is assumed to be arbitrary. Further, let Φ be the time-step sensitive encoding function of the new modality for $\mathcal{C}$, such that,

$$\tilde{\mathbf{c}}_t = \Phi(\mathcal{C}, t). \tag{2.24}$$

We incorporate Φ in the recurrent activation function Ψ of Eq. (2.19) simply as an additional input to the recurrent function:

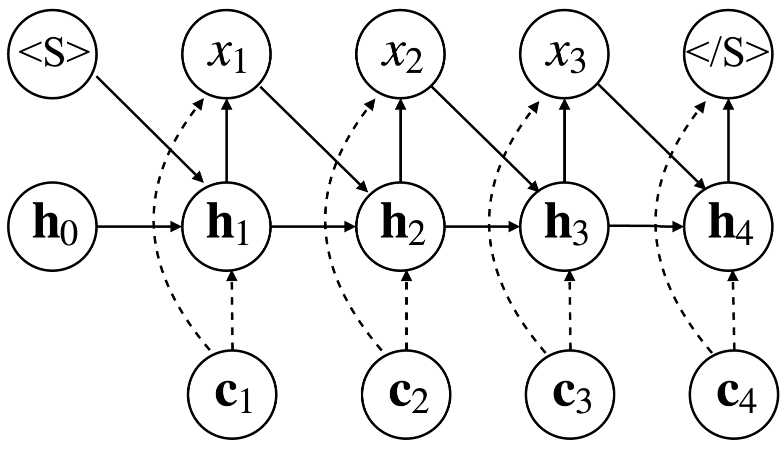


Figure 2.5: A conditional recurrent language model (cRNN-LM). Dotted connections are additional connections on top of a regular RNN-LM model. At each time step, a cRNN-LM looks at its input x_t, previous hidden state $\mathbf{h}_{t-1}$ and additional source of information $\mathbf{c}_t$ via modality encoding function Φ. Note that, encoding function Φ can be time-step insensitive, effectively $\{\forall i, j | \Phi(\mathcal{C}, i) = \Phi(\mathcal{C}, j)\}$.

$$\mathbf{h}_t = \Psi(\mathbf{h}_{t-1}, \mathbf{E}\left[x_t\right], \Phi(\mathcal{C}, t)), \tag{2.25}$$

It is common to take into account (condition on) the auxiliary information also in the output function g. Then, the conditional probability of the j-th symbol at time step t becomes:

$$p(x_t = j | x_{<t}, \mathcal{C}) = \frac{\exp(g_j(\mathbf{h}_t, \tilde{\mathbf{c}}_t))}{\sum_{j'=1}^{|V|} \exp(g_{j'}(\mathbf{h}_t, \tilde{\mathbf{c}}_t))}. \tag{2.26}$$

Training is executed similar to RNN-LM, however the dataset $\mathcal{D}$ consists of $(\mathbf{x}, \mathcal{C})$ pairs, $\mathcal{D} = \{(\mathbf{x}^{(m)}, \mathcal{C}^{(m)})\}_{m=1}^{M}$ and cRNN-LM is again trained to minimize the negative log-likelihood of the training set $\mathcal{D}$, which is defined as:

$$\mathcal{L}(\mathcal{D}; \theta) = \arg\min_{\theta} -\frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T_m} \log p(x_t^{(m)} | x_{<t}^{(m)}, \mathcal{C}^{(m)}). \tag{2.27}$$

See Figure 2.5 for an illustration of cRNN-LM.

A natural use-case of cRNN-LM is combination with another RNN-LM. The latter

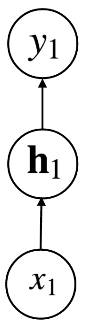
RNN-LM in this case, is responsible for generating the auxiliary information source, namely the set of representations $\mathcal{C}$. This specific tying of a cRNN-LM and a RNN-LM constructs the basis of the sequence to sequence models which we will explain next. The important distinction across the sequence to sequence models is the definition and usage of representation set $\mathcal{C}$ and mapping function Φ, which we explain in the next section.

2.2 Sequence to Sequence Models - seq2seq

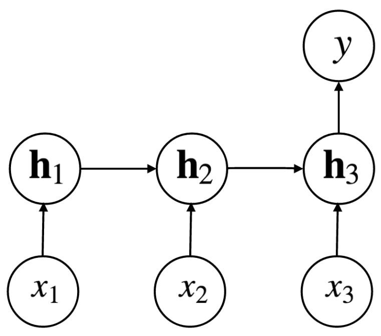
Sequence to sequence models aim at building a single neural network that takes input sequence $X = (x_1, \ldots, x_{T_x})$ of length T_x and generates the corresponding output (translation) sequence $Y = \left(y_1, \ldots, y_{T_y}\right)$ of length T_y. Then, sequence to sequence models are one of the most generic frameworks in machine learning that we can use to map any input sequence to any output sequence.

Let us illustrate some architectures from the sequence to sequence modeling perspective. The simplest example is encountered when input and output sequence lengths are equal to one, in other words, $T_x = T_y = 1$, which is basically a feed-forward neural network for classification/regression, Fig. 2.6a. Next group of problems involve many-to-one mapping, (Fig. 2.6b) when $T_x > 1$ and $T_y = 1$, as in sentiment analysis [123]. Similarly one-to-many mapping problems , (Fig. 2.6c), with $T_x = 1, T_y > 1$, as in image captioning [133]. When both source and target sequences lengths are larger than one, problems get more complicated. These problems can be handled within the seq2seq framework. One particular set of problems is sequence labeling [96], where $T_x = T_y > 1$, Fig. 2.6d. Finally, the most generic problems are formalized, when both source and target sequence lengths are larger than one and not necessarily equal in length with each other, Fig. 2.6e, $T_x > 1, T_y > 1$, as in neural machine translation [24, 118].

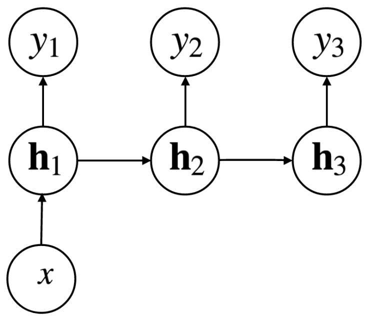
Surprisingly, this end-to-end approach was independently discovered multiple times withing the last 20 years [41, 16, 70, 24, 118]. However, they became popular after the resurgence of modern neural networks [46]. In principal, all seq2seq models rely on, first encoding an input sequence into an arbitrary length representation ($\mathcal{C}$), and then



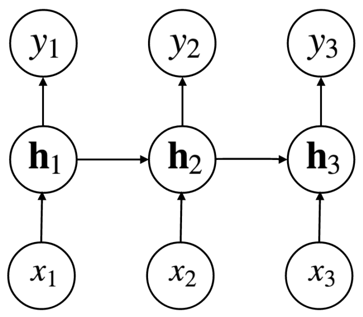
(a) One-to-one



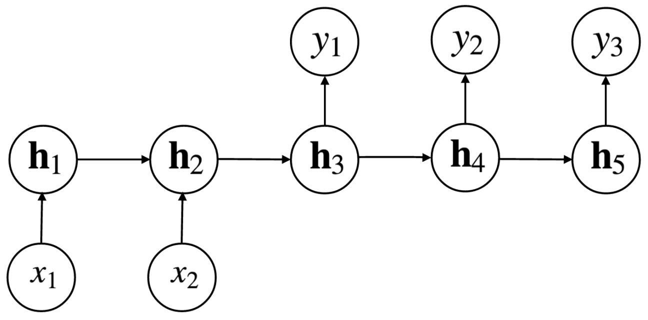
(b) Many-to-one



(c) One-to-many



(d) Many-to-many


y_1 y_2 y_3
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ $\mathbf{h}_5$
x_1 x_2

(e) Sequence-to-sequence

Figure 2.6: Sequence-to-sequence models with varying source and target sequence lengths T_x and T_y.

generating an output sequence by decoding this intermediate representation. Next, we detail two common approaches of seq2seq, (1) basic encoder-decoder model and (2) encoder-decoder model with attention.

2.2.1 Basic Encoder-Decoder Model

A basic encoder-decoder network consists of two recurrent networks. The first network, called *encoder*, maps x, an input sequence of variable length, into a point in

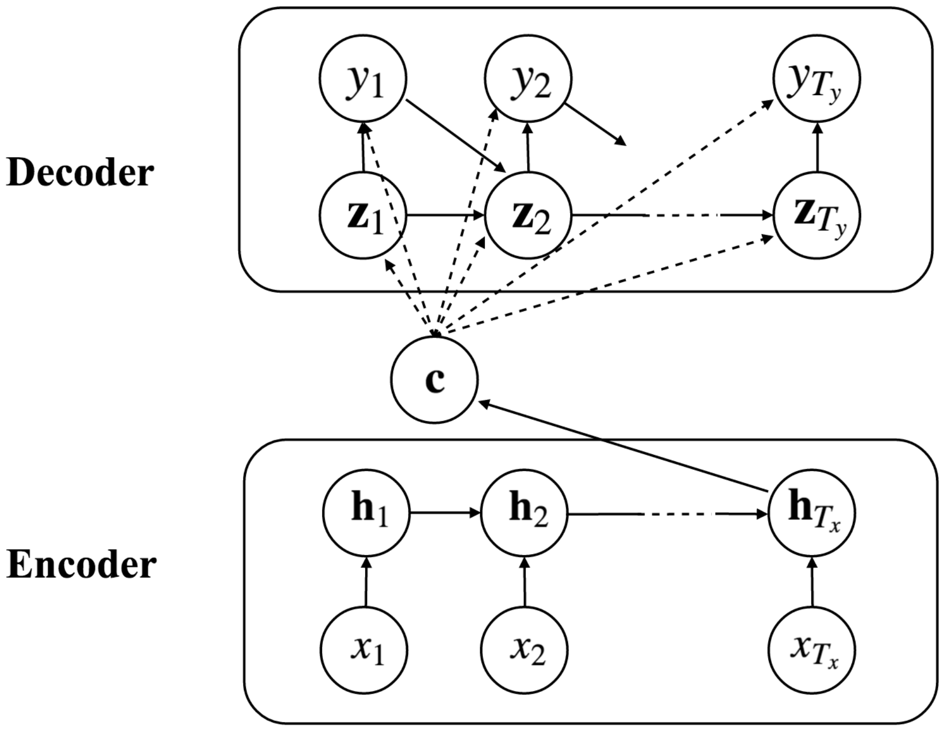


Figure 2.7: Simple sequence to sequence model, Encoder-Decoder architecture.

a continuous vector space, resulting in a fixed-dimensional context vector $\mathbf{c}$. Then, the second recurrent neural network, called *decoder*, generates a target sequence $\mathbf{y}$, again of variable length, starting from the context vector $\mathbf{c}$, see Fig. 2.7. Although we introduce the context vector $\mathbf{c}$ as an additional variable, it is considered as the last hidden state of the encoder RNN, $\mathbf{c} = \mathbf{h}_{T_x}$, as explained in Section 2.1.1.

Following the descriptions in the previous section, a basic encoder-decoder network can be constructed, simply by stitching an RNN-LM with a cRNN-LM. Extending Eq. (2.24)-(2.26), with $\mathcal{C} = \{\mathbf{c}\}$ and making $\Phi(\cdot)$ time-step insensitive identity mapping, we obtain a simple encoder-decoder model:

$$\mathbf{h}_{T_x} = \Psi_{enc}(\mathbf{h}_{T_x-1}, \mathbf{E}_x[x_t]), \tag{2.28}$$

$$\mathbf{c} = \Phi(\mathcal{C}) = \mathbf{h}_{T_x}, \tag{2.29}$$

$$\mathbf{z}_t = \Psi_{dec}(\mathbf{z}_{t-1}, \mathbf{E}_y[y_t], \mathbf{c}). \tag{2.30}$$

In the above equations, encoder Ψ_{enc} is implemented as an RNN-LM and decoder Ψ_{dec} is implemented as a cRNN-LM. Having each symbol in both source and target

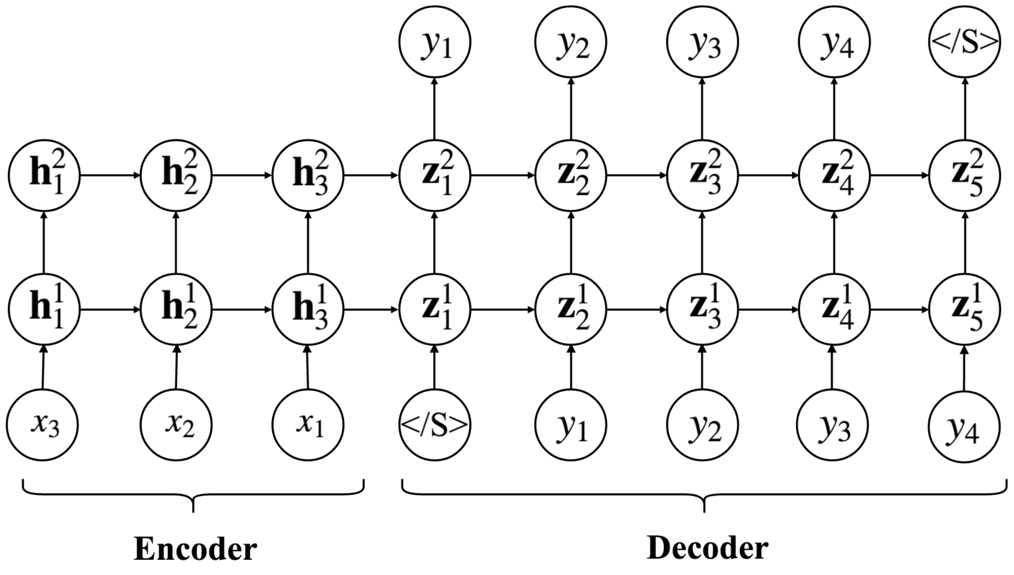


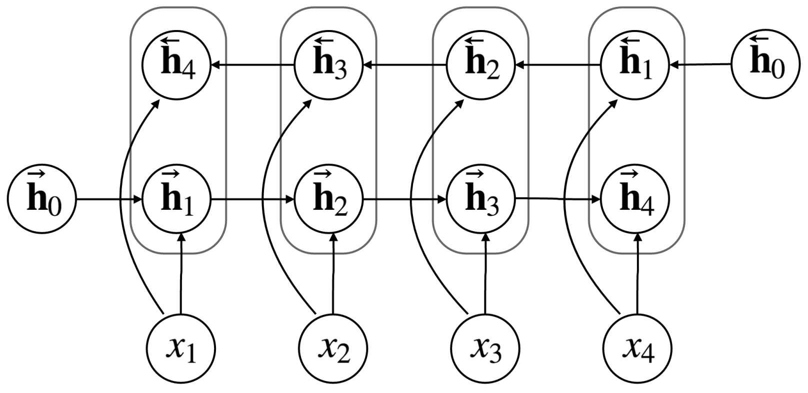
Figure 2.8: A simple Encoder-Decoder architecture (without attention) with input sequence reversing and stacking recurrent layers.

sequences, x_t or y_t, (an integer index of the symbol in a vocabulary as described in Section 2.1.2), we use $\mathbf{E}_x$ and $\mathbf{E}_y$ to refer input and output embedding matrices respectively.

One last issue is about setting the initial hidden state of the decoder, $\mathbf{z}_0$. From a trajectory learning perspective described previously, decoder RNN should follow a trajectory that generates the output sequence $\mathbf{y}$, by traversing the state space from $\mathbf{z}_0$ to $\mathbf{z}_{T_y}$. Tracing back this trajectory all the way to $\mathbf{z}_0$, we can see that, in order to start generating an output sequence that is relevant to the input sequence, the initial position in the decoder state space $\mathbf{z}_0$, should be put into the right spot, practically should use the right bias coming from the encoder. Luckily, we can make use of the last hidden state of the encoder $\mathbf{h}_{T_x}$, which corresponds to the end point of the encoded trajectory, to give the right bias to the decoder trajectory. This can be achieved simply by setting $\mathbf{z}_0 = \mathbf{c}$, but also, having a parametric function that initializes $\mathbf{z}_0$:

$$\mathbf{z}_0 = f_{init}(\mathbf{c}). \tag{2.31}$$

Please see Fig. 2.7 for a graphical illustration of the basic encoder-decoder architecture.



$\overleftarrow{\mathbf{h}}_4$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_0$
$\overrightarrow{\mathbf{h}}_0$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
x_1
x_2
x_3
x_4

Figure 2.9: Bidirectional encoder, with two RNN. The final hidden state $\mathbf{h}_i$ is concatenation of forward and backward RNN hidden states at time point i, represented by the vertical boxes at each time-step.

The basic encoder-decoder approach however has been reported to be inefficient in [23] for handling long sequences, due to the difficulty in learning a complex mapping between an arbitrary long sentence and a single fixed-dimensional vector. An early attempt to improve the basic encoder-decoder architecture makes two important modifications [118]. First, reversing the input sequence to align the beginning of source and target trajectories closely. Second, stacking more recurrent layers to increase the capacity/complexity of the seq2seq architecture. With these two enhancements, basic encoder-decoder architecture achieved state-of-the-art performance on challenging large scale tasks, such as machine translation. Please see Fig. 2.8 for a graphical illustration. Next, we describe a technique to improve the basic encoder-decoder model by extending function $\Phi(\mathbf{y})$ to a time-step sensitive function such that, $\Phi(\mathbf{y}, t)$.

2.2.2 Encoder-Decoder Model with Attention

The attention-based encoder-decoder model was first proposed in [3]. It was motivated from the observation in [23] that a basic encoder-decoder (translation) model, described in the previous section, [24, 118] suffers from translating a long source sequence into a target sequence, efficiently. This is mainly due to the fact that the encoder of this basic approach needs to compress (encode) a whole source sequence into a single vector. This vector representation creates an information bottleneck between encoder and decoder, during the end-to-end optimization of cost function of the

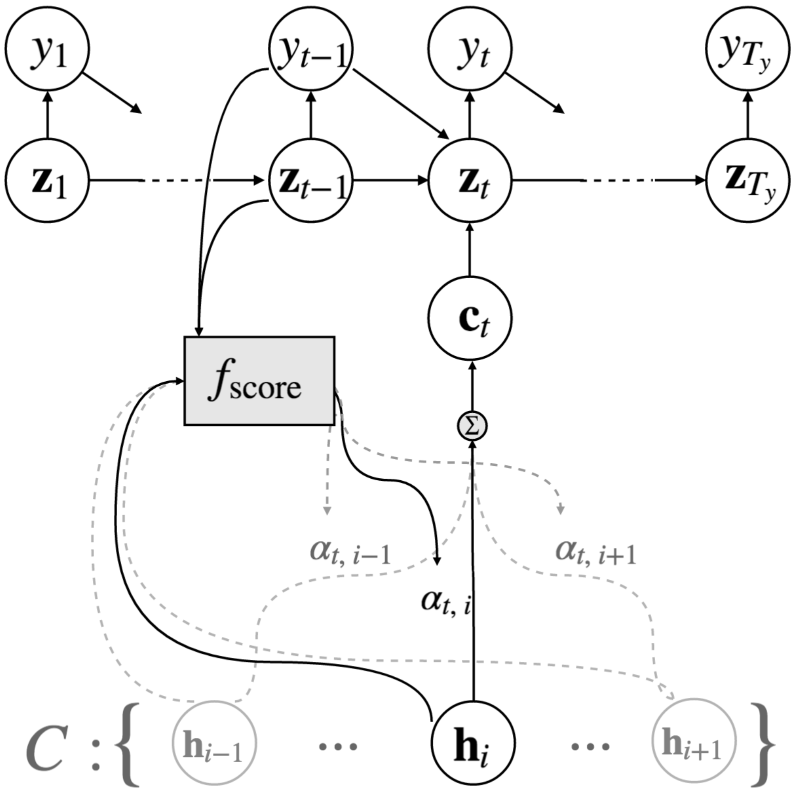


Figure 2.10: Representation of a decoder time-step t for an attention based encoder-decoder architecture. Source sequence is first annotated with context set $\mathcal{C}$. For time step t of the decoder, attention module (f_{score}) generates attention scores $\alpha_{t,i}$ for each annotation i in the context set $\mathcal{C}$ and then the weighted average of context set $\mathbf{c}_t$ is fed to the decoder (cRNN-LM) for time step t.

network, especially when it is deep. Here we describe the attention-based encoder-decoder model in detail that remedies above issues.

The encoder of the attention-based model encodes a source sequence into a set of context vectors $\mathcal{C} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}\}$, whose size varies with respect to the length of the source sequence. This context set is constructed by a bidirectional recurrent neural network (RNN) [109], which consists of a forward RNN and reverse RNN, see Fig. 2.9. The forward RNN reads the source sequence from the first token until the last one, resulting in the forward context vectors $\left\{\overrightarrow{\mathbf{h}}_1, \ldots, \overrightarrow{\mathbf{h}}_{T_x}\right\}$, where

$$\overrightarrow{\mathbf{h}}_t = \overrightarrow{\Psi}_{\text{enc}}\left(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{E}_x\left[x_t\right]\right),$$

and $\mathbf{E}_x \in \mathbb{R}^{|V_x| \times d}$ is an embedding matrix containing row vectors of the source symbols. The reverse RNN in an opposite direction, resulting in $\left\{\overleftarrow{\mathbf{h}}_1, \ldots, \overleftarrow{\mathbf{h}}_{T_x}\right\}$, where

$$\overleftarrow{\mathbf{h}}_t = \overleftarrow{\Psi}_{\text{enc}}\left(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{E}_x\left[x_t\right]\right).$$

$\overrightarrow{\Psi}_{\text{enc}}$ and $\overleftarrow{\Psi}_{\text{enc}}$ are recurrent activation functions such as long short-term memory units (LSTM, [57]) or gated recurrent units (GRU, [24]). At each position in the source sentence, the forward and reverse context vectors are concatenated to form a full context vector, i.e.,

$$\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t\right]. \tag{2.32}$$

The decoder, which is implemented as an cRNN, generates one symbol at a time, the translation of the source sequence, based on the context set $\mathcal{C}$ returned by the encoder. At each time step t in the decoder, a time- dependent context vector $\mathbf{c}_t$ is computed based on the previous hidden state of the decoder $\mathbf{z}_{t-1}$, the previously decoded symbol $\tilde{y}_{t-1}$ and the whole context set $\mathcal{C}$. This starts by computing the relevance score of each context vector as

$$e_{t,i} = f_{\text{score}}(\mathbf{h}_i, \mathbf{z}_{t-1}, \mathbf{E}_y\left[\tilde{y}_{t-1}\right]), \tag{2.33}$$

for all $i = 1, \ldots, T_x$. f$_{\text{score}}$ can be implemented in various ways [86]. In general, we can use a simple single- layer feedforward network. This relevance score measures how relevant the i- th context vector of the source sequence is in deciding the next symbol in the translation. The relevance scores are further normalized:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T_x}\exp(e_{t,j})}, \tag{2.34}$$

and we call $\alpha_{t,i}$ the attention weight. Note that, each attention weight $\alpha_{t,i}$ is between $[0-1]$ letting us to interpret them as probabilities.

The time-dependent context vector $\mathbf{c}_t$ is then defined as the weighted sum of the context vectors with their weights being the attention weights from above:

$$\mathbf{c}_t = \sum_{i=1}^{T_x}\alpha_{t,i}\mathbf{h}_i. \tag{2.35}$$

With this time-dependent context vector $\mathbf{c}_t$, the previous hidden state $\mathbf{z}_{t-1}$ and the previously decoded symbol $\tilde{y}_{t-1}$, the decoder's hidden state is updated by

$$\mathbf{z}_t = \Psi_{\text{dec}} \left(\mathbf{z}_{t-1}, \mathbf{E}_y \left[\tilde{y}_{t-1} \right], \mathbf{c}_t \right), \tag{2.36}$$

where Ψ_{dec} is a recurrent activation function.

The initial hidden state $\mathbf{z}_0$ of the decoder is initialized based on the last hidden state of the reverse RNN:

$$\mathbf{z}_0 = f_{\text{init}} \left(\overleftarrow{\mathbf{h}}_{T_x} \right), \tag{2.37}$$

where f_{init} is a feedforward network with one or two hidden layers.

The probability distribution for the next target symbol is computed by

$$p(y_t = k | \tilde{y}_{<t}, X) \propto e^{g_k(\mathbf{z}_t, \mathbf{c}_t, \mathbf{E}[\tilde{y}_{t-1}])}, \tag{2.38}$$

where g_k is a parametric function that returns the unnormalized probability for the next target symbol being k. In summary, an encoder-decoder model with attention alternates between three phases at each time-step of the decoding process. (1) look, where attention module attends to the context set, (2) update where decoder cRNN-LM updates its internal hidden state $\mathbf{z}_t$ with the provided information and at last (3) generate, where decoder generates the output token y_t for time-step t. These phases are further summarized in Table. 2.1.

Training this attention-based model is done again by maximizing the conditional log-likelihood of the training set $\mathcal{D} = \{\mathbf{x}^{(m)}, \mathbf{y}^{(m)}\}_{m=1}^{M}$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{n=1}^{N} \sum_{t=1}^{T_y} \log p(y_t = y_t^{(m)} | y_{<t}^{(m)}, \mathbf{x}^{(m)}), \tag{2.39}$$

where the log probability inside the inner summation is from Eq. (2.38). It is important to note that the ground-truth target symbols $y_t^{(n)}$ are used during training. The entire model is differentiable, and the gradient of the log-likelihood function with respect to all the parameters $\boldsymbol{\theta}$ can be computed efficiently by backpropagation. This makes it straightforward to use stochastic gradient descent or its variants to train the whole model jointly to maximize the translation performance as explained in the previous sections.

Table 2.1: Decoder phases in an attentional encoder-decoder architecture.

Encoder-Decoder with Attention	
Phase	Output $\leftarrow$ Input
Look	$\mathbf{c}_j \leftarrow \mathbf{z}_{j-1}, y_{j-1}, \mathcal{C}$
Update	$\mathbf{z}_j \leftarrow \mathbf{z}_{j-1}, y_{j-1}, \mathbf{c}_j$
Generate	$y_j \leftarrow \mathbf{z}_j, y_{j-1}, \mathbf{c}_j$

2.3 Neural Machine Translation

In this section, we give the necessary context for Neural Machine Translation (NMT) starting from history of Machine Translation (MT) followed by the definition of NMT. We then give the motivation behind using NMT as the test-bed of this thesis and provide the details about the specifics of NMT. Specifically, structure of datasets, training and testing (decoding) of NMT models, evaluation metrics and model selection. Finally we explain some specifics about the granularity of input and output sequences.

As described in the previous section, Sequence-to-Sequence models (seq2seq) refer to a very broad family of models, that encapsulate each an every model that map input sequences to output sequences. NMT on the other hand, focuses on automatic translation (machine translation) of a source sequence into a target sequence where both sequences are representations in different natural languages.

Before going into the details of NMT, let us first dissect the term NMT and explain each component from the perspective of this thesis. *Translation* is the task of converting a sequence into another sequence, while preserving the underlying semantics or meaning. Intuitively, this conversion can be represented as a mapping problem between a source sequence and a target sequence. When this mapping problem is addressed by automatic methods, either relying on data covering source and target sequences (eg. parallel data) or prior knowledge about the nature of the sequences (eg. linguistic knowledge), leads us to the basic definition of *Machine Translation*. Finally, *Neural Machine Translation*, is a data-driven statistical machine translation approach to model the mappings between sequences, relying on *connectionist* methods and parameterizations [21].

From our perspective, *machine translation* is a machine learning problem that the task is mapping an arbitrary length input sequence $\mathbf{x}$ to an arbitrary length output sequence $\mathbf{y}$. Since we treat machine translation as a machine learning problem, and we rely on end-to-end connectionist approaches (deep learning methods), the linguistic motivations (or common NLP approaches) are kept out of scope of this thesis. But since *Neural Machine Translation* is chosen as the test-bed of this thesis, we mention the history of *Machine Translation* very briefly.

Although there exist earlier attempts in seventeenth century for mechanical translation and the idea of philosophical (universal) languages with mechanical dictionaries, *Machine Translation* research precursors and pioneers can be dated back between 1933-1956. This era laid the foundations of *Machine Translation* where researchers focused on multilingual mechanical dictionaries later forming the idea of multilingual translation devices [61] and the concept of *interlingua* from well-known Weaver's 1949 memorandum on translation [82]. The after war era between 1956-1966 focused on practical developments and formed one of the most fundamental research group for machine translation at the IBM Corporation (for the early history of MT, please see [60]). A quiet decade between 1967-1976 was followed by operational and commercial systems between 1976-1989, where the research on MT was revived. Research after 1989 focused on corpus-based approaches relying on data statistics and formed the foundations of Statistical Machine Translation (SMT) methods[76]. Incremental research on SMT dominated the field up until 2014, where *Neural Machine Translation* was invented by two Machine Learning groups simultaneously and independently, one from University of Montreal [24] and the other from Google Inc. [118]. Since 2014, NMT not only revolutionized the field of MT [131] but also many other research areas of NLP [95], Computer Vision [22] and Robotics [cite].

NMT is a data-driven, end-to-end connectionist approach for statistical machine translation, requiring no linguistic prior, domain knowledge on NLP or ad-hoc algorithm pipelines tailored for solving the translation (or sequence mapping) tasks only.

Why did we choose NMT as our test-bed?

To our goal, machine translation is a very hard task to be solved, in order to take a step towards understanding the artificial cognition. Now we enumerate the rationales behind choosing NMT as our test-bed.

1. Machine Translation is a challenging real world problem for natural language understanding, and requires a connectionist model to capture the semantics of input and output sequences. In this thesis, our goal is modelling multiple input and output sequences within the same connectionist model (superset of seq2seq). Unlike small toy problems, such as hand-written digit classification, developing such generic models requires hard tasks to solve, where MT is one suitable stress test for development.

2. In many machine learning problems, the limiting factor is the amount of available data, either labelled or unlabelled. The amount of data becomes more important for end-to-end approaches and for the models that are more complex, such as connectionist models (due to the increase in data complexity). The scarcity of the data usually prohibits the model architectures explored (such as restricted use of linear models), preventing researchers to exploit high compute capabilities of modern machinery. MT on the other hand, offers a rich amount of data for many languages, both in terms of quality and quantity. This wide range of datasets are constructed by parliament proceedings, web-crawls, international agencies such as United Nations or European Parliament and many more formal and informal sources. With this large amount of data, we can attack more challenging problems by making use of more sophisticated models. Thanks to the availability of such datasets, researchers quite easily find their models under-fitting even with the vast amount of computational resources. As we aim to solve a very challenging task, modelling not only the mapping between two sequences, but modelling the mapping between multiple sequences, the availability of the data becomes utmost important.

3. Large amount of data mentioned in the previous item also allows us to do controlled experiments by reducing the amount of data used, or choosing complex datasets to benchmark the machine learning architectures. As expected, not all the mappings between any two language pairs are at the same level of diffi-

culty and for some language pairs the amount of data is not as large as the other common spoken languages (low-resource languages eg. Uzbek$\rightarrow$English translation task). MT further allows us to choose and/or simulate the complexity and the amount of data used for developing new model architectures. We can develop architectures using the large datasets and then adjust them to work with increasing complexity and reduced amount of data, for the sake of generality.

4. Another important reason of choosing MT as our test-bed is the availability of multi-view data. In a multi-view dataset (multi-text or multi-way parallel data) each example consists of a tuple of n data points where $n >= 2$. An example in a sequence to sequence mapping problem is a 2-tuple $(\mathbf{x}, \mathbf{y})$ pair, but in a multi-view dataset, which can be exploited by a multi-sequence modelling architecture, an example can be an n-tuple such as $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{y}_1, \mathbf{y}_2)$ where $n = 5$, having three source data points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and two target data points $\mathbf{y}_1, \mathbf{y}_2$, all representing the same semantics of a single observation. Such datasets are quite common in MT community, thanks to multi-national organizations and multi-lingual countries, practically having the same sequence in multiple languages, which is called n-way parallel data or multi-text in the literature. We can develop architectures that can exploit the complementary information across multiple-views, practically benchmark multi-sequence modelling architectures on many-to-one mapping problems (multi-source translation, eg. translating a source sequence given in multiple source languages into a target language) or one-to-many mapping problems (multi-target translation) or even many-to-many mapping problems. This further allows us to extend the architectures to make use of different modalities, opening new research directions for multi-task and multi-modal mapping problems.

5. Finally, we believe that, being able to understand the semantics of language, which is the medium intelligent bodies use to communicate, and coming up with models that can express this complex behavior is a significant step towards developing agents that can make sense of the observational world.

Next, we will describe the specific details about the datasets used in NMT and training procedure for adjusting the model parameters.

2.3.1 Training

In the previous sections, we described a general family of approaches, called sequence-to-sequence models, that can learn to map an input sequence $\mathbf{x}$ to an output sequence $\mathbf{y}$ by learning a continuous function that is trained to estimate the conditional probability of an output sequence given an input sequence, $p(\mathbf{y}|\mathbf{x})$. We explained how to parametrize sequence-to-sequence models by using encoder-decoder architectures and also mentioned how to adjust the parameters of the model given a training set using Stochastic Gradient Descent and Backpropagation. In this section, we detail the training procedures, focusing on NMT specific routines. We start by explaining the structure of the datasets for NMT, *Bilingual Corpus*. Then, the data iteration schemes, how to efficiently make use of available data, *Data Iteration* and finally putting all of them together with improved step rules for SGD, *Training Loop*.

Bilingual Corpus: Recall that, given a training set of $(\mathbf{x}, \mathbf{y})$ pairs $\mathcal{D} = \{\mathbf{x}^{(m)}, \mathbf{y}^{(m)}\}_{m=1}^{M}$ with M samples, training an NMT model is done by maximizing the conditional log-likelihood of the training set :

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{n=1}^{N} \sum_{t=1}^{T_y} \log p(y_t = y_t^{(m)} | y_{<t}^{(m)}, \mathbf{x}^{(m)}). \tag{2.40}$$

Each input-output (source-target) pair $(\mathbf{x}^{(m)}, \mathbf{y}^{(m)})$, consists of sequences having the same meaning. For NMT, a general approach is to use sentences to represent sequences, meaning each source-target pair is a sentence given in two languages. When two documents are aligned in a way that each sentence in a document has a corresponding translation sentence in the other document, it is called a *bilingual corpus*, *parallel text* or *bi-text*.

Following the basic training procedure of mini-batch SGD in Alg. (1), we need to form mini-batches to feed the training algorithm. However, there is an efficiency problem in sequential data of arbitrary length. The samples chosen by SGD can be in varying length inside of a mini-batch, causing the training procedure waste computational time for short sequences, when they are mixed with long sequences within the same mini-batch. This problem is partially mitigated by following an iteration scheme.

Algorithm 2 Bilingual Corpus Iteration Scheme

```
Given Dataset $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{M}$ of $M$ examples, batch-size $n$ and read-ahead count $k$
repeat until the end of epoch
    1: Shuffle dataset, preserving source-target pair alignments.
    2: Read-ahead $k \times n$ examples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$.
    3: Sort according to target (or source) sequence lengths.
    4: Form batches of approximately equal sized examples.
    5: Feed training algorithm with the formed batches one-by-one.
```

Data Iteration: The main goal of the data iteration scheme is forming mini-batches where each example in the mini-batch has approximately the same length. By preserving such homogeneous batches, (1) we save computation time, (2) the average gradient for the mini-batch becomes consistent and can be associated with a particular sequence length. We use Alg. (2) as our iteration scheme, which is also illustrated in Fig. 2.11. Note that, the batching strategy described above is being used in almost all of the seq2seq implementations [118, 24, 25, 27, 3] but details can only be found in public codebases[15] [16] [17], therefore we value mentioning it here.

Note that, the above procedures can trivially be extended to multi-way parallel data.

Training Loop: After deciding on an efficient iteration scheme, next we focus on the training loop, where an iterative procedure is executed to optimize the model parameters over the training set $\mathcal{D}$. Although vanilla mini-batch SGD algorithm is quite robust and resilient, practitioners usually need to experiment for a convenient learning schedule (ie. annealing the learning rate η with a schedule). This process necessitates trial and error which can be very expensive when the training time for a model is as long as weeks. In such cases, we resort to adaptive learning rate algorithms, such as Rmsprop[122], Adadelta [135] or Adam [73]. In this thesis, in order to avoid the manual tuning of the learning rates, we choose to use Adam step-rule to optimize the model parameters. Adam step-rule is described in Alg. (3).

[15] https://github.com/lisa-groundhog/GroundHog

[16] https://github.com/mila-udem/blocks-examples/tree/master/machine_translation

[17] https://github.com/nyu-dl/dl4mt-tutorial

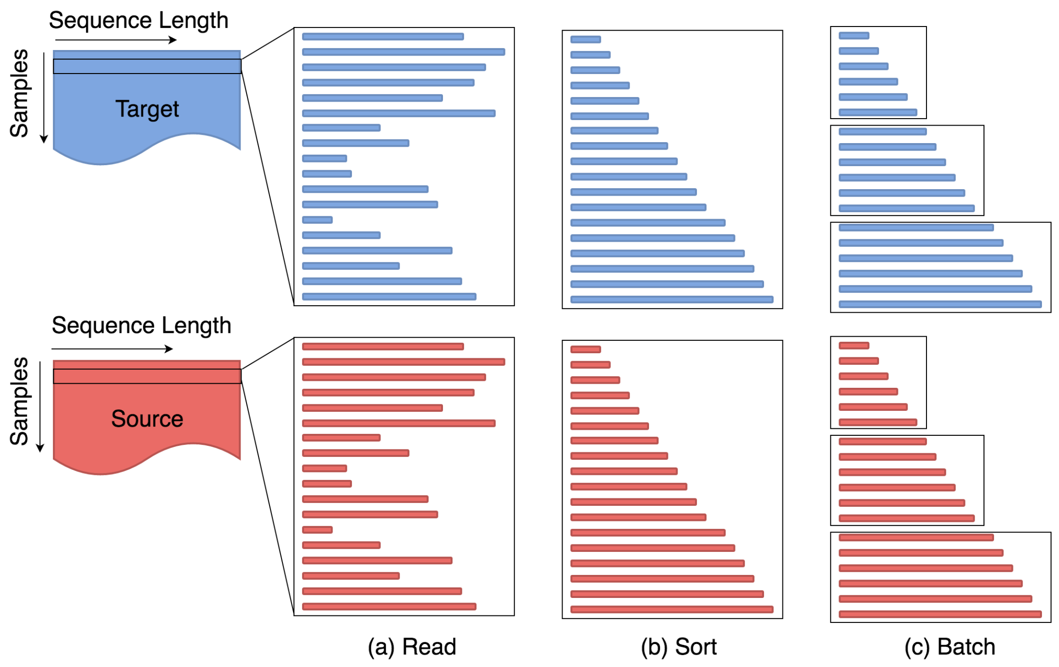


Figure 2.11: Batching strategy for parallel, data-iteration. The data iterator first reads ahead multiple source-target pairs (examples) from corresponding source and target corpora. Then the sequences are sorted according to the sequence lengths (either wrt source or target) in ascending order. Then sorted chunks will be formed as batches to be fed to the training algorithm.

2.3.2 Testing

After having trained a NMT model which estimates the conditional probability $p(\mathbf{y}|\mathbf{x})$, the next question is how to generate the translation given a source sequence $\mathbf{x}$. Formally,

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}). \tag{2.41}$$

As can be seen above, exact solution to Eq. (2.41) requires computing all possible $\mathbf{y}$. Unfortunately, most probable translation cannot be found exactly, because of the intractability search space in the output $\mathbf{y}$ induced by the recurrent formulation, approximate search methods are employed in the literature. A common approach is to use greedy beam-search. Beam-search starts with a set (beam) of hypotheses translations, empty at the very beginning of decoding. At each iteration/decoding step,

Algorithm 3 Adaptive Moment Estimation SGD (Adam)

```
1: Given Hyperparameters $\eta, \beta_1, \beta_2, \epsilon$
2: Initialize update-step $t \leftarrow 0$
3: Initialize $1^{st}$ moment vector $m_0 \leftarrow 0$
4: Initialize $2^{nd}$ moment vector $v_0 \leftarrow 0$
5: repeat until convergence
6:     $t \leftarrow t + 1$
7:     Compute gradients of the loss w.r.t. model parameters: $g_t \leftarrow \nabla_\theta \mathcal{L}$
8:     Update biased $1^{st}$ moment estimate: $m_t \leftarrow \beta_1 \cdot m_t + (1 - \beta_1) \cdot g_t$
9:     Update biased $2^{nd}$ moment estimate: $v_t \leftarrow \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_t^2$
10:    Bias correction for $1^{st}$ moment: $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$
11:    Bias correction for $2^{nd}$ moment: $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$
12:    Update: $\theta \leftarrow \theta - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$
```

candidate translations are formed by extending hypothesis sequences in the beam by one token. After this extension of each candidate translation, all the sequences in the beam are scored using the model (simply checking the log-probabilities) and only the top k scoring hypothesis are kept in the beam, k being the beam-size, the rest of the hypothesis are discarded. The decoding stops when a beam is appended an end of sequence token </s>, ie. when the model emits an end of sequence token.

Beam search is quite robust to the selection of beam-size hyper-parameter k [26, 69] and a small beam-size between 2-10 is usually used for NMT.

2.3.3 Automatic Metrics for Evaluation

In the previous sections we describe, how to train a NMT model, and how to generate translation using the trained model. Now, we describe evaluation metrics to measure the goodness of generated translations. As in many other evaluation tasks, the ultimate evaluation is the assessment by human evaluators, unfortunately, human evaluation is costly, time-consuming and necessitates trained human translators. Therefore MT research relies on automatic metrics for evaluation, that try to mimic human evaluation and give a score to a generated hypothesis translation, using gold-label (true) translation as reference. In this section, we describe three automatic metrics for the evaluation of generated translations, Bilingual Evaluation Understudy Score (BLEU), Translation Error Rate (TER) and TB score which is the average of BLEU and TER.

Bilingual Evaluation Understudy - BLEU[99] BLEU score is the most common automatic metric for evaluating machine translated sequences. BLEU is a precision based metric and compares a hypothesis (candidate) translation against a reference (true) translation. BLEU computes the geometric mean of the modified n-gram precision scores multiplied by a brevity penalty, formally:

$$BLEU = BrevityPenalty * \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \tag{2.42}$$

$$BrevityPenalty = \begin{cases} 1, & \text{if } \ c > r, \\ \mathrm{e}^{1-r/c}, & \text{otherwise} \end{cases} \tag{2.43}$$

$$p_n = \frac{\sum_{S \in C} \sum_{ngram \in S} \hat{c}(ngram)}{\sum_{S \in C} \sum_{ngram \in S} c(ngram)}, \tag{2.44}$$

where C is a corpus of all the translations, S is a set of all unique n-grams in one sentence in C, $c(ngram)$ is the count of n-gram, w_n is the weight associated with p_n (usually taken as a uniform weight, $w_n = 1/N$) and finally, $\hat{c}(ngram)$ is $min(c(ngram), c_{ref}(ngram))$ as $c_{ref}(ngram)$ being the count of n-gram in the reference sequence. N is usually selected to be four, to consider n-grams only up to four and brevity penalty controls the effect of length, in order not to favor short translations. Intuitively, BLEU score tries to match n-gram overlaps between candidate and reference translations weighted by a brevity penalty that penalizes sequences that are not matching in sequence lengths. BLEU scores are usually normalized to be between 0-100 range and higher BLEU scores are better.

Translation Error Rate TER is a simple error metric for machine translation that measures the number of edits required to change a candidate translation into reference translation. TER scores are usually normalized between 0-100 and lower TER scores are better.

TB score is simply a way of combining BLEU and TER to have a single measure and computed by TB = (TER-BLEU)/2.

All the measures mentioned above, are correlated (with a high positive or negative value) with the log of the conditional probabilities defined in Eq. (2.38), and this observation is illustrated in Fig. 2.12 comparing BLEU with negative log-probability and Fig. 2.13 for TB with negative log-probability for validation sets.

2.3.4 Linguistic Units of Input and Output Sequences

One final topic to be mentioned which is necessary for NMT context is the granularity of input and output tokens. Consider a scenario, where the total number of unique symbols in vocabulary V, the vocabulary size $|V|$ is extremely large (revisiting Section 2.1.2), resulting giant embedding matrices for both source and target tokens ($\mathbf{E}_x$ and $\mathbf{E}_y$) also a softmax layer that again spans the vocabulary size V. This problem is called large-vocabulary problem [63] which is still an open research question. Since we are not interested about the particular NLP vocabulary reduction approaches or approximation techniques for large integrals, we employ a simple unsupervised technique called Byte Pair Encoding (BPE) [111] that segments words in a sequence of sub-word units according to the co- occurrence statistics of characters n-grams. BPE encodes the sequences in a way that, words appearing in the training corpus with high frequency are kept intact, but rare words are splitted into subword units or even character strings. BPE allows a budget for the number of co-occurrence statistics to be considered, practically letting us choose the vocabulary size $|V|$ that is convenient for the task at hand.

2.4 Multi-Sequence Modelling

In the previous sections, we defined the internals and functionalities of the building blocks of this thesis, namely connectionist sequence modelling. We further formalized the Sequence-to-Sequence (seq2seq) approach as an introduction to multi-sequence modelling, since seq2seq architectures, by definition, are the simplest form of multi-sequence modelling architectures but only consider two sequences, input and output sequences. In this thesis, we focus on a broader class of multi-sequence mod-

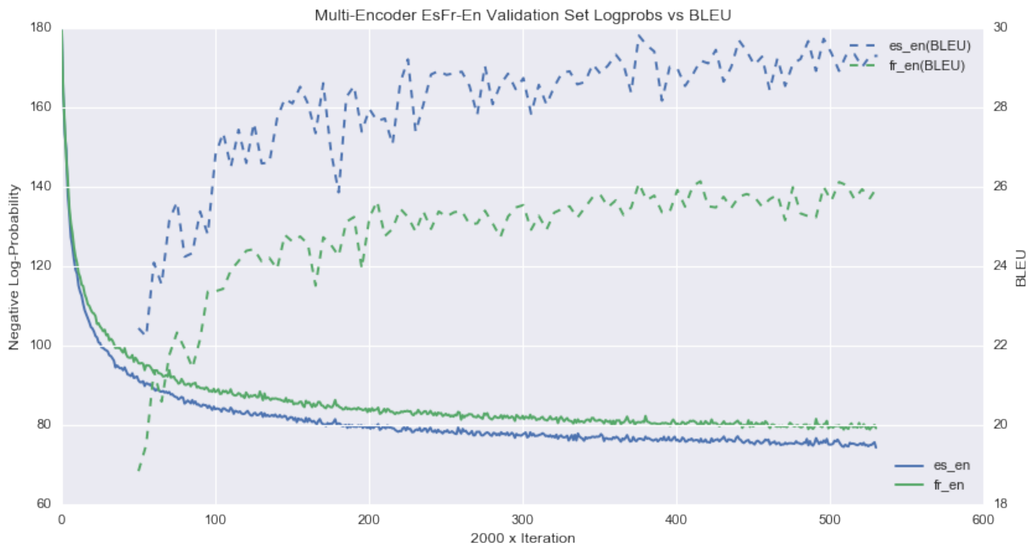


Figure 2.12: BLEU Score versus Negative Log-Probability (NLL) on validation. The x-axis corresponds to the number of iterations taken during training, and y-axis corresponds to Negative Log-Probability (left) and BLEU Score (right). As can be seen, BLEU Score and NLL negatively correlates for two translation pairs, French-English (green curves) and Spanish-English (blue curves).

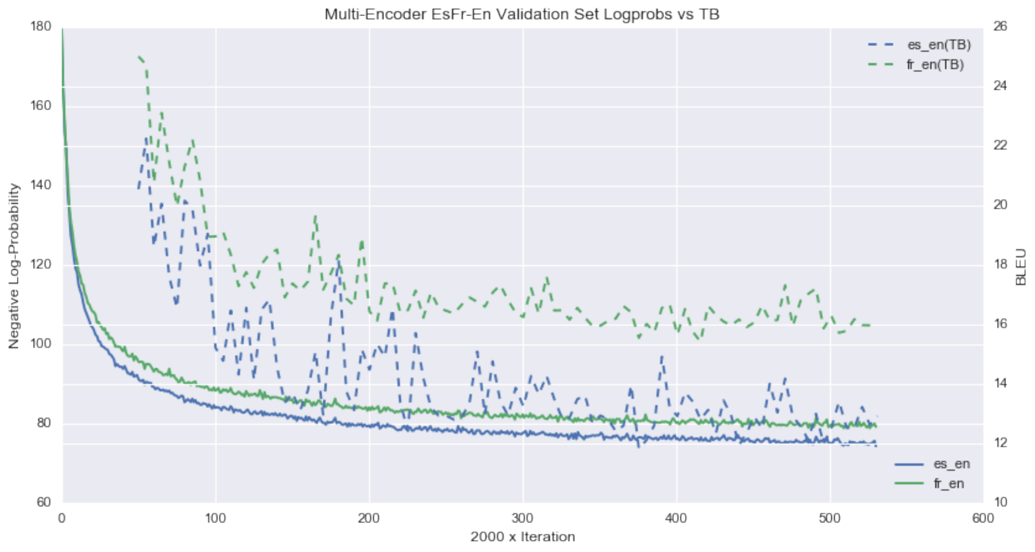


Figure 2.13: TB Score versus Negative Log-Probability (NLL) on validation. The x-axis corresponds to the number of iterations taken during training, and y-axis corresponds to Negative Log-Probability (left) and TB Score (right). As can be seen, TB Score and NLL positively correlates for two translation pairs, French-English (green curves) and Spanish-English (blue curves).

elling architectures, where we have more than two associated sequences both at the input and the output.

Let us first define and formalize what do we mean by multi-sequence modelling, using the definitions and building blocks introduced in the previous sections. Let $\mathbf{\Gamma}$ be a set of N input sequences, such that $\mathbf{\Gamma} = \{\mathbf{x}^i\}_{i=1}^N$, where each input sequence $\mathbf{x}^i$ belongs to a different source domain. For example, for $1 = en$ and $2 = fr$, $\mathbf{x}^{1=en}$ can be the sequence of tokens representing a sentence in English, $\mathbf{x}^{en} = (x_1^{en}, \dots, x_{T_x^{en}}^{en})$ and $\mathbf{x}^{2=fr}$ can be the sequence of tokens representing a sentence in French, $\mathbf{x}^{fr} = (x_1^{fr}, \dots, x_{T_x^{fr}}^{fr})$. Without loss of generality each $\mathbf{x}^i$ can represent an input sequence in a language (stream of words), image (stream of patches), video (stream of images) or speech (stream of audio signals). Further let $\mathbf{\Lambda}$ be a set of M output sequences, $\mathbf{\Lambda} = \{\mathbf{y}^j\}_{j=1}^M$, where each output sequence j belongs to a different domain (eg. $\mathbf{y}^1$ representing a sentence in German and $\mathbf{y}^2$ a sentence in Turkish).

Given a set of inputs $\mathbf{\Gamma}$ and a set of outputs $\mathbf{\Lambda}$, we define the mapping problem of $\mathbf{\Gamma} \rightarrow \mathbf{\Lambda}$ as *multi-sequence mapping* and we define the predictive modelling of this mapping using a single parametric function $\mathbf{\Pi}$ as *multi-sequence modelling*, such that $\mathbf{\Pi} : \mathbf{\Gamma} \rightarrow \mathbf{\Lambda}$.

We further state that, such multi-sequence mapping models should not necessarily rely on the existence of all of its inputs and outputs at the same time, both during training and inference steps. Since this behavior is imperative, it needs further clarification. Given the set of N input domains and M output domains, it is very unlikely that the real world data is available for all possible pairs from source to target ($N \times M$ cartesian). A realistic scenario is having high amount of parallel data between some pairs (i, j), from source domain i to target domain j. Concretely, data in the form of $(\mathbf{x}^i, \mathbf{y}^j)$ (eg. bi-text between English and French). In addition to the available parallel data, we usually have access to a small amount of n-way parallel data, such that data in the form of $(\mathbf{x}^i, \dots, \mathbf{x}^j, \dots, \mathbf{y}^k)$ where $n = |(\mathbf{x}^i, \dots, \mathbf{x}^j, \dots, \mathbf{y}^k)|$, (eg. 3-way parallel data, between English, French and German)[18].

For the majority of the pairs (i, j), we usually do not have access to any direct parallel data (eg. between Turkish-Afrikaans). A multi-seq2seq architecture should be able to

[18] The same sentence is given in three different languages.

be trainable by only parallel data and still be able to operate across all possible pairs from source to target, with the premise of generalizing to unseen pairs (i, j) during training.[19]. The model should indeed exploit and make use of any available n-way parallel data both during training and inference, but not solely rely on it's availability. Given the above premise for multi-seq2seq architecture, in this thesis, we do not impose any necessity of the availability of data conditions to multi- seq2seq and we call this necessity, as *multi-way parallel data necessity*.

As can be seen from the definition of $\mathbf{\Pi}$, it can represent a broad family of functions (many-to-one mappings, one-to-many mappings, many-to-many mappings across source and target domains) and many machine learning problems, if not all, can be represented in this framework. The definition of $\mathbf{\Pi}$ is intentionally not been narrowed down at this point in order to emphasize the magnitude of problems that can be encountered, which are analyzed in the subsequent sections.

2.4.1 From Sequence to Sequence Models to Multi-Task Sequence to Sequence Models

Now, let us narrow down the definition of $\mathbf{\Pi}$ starting from Sequence-to-Sequence mapping (seq2seq) problem by varying the cardinality of input set $\mathbf{\Gamma}$ and output set $\mathbf{\Lambda}$, N and M respectively. We first set both cardinalities to one, $N = M = 1$, and then first increase the input set cardinality to be greater than one, $N > 1, M = 1$, second increase both input and output set cardinalities to be greater than one, $N > 1, M > 1$.

2.4.1.1 One-to-One seq2seq

Following the task of seq2seq mapping problem, multi-sequence mapping (multi-seq2seq) problem can be reduced to seq2seq when we consider only one domain of input and one domain of output sequences. Explicitly, when $N = 1$ and $M = 1$, multi-seq2seq reduces to seq2seq problem. As described in the previous section, we consider seq2seq models that parametrize the mapping function from source to target using neural networks only and in the framework of *encoder-decoder* architectures.

[19] even there does not exist any parallel data between

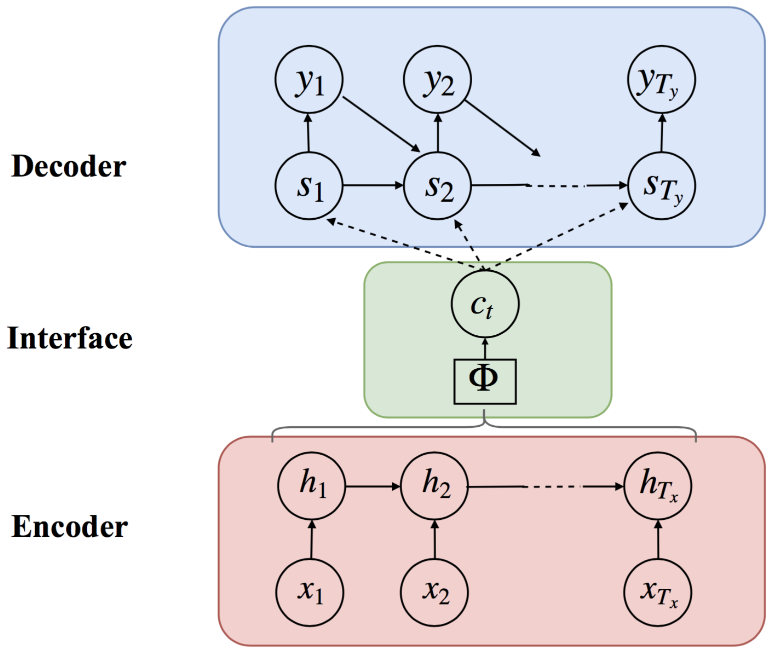
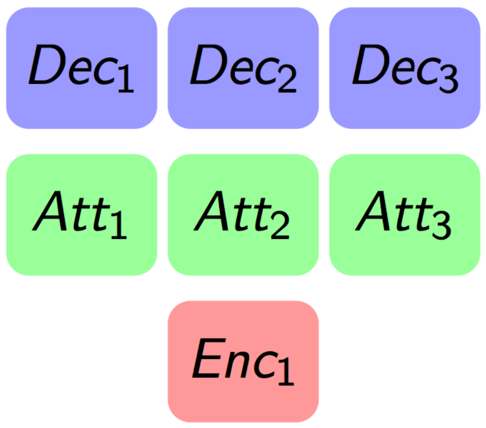


Figure 2.14: A Generalized View of Sequence to Sequence Models.

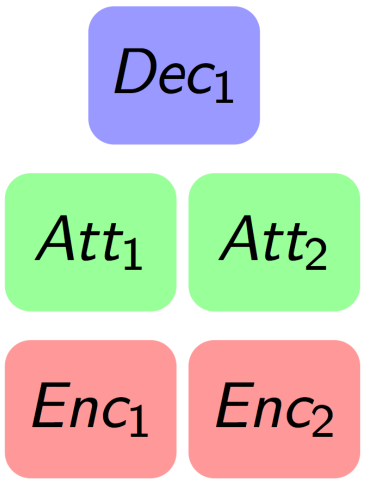
To start with, let us generalize the *encoder-decoder* architecture that can be used to build multi-seq2seq models.

To examine at a high level, regardless of the type of seq2seq architecture, whether basic encoder-decoder (Sec. 2.2.1) or encoder-decoder with attention (Sec. 2.2.2), all seq2seq models consist of three major components (sub-modules or sub-networks), see Fig. 2.14. (1) an encoder, that summarizes the input sequence $\mathbf{x}$ into a set of representations (annotations) $\mathcal{C}$, the red box in Fig. 2.14, (2) a decoder, that generates an output sequence $\mathbf{y}$ by conditioning on the previously generated outputs $y_{<t}$ and the information coming from encoder c_t, the blue highlighted box in Fig. 2.14. The last component, (3) the interface, is the most crucial component of a seq2seq architecture, and responsible to provide the necessary information c_t to the decoder, whenever it is queried, the green box in Fig. 2.14. The interface, practically stitches an encoder with a decoder and can be as simple as an identity function (basic encoder-decoder) or can be a high capacity, time-step sensitive neural network (encoder-decoder with attention).

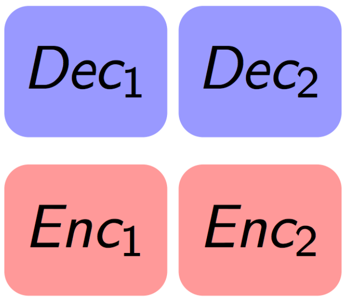
By definition, seq2seq models consider only one domain of input sequence and solve the task of mapping into only one domain of output sequence. Such tasks can easily be



(a) One-to-One seq2seq



(b) Many-to-One seq2seq



(c) Many-to-Many seq2seq

Figure 2.15: Multi Sequence-to-Sequence Model Variants.

machine translation [118, 24], image captioning [127, 133], video captioning [134], speech recognition [19], solving discrete problems [125] or even image to image translation [62].

To date, seq2seq models achieve state-of-the-art results in many fields cited above and one of the most generic and powerful sequence mapping tools we have. Unfortunately, the generality of seq2seq models do not apply directly when we switch to multi-seq2seq, problems arise as we start considering more than one domain of inputs or outputs, increasing N and M.

2.4.1.2 One-to-Many seq2seq

Consider the scenario when we have only one source domain, $N = 1$, but multiple target domains, $M > 1$. The task is then, mapping of a source sequence into multiple output sequences, each in a different domain. As an example, translating a sequence of tokens in English, into first a sequence of output tokens in French, and second a sequence of output tokens in German, $N = 1, M = 2$. Such problems are known as multi-task learning problems [15], and involves solving multiple tasks (two different machine translation tasks in our example) withing the same model.

In order to use seq2seq framework to attack one-to-many seq2seq, we first need to address one particular issue, the interface problem. As we increase M, (1) how do we model/parametrize the interface?, (2) does it scale as we keep increasing M? The simplest solution is to use multiple interfaces, one for each decoder, see Fig. 3.2a. Individual interfaces, then can be trained using the available parallel data between the source and the corresponding target domain. One such example for neural machine translation is proposed by [36] where it has been shown that one-to-many (multi-task) seq2seq can be applied to machine translation. We see two fundamental problems with the approaches that use dedicated interfaces for each task. First, the knowledge transfer from one task to the other task is lightly handled, because only the encoder side of the whole architecture is shared across tasks, hence it is not data efficient. The second problem is about the generality. In order to introduce a new task to the model (increasing M by one), we do not only need adding a new decoder, but also a new interface. This may not seem to be a problem when the interface is an identity function (basic encoder-decoder) but becomes significant when the parametrization of the interface involves a complex component (encoder- decoder with attention). It is necessary to emphasize that, basic encoder- decoder architecture is not competitive with it's counterpart due to the problems mentioned in Section 2.2.1.

2.4.1.3 Many-to-One seq2seq

Next, we consider the scenario when we have multiple source sequences from different domains, only one output domain, $N > 1, M = 1$. This family of problems are one of the most common problems in machine learning, where the task is using multiple sources of information to satisfy one task. From multi-seq2seq perspective, it is similar to one-to-many seq2seq with some minor differences. Here there is only one task to solve, but the model can make use of two separate source of information, either at the same time, or one by one. When the sources are complementary, this multi-source setup has potential to benefit from a diverse set of source information and capture the regularities across source domains. The application of multi-source setup is quite broad, and easily extends to multiple modalities, from Neural Machine Translation [139], image-grounded translation [14, 13] to lip-reading [2, 31].

In a recent study, [139] an architecture with multiple encoders and a single decoder is suggested to achieve multi-source seq2seq, where, two sources are considered, and for both sources, a separate interface is employed. In addition, a dedicated module is devised to combine the information coming from multiple interfaces, see Fig. 3.2b. From a general multi-seq2seq perspective, we can observe some major issues with this approach. First, interface is again kept separate, which leads to data inefficiency for knowledge transfer. Next, every new source side domain has to be plugged into the model with additional interfaces, preventing it to be scalable. Finally, the model trained only using n-way parallel data, and can only do inference when all sources are available, leading *multi-way parallel data necessity*. To develop general multi-seq2seq architectures, we should ensure that the model can still operate even when only one of the sources is given. However, we should be able to exploit the complementary information when such multi- way data is available, during the training and inference stages.

2.4.1.4 Many-to-Many seq2seq

In the family of multi-seq2seq, many-to-many seq2seq is the most generic approach where the model has multiple source domains and multiple target domains, both $N > 1, M > 1$. All the difficulties and problems in both many-to-one seq2seq and one-to-many seq2seq still remain in many-to-many seq2seq, and further emphasized again because of the interface. Consider a many-to-many seq2seq architecture that has N encoder and M decoders and we introduce an additional target domain into the mix. The interface problem becomes more important, since for each new target domain added, we also need to introduce N separate interfaces, to stitch existing encoders to the newly added decoder. This quadratic growth makes many-to-many seq2seq impractical to scale to multiple source and target domains.

When we do not consider any interfaces between encoders and decoders (ie. basic encoder-decoder) where the interface is just an identity function, we can dodge this problem, but resulting multi-seq2seq model will not be compatible due to the lack of attention modules [83], see Fig. 3.2c. [20]

Towards a generic multi-seq2seq architecture, we need to address all the above problems. We should, (1) revisit the notion of interface to mitigate the interface problem observed when we combine the encoders with decoders. (2) enable the multi-seq2seq architecture to be easily extendible to new source domains (encoders) and target domains (decoders). (3) eliminate *multi-way parallel data necessity*. In short, a generic multi-seq2seq architecture should be scalable to new domains, flexible to use any combination of available data without sacrificing its representational capability.

In the next Chapter, we propose our solution to the above problems, by introducing the notion of *shared functional-medium*. As a background, we first build the context for *shared functional-medium* in the next Section from multi-sequence Neural Machine Translation perspective.

2.4.2 Multi-lingual Sequence Modelling (Multi-lingual Neural Neural Machine Translation)

In this section we first answer the question "how does the multi-seq2seq problem can be adapted in multi-lingual neural machine translation?" and then give the intellectual motivation behind the proposed architecture in this thesis.

Communication, by the model of Claude Shannon and Warren Weaver [113], requires three major components: sender, receiver and a channel. We can trivially cast a sequence to sequence mapping problem as a communication problem, simply by renaming sender/encoder, receiver/decoder and channel/interface. This thesis is about the implications of multiple senders and multiple receivers to the structure of the channel and vice-versa. We focus on modelling the channel itself, in order to transmit information from any sender(s) to any receiver(s) and study the channel requirements with respect to the availability and amount of data between sender(s) and receiver(s).

[20] Note that, having an identity function as the interface has an implicit assumption that the interface function is a fixed-point function everywhere in its domain, which is a very restrictive assumption.

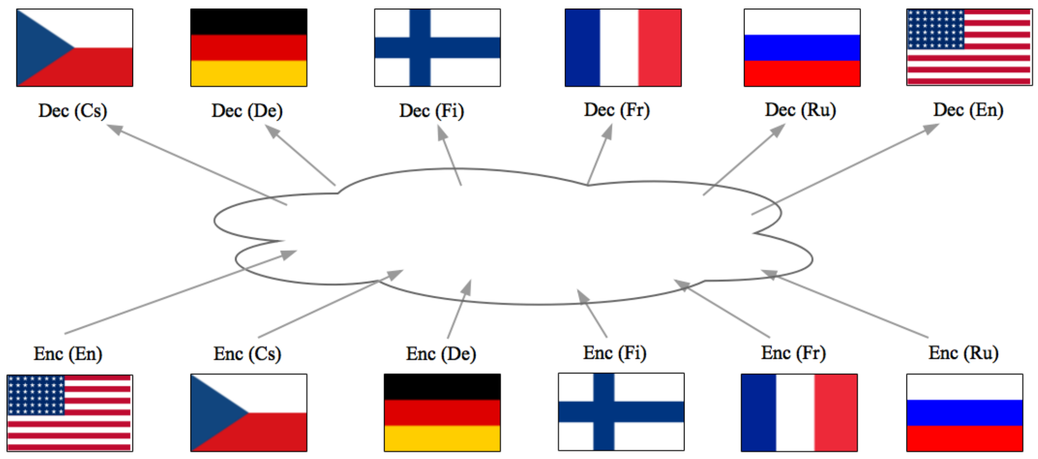


Figure 2.16: Multi-lingual Language Translation as an application of Multi-seq2seq.

The translation problem is indeed a communication problem, and multi-seq2seq problem is just one instance of a communication problem where we have multiple senders and receivers. In Fig. 2.16, an illustration of multi-seq2seq for language translation is provided, where each source and target is a different language, and the channel is represented with a cloud in between.

2.4.2.1 Weaver's Memorandum and Interlingua

Warren Weaver's Memorandum about the translation states that:

".. it is very tempting to say that a book written in Chinese is simply a book written in English which was coded into the "Chinese code." If we have useful methods for solving almost any cryptographic problem, may it not be that with proper interpretation we already have useful methods for translation?"

and the analogy of tall towers:

"..think, by analogy, of individuals living in a series of tall closed towers, all erected over a common foundation. When they try to communicate with one another, they shout back and forth, each from his own closed tower. It is difficult to make the sound penetrate even the nearest towers, and communication proceeds very poorly indeed. But, when an individual goes down his tower, he finds himself in a great open basement, *common to all the towers*. Here he establishes easy and useful communication with the persons who have also descended from their towers. Thus it may be true

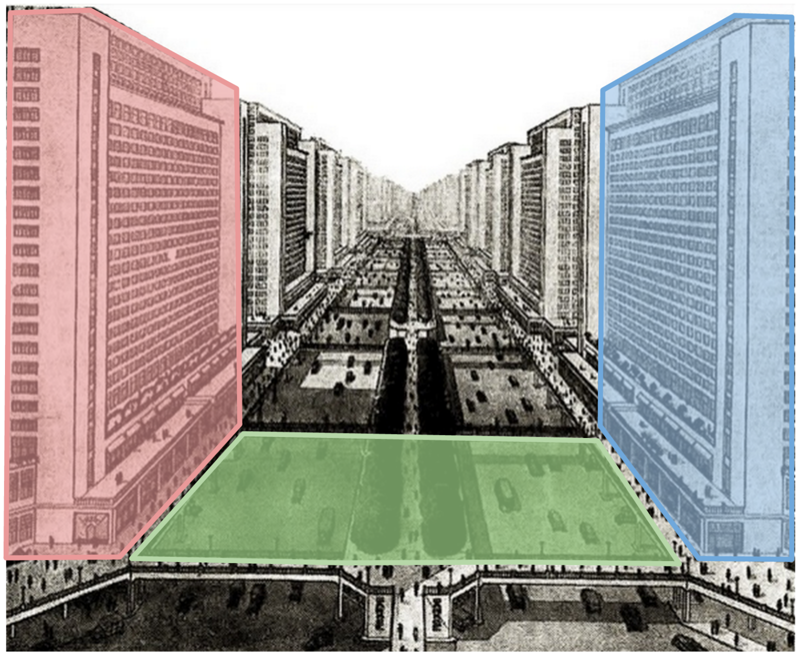


Figure 2.17: Communication medium for single source (sender) and single target (receiver).

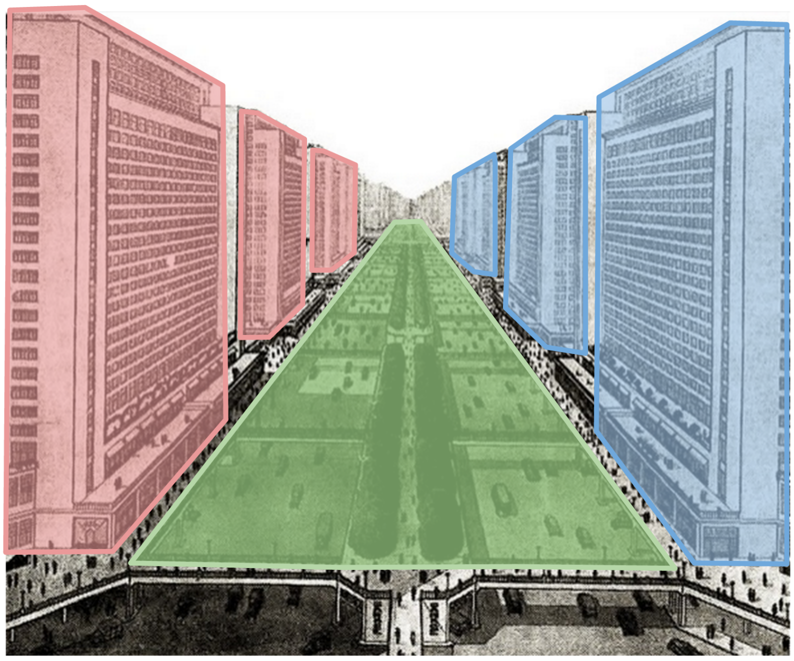


Figure 2.18: Communication medium for multiple sources (senders) and multiple targets (receivers).

that the way to translate from Chinese to Arabic, or from Russian to Portuguese, is not to attempt the direct route, shouting from tower to tower. Perhaps the way is to descend, from each language, down to the common base of human communication – the real but as yet undiscovered *universal language* – and then re-emerge by whatever particular route is convenient.”

The *common base of communication* or the *great open basement common to all the towers* is known as *interlingua.*

The research on modelling *interlingua* has a long history and we can trace back the roots of connectionist approaches to the notion of *interlingua.* Let us first provide some history for *interlingua* from connectionist perspective.

1956-1966 some groups pursued the interlingua ideal, and believed that only fundamental research on human thought process (what would later be called artificial intelligence) would solve the problem of automatic translation.[cite]

In this early period, MT was seen to be of wide relevance in many fields concerned with the application of computers to intellectual tasks; this was true in particular for the research on interlingual aspects of MT, regarded as significant for the development of information languages.[cite]

Solvio Ceccato concentrated on the development of an interlingua based on cognitive process, specifically on the conceptual analysis of words (species, genus, activity type, physical properties etc.) and their possible correlations with other words in texts - a forerunner of the .neural networks of later years.

Interlingua investigations were consonant with the multilingual needs of the Soviet Union and undertaken at a number of other centers (Archaimbault and Leon 1997). The principal one was at Leningrad State University, where a team under Nikolaj Andreev conceived an interlingua not as an abstract intermediary representation but as an artificial language complete in itself with its own morphology and syntax, and having only those features statistically most common to a large number of languages.

In latter half of 1980s, there was a general revival of interest in interligua systems, motivated by contemporary research in artificial intelligence.

Let us now revisit the sequence to sequence problem with Weaver's analogy of towers. It is straight-forward to substitute the encoder with one of the tall buildings and the decoder with the other. The common base open to all towers is then the interface, which navigates the information in the common base, *interlingua*, between an encoder and a decoder, see Fig. 2.17 for illustration.

The multi-sequence modelling, or multi-lingual translation is then, is about modelling the interlingua, when multiple tall buildings are trying to communicate. We increase the number of encoders, the tall buildings on the left in Fig. 2.18 and the decoders, tall buildings on the right, and still keeping a single common base, *interlingua*, among all the source and targets.

We can rephrase the term *interlingua* as *a shared medium* and found our intellectual motivation on top of it. We focus on building a single *shared medium* to control the information flow between any source and target. The modelling of the *shared medium* is critical to the success of multi-seq2seq and it is the major question of this thesis.

2.4.2.2 Benefits and Caveats

Next, let us enumerate the questions we ask in this thesis, before delving deep into the answers given by this thesis.

In this thesis, we hypothesize that, *interlingua* can be modelled using connectionist approaches for the task of multi-seq2seq and we attempt to find answers to the following questions:

1. In Section 3, we try to build a general architecture that is shared across all sources and targets?
2. In Section 4, we handle under-represented (due to the scarcity of available data) pairs in such multi-sequence mapping problem?
3. In Section 5, we benefit having multiple sources/views of the same observation?
4. In Section 6, we transfer the representation capabilities of the proposed architecture, for the cases when we have no direct data between two pairs in the

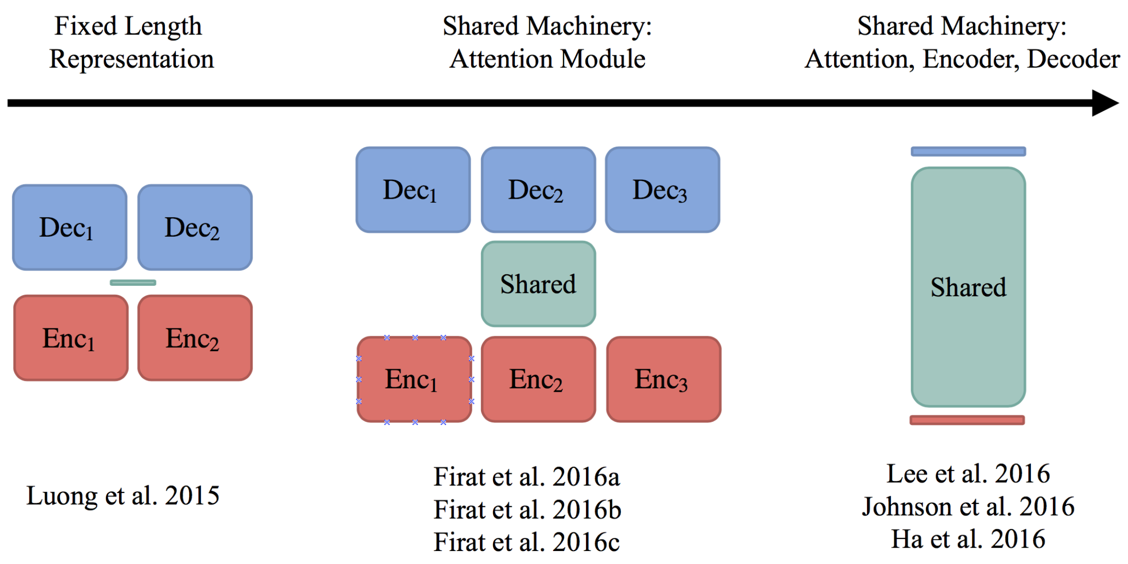


Figure 2.19: Evolution of the Shared Medium Hypothesis

multi-sequence mapping architectures?

The notion of *interlingua* and *shared medium* is not new to the field at all. However, this thesis is one of the first which formalizes a multi-seq2seq problem as a communication problem that relies on a parametrized *shared medium* with neural networks. The research conducted to build this thesis has triggered the field of multi-lingual neural machine translation and as of the writing of this thesis. Our proposed approaches, mainly the *shared medium hypothesis*, adopted and now being used by major research institutes at Google, IBM, Facebook, Microsoft and many others, see Fig. 2.19.

CHAPTER 3

MULTI-SEQUENCE MODELING VIA UNIVERSAL MEDIUM: SHARED ATTENTION MECHANISM

"... frequencies of letters, letter combinations, intervals between letters and letter combinations, letter patterns, etc. which are to some significant degree independent of the language used."

WARREN WEAVER, 1949

In this section, we propose a multi-way, multilingual neural machine translation method which is the first multi-seq2seq architecture, explicitly modelling *interlingua* and constitutes the basis of this thesis. The proposed approach enables a single neural translation model to translate between multiple languages, with a number of parameters that grows only linearly with the number of languages. This is made possible by having a single attention mechanism, that is shared across all language pairs, *the shared medium*. We demonstrate that a single neural network can be trained on ten language pairs from WMT'15 simultaneously and observe a substantial performance improvements over models trained on a single language pair.

3.1 Motivation and Related Work

Existing machine translation systems, mostly based on a phrase-based system or its variants, work by directly mapping a symbol or a subsequence of symbols in a source language to its corresponding symbol or subsequence in a target language. This kind of mapping is strictly specific to a given language *pair*, and it is not trivial to extend

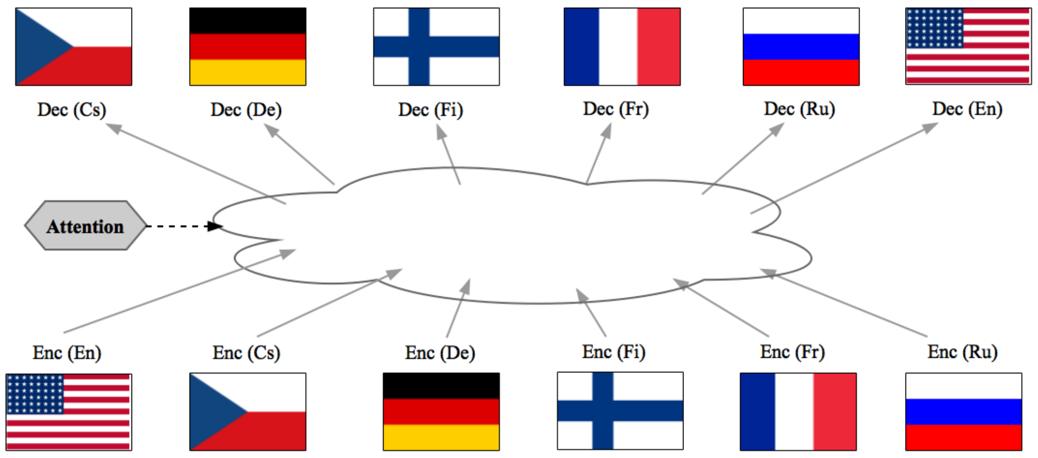


Figure 3.1: Multi-lingual Language Translation with the Shared Medium as Attention Function.

this mapping to work on multiple pairs of languages.

A system based on sequence-to-sequence models (neural machine translation), on the other hand, can be decomposed into two modules. The encoder maps a source sentence into a continuous representation, either a fixed-dimensional vector in the case of the basic encoder-decoder network or a set of vectors in the case of attention-based encoder-decoder network. The decoder then generates a target translation based on the source representation. This makes it possible conceptually to build a system that maps a source sequence in any language to a common continuous representation space and decodes the representation into any of the target sequences, allowing us to make a *multilingual machine translation* system.

This task may look straightforward to implement and has been validated in the case of basic encoder-decoder networks [84]. However, it is not easy to achieve, in the case of the attention-based encoder-decoder network, as the attention mechanism, or originally called the alignment function in [3], is conceptually language pair-specific. In [37], the authors cleverly avoided this issue of language pair-specific attention mechanism by considering only a one-to-many translation, where each target language decoder embedded its own attention mechanism. Also, we notice that both of these works have only evaluated their models on relatively small-scale tasks, making it difficult to assess whether multilingual neural machine translation can scale beyond data-low language translation (low-resource translation).

In [139], authors proposed multi-source encoder-decoder networks with attention for the case where two input sources are given at the same time to exploit multi-text (three-way parallel data). Their proposed attention mechanism consists of two parametric alignment functions, both of which attends two sources separately and then combines context vectors by concatenation. As this architecture still uses a separate attention module for each source-target pair and strictly requires the availability of multi-text, it is not clear how to extend it for the cases where only parallel text is given for multiple source and target pairs.

In this section, we first step back from the currently available multilingual neural translation systems proposed in [84, 37, 139] and ask the question of whether the attention mechanism can be shared across multiple language pairs. As an answer to this question, we propose an attention-based encoder-decoder network that admits a *shared attention mechanism* with multiple encoders and decoders. We use this model for all the experiments, which suggests that it is indeed possible to share an attention mechanism across multiple language pairs.

We train a single model with the proposed architecture on all the language pairs from the WMT'15; English, French, Czech, German, Russian and Finnish. We compare this multi-way, multilingual model against 10 single-pair models and show that they perform comparably to each other, while the number of parameters in the multilingual model is substantially smaller than that of all the single-pair models. This confirms that it is indeed possible to train a single attention-based network to perform multi-way translation, validation the concept of *shared medium*, illustrated in Fig. 3.1.

3.1.1 Existing Approaches

Let us briefly discuss the recent approaches to multilingual neural machine translation in [84, 37].

Neural Machine Translation without Attention In [84], the authors extended the basic encoder-decoder network for multi-task neural machine translation. They extended the basic encoder-decoder network to a set of encoders and decoders, where each of the encoder projects a source sentence into a common vector space. A point in the common space is then decoded into different languages, or into linearized parse tree in the case of parsing, by separate decoders. The authors tried a number of distinct tasks including translation, autoencoding [34], parsing [126] and skip thought [74].

The major difference between [84] and our work is that we extend the attention-based encoder-decoder instead of the basic model. This is an important contribution, as the attention-based neural machine translation has become *de facto* standard in neural translation literatures recently [64, 65, 86, 112, 110], opposed to the basic encoder-decoder.

Another difference is that they do not utilize multilinguality in depth. The authors of [84] tested their models with a single language pair, namely, English and German. On the other hand, in this section, we test with six languages–English, French, Czech, German, Russian, Finnish, and ten pairs across these languages.

One-to-Many Neural Machine Translation In [37], a multilingual translation model is proposed based on the *attention-based neural machine translation*. Unlike the proposal in this thesis, they tested it on one-to-many translation, similarly to the work in [32] where one-to-many natural language processing was done. In this setting, it is trivial to extend the single-pair attention-based model into multilingual translation by simply having a single encoder for a source language and pairs of a decoder and attention mechanism (Eq. (2.33)) for each target language. We will shortly discuss more on why, with the attention mechanism, one-to-many translation is trivial, while multi-way translation is not.

Many-to-One Neural Machine Translation More recently, Zoph & Knight [139] introduced a multi-source neural machine translation which takes as input two source sentences in two different languages and outputs their translation in yet another language. Similarly to [37], the multi-source neural machine translation is equipped with two separate attention mechanisms that operate on each of the two source languages. Beside the fact that this approach works with a single target language, there is another major difference. That is, they assume the availability of three-way parallel training corpora (two source and one target languages) which is often difficult to find in large scale.

3.2 Interlingua in a Shared Functional Form

In this section, we discuss issues and our solutions in extending the conventional *single-pair* attention-based neural machine translation (seq2seq) into *multi-way, multilingual* model (multi-seq2seq).

3.2.1 Problem Definition

We assume $N > 1$ source languages

$$\{X^1, X^2, \ldots, X^N\}$$

and $M > 1$ target languages

$$\{Y^1, Y^2, \ldots, Y^M\},$$

and the availability of $L \leq M \times N$ *bilingual* parallel corpora $\{D_1, \ldots, D_L\}$, each of which is a set of sentence pairs of one source and one target language. We use $s(D_l)$ and $t(D_l)$ to indicate the source and target languages of the l-th parallel corpus.

For each parallel corpus l, we can directly use the log-likelihood function from Eq. (2.40) to define a pair-specific log-likelihood $\mathcal{L}^{s(D_l),t(D_l)}$. Then, the goal of multi-way, multilingual neural machine translation is to build a model that maximizes the joint log-likelihood function with parameters $\boldsymbol{\theta}$,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{L}\sum_{l=1}^{L} \mathcal{L}^{s(D_l),t(D_l)}(\boldsymbol{\theta}). \tag{3.1}$$

Once the training is over, the model can do translation from any of the source languages to any of the target languages included in the parallel training corpora.

3.2.2 Combining Encoders with Decoders: Challenges

A quick look at neural machine translation seems to suggest a straightforward path toward incorporating multiple sequences in both source and target sides. As described earlier already, the basic idea is simple. We assign a separate encoder to each source language and a separate decoder to each target language. The encoder will project a source sentence in its own language into a common, language-agnostic space, from which the decoder will generate a translation in its own language.

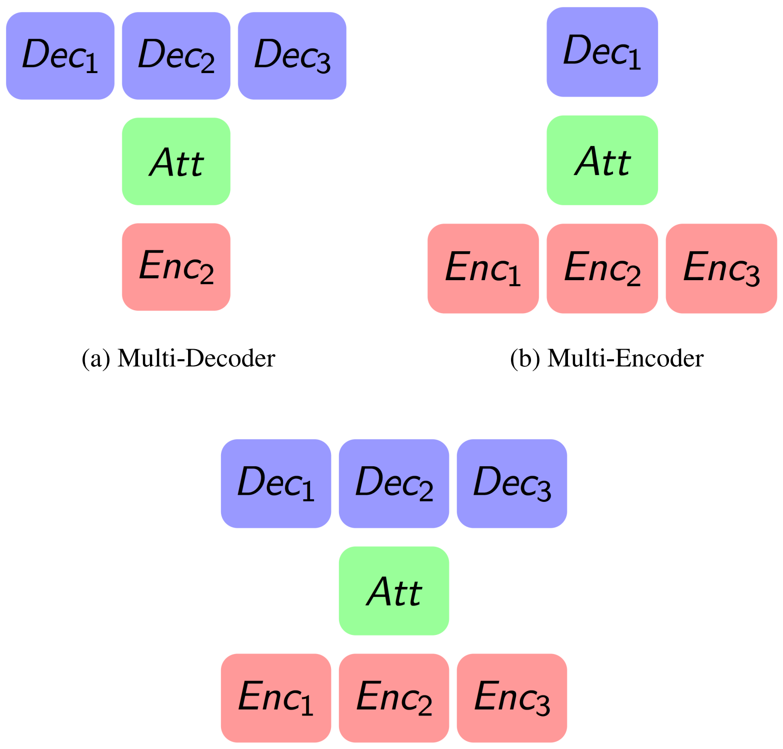
Unlike training multiple single-pair neural translation models, in this case, the encoders and decoders are shared across multiple pairs. This is computationally beneficial, as the number of parameters grows only linearly with respect to the number of languages ($\mathcal{O}(L)$), in contrary to training separate single-pair models, in which case the number of parameters grows quadratically ($\mathcal{O}(L^2)$.) This further prevents overfitting, as the number of training examples stays constant.

The attention mechanism, which was initially called a soft-alignment model in [3], aligns a (potentially non-contiguous) source phrase to a target word. This alignment process is largely specific to a language pair, and it is not clear whether an alignment mechanism for one language pair can also work for another pair.

The most naive solution to this issue is to have $\mathcal{O}(L^2)$ attention mechanisms that are *not* shared across multiple language pairs. Each attention mechanism takes care of a single pair of source and target languages. This is the approach employed in [37, 139], where each decoder or encoder had its own attention mechanism.

There are two issues with this naive approach. First, unlike what has been hoped initially with multilingual neural machine translation, the number of parameters again grows quadratically w.r.t. the number of languages. Second and more importantly, having separate attention mechanisms makes it less likely for the model to fully benefit from having multiple tasks [15], especially for transfer learning towards data-low regimes (resource-poor languages).

In short, the major challenge in building a multi-way, multilingual neural machine translation is in avoiding independent (i.e., quadratically many) attention mecha-



(a) Multi-Decoder (b) Multi-Encoder

(c) Multi-Way

Figure 3.2: Multi Sequence-to-Sequence Model Variants.

nisms. There are two questions behind this challenge. The first one is whether it is even possible to share a single attention mechanism across multiple language pairs. The second question immediately follows: how can we build a neural translation model to share a single attention mechanism for all the language pairs in consideration?

3.3 Shared Attention Mechanism and Multi-Way, Multilingual Model

We describe in this section, the proposed *multi-way, multilingual attention-based neural machine translation*. The proposed model consists of N encoders $\{\Psi^n_{\text{enc}}\}_{n=1}^N$ (see Eq. (2.32)), M decoders $\{(\Psi^m_{\text{dec}}, g^m, f^m_{\text{init}})\}_{m=1}^M$ (see Eqs. (2.36)–(2.38)) and a shared attention module with two sub-modules, *where* component for scoring, f_{score} (see Eq. (2.33) in the single language pair case) and *what* component for aggregation (adaptation), f_{adp}.

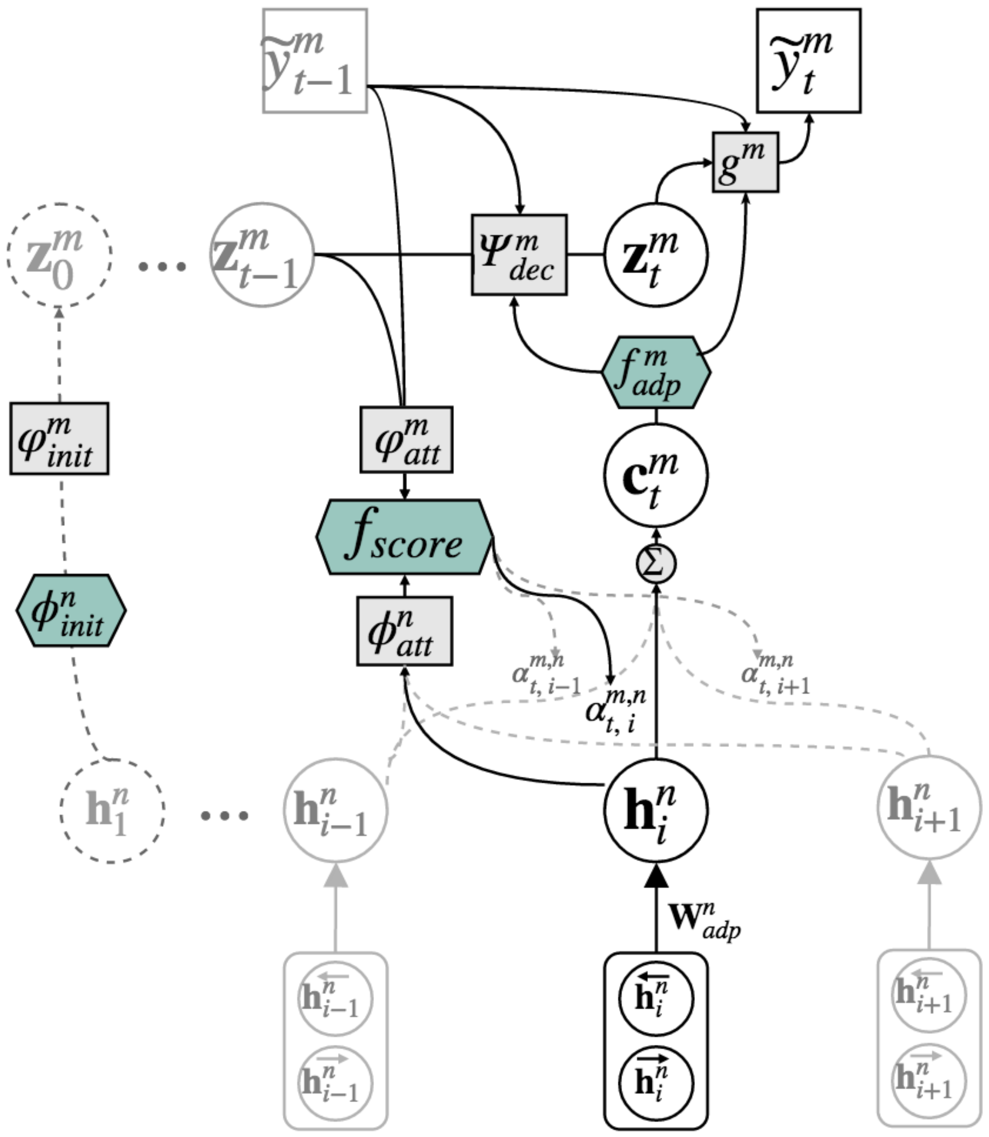


Figure 3.3: One step of the proposed multi-way. multilingual Neural Machine Translation model, for the n-th encoder and the m-th decoder at time step t. Shaded boxes are parametric functions and square boxes represent intermediate variables of the model. Initializer network is also illustrated as the left-most network with dashed boxes. All the shared components are drawn with diamond boxes highlighted in green. See Sec. 3.3 for details.

3.3.1 Encoders

Similar to [86, 139], we suggest one encoder per source language, in which a single encoder is shared for translating the language to multiple target languages. In order to handle different source languages better, we may use a different type of encoder for each source language. For instance, of different size (in terms of the number of recurrent units) or of different architecture (convolutional instead of recurrent.)[1] This allows us to efficiently incorporate varying types of languages in the proposed multilingual translation model.

This however implies that the dimensionality of the context vectors in Eq. (2.32) may differ across source languages. Therefore, we add to the original bidirectional encoder from Sec. 2.2.2, a linear transformation layer consisting of a weight matrix $\mathbf{W}^n_{\text{adp}}$ and a bias vector $\mathbf{b}^n_{\text{adp}}$, which is used to project each context vector into a common dimensional space:

$$\mathbf{h}^n_t = \mathbf{W}^n_{\text{adp}} \left[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t \right] + \mathbf{b}^n_{\text{adp}}, \tag{3.2}$$

where $\mathbf{W}^n_{\text{adp}} \in \mathbb{R}^{d \times (\dim \overrightarrow{\mathbf{h}}_t + \dim \overleftarrow{\mathbf{h}}_t)}$ and $\mathbf{b}^n_{\text{adp}} \in \mathbb{R}^d$. This transformation can also be thought as an adapter layer, that will be necessary when one needs to plug-in a pre-trained encoder to a multi-way multilingual model.

In addition, each encoder exposes two transformation functions ϕ^n_{att} and ϕ^n_{init}. The first transformer ϕ^n_{att} transforms a context vector to be compatible with a shared attention mechanism:

$$\tilde{\mathbf{h}}^n_t = \phi^n_{\text{att}}(\mathbf{h}^n_t). \tag{3.3}$$

This transformer can be implemented as any type of parametric function, and in this thesis, we simply apply an element-wise $\tanh$ to $\mathbf{h}^n_t$.

The second transformer ϕ^n_{init} transforms the first context vector $\mathbf{h}^n_1$ to be compatible with the initializer of the decoder's hidden state (see Eq. (5.3)):

$$\hat{\mathbf{h}}^n_1 = \phi^n_{\text{init}}(\mathbf{h}^n_1). \tag{3.4}$$

[1] For the pairs without enough parallel data, one may also consider using smaller encoders to prevent over-fitting or character/unicode encoders for the task at hand.

Similarly to ϕ^n_{att}, it can be implemented as any type of parametric function. In this thesis, we use a feedforward network with a single hidden layer and share one network ϕ_{init} for all encoder-decoder pairs.

3.3.2 Decoders

We first start with an initialization of the decoder's hidden state. Each decoder has its own parametric function φ^m_{init} that maps the last context vector $\hat{\mathbf{h}}^n_{T_x}$ of the source encoder from Eq. (3.4) into the initial hidden state:

$$\mathbf{z}^m_0 = \varphi^m_{\text{init}}(\hat{\mathbf{h}}^n_{T_x}) = \varphi^m_{\text{init}}(\phi^n_{\text{init}}(\mathbf{h}^n_1))$$

φ^m_{init} can be any parametric function, and in our thesis, we used a feedforward network with a single $\tanh$ hidden layer.

Each decoder exposes a parametric function φ^m_{att} that transforms its hidden state and the previously decoded symbol to be compatible with a shared attention mechanism. This transformer is a parametric function that takes as input the previous hidden state $\mathbf{z}^m_{t-1}$ and the previous symbol $\tilde{y}^m_{t-1}$ and returns a vector for the attention mechanism:

$$\tilde{\mathbf{z}}^m_{t-1} = \varphi^m_{\text{att}} \left(\mathbf{z}^m_{t-1}, \mathbf{E}^m_y \left[\tilde{y}^m_{t-1} \right] \right) \tag{3.5}$$

which replaces $\mathbf{z}_{t-1}$ in Eq. 2.33. In this thesis, we use a feedforward network with a single $\tanh$ hidden layer for each φ^m_{att}.

Given the previous hidden state $\mathbf{z}^m_{t-1}$, previously decoded symbol $\tilde{y}^m_{t-1}$ and the time-dependent context vector $\mathbf{c}^m_t$, which we will discuss shortly, the decoder updates its hidden state:

$$\mathbf{z}_t = \Psi_{\text{dec}} \left(\mathbf{z}^m_{t-1}, \mathbf{E}^m_y \left[\tilde{y}^m_{t-1} \right], f^m_{\text{adp}}(\mathbf{c}^m_t) \right), \tag{3.6}$$

where f^m_{adp} affine-transforms the time-dependent context vector to be of the same dimensionality as the decoder. We share a single affine-transformation layer f^m_{adp}, for all the decoders m.

Once the hidden state is updated, the probability distribution over the next symbol is computed exactly as for the pair-specific model (see Eq. (2.38).)

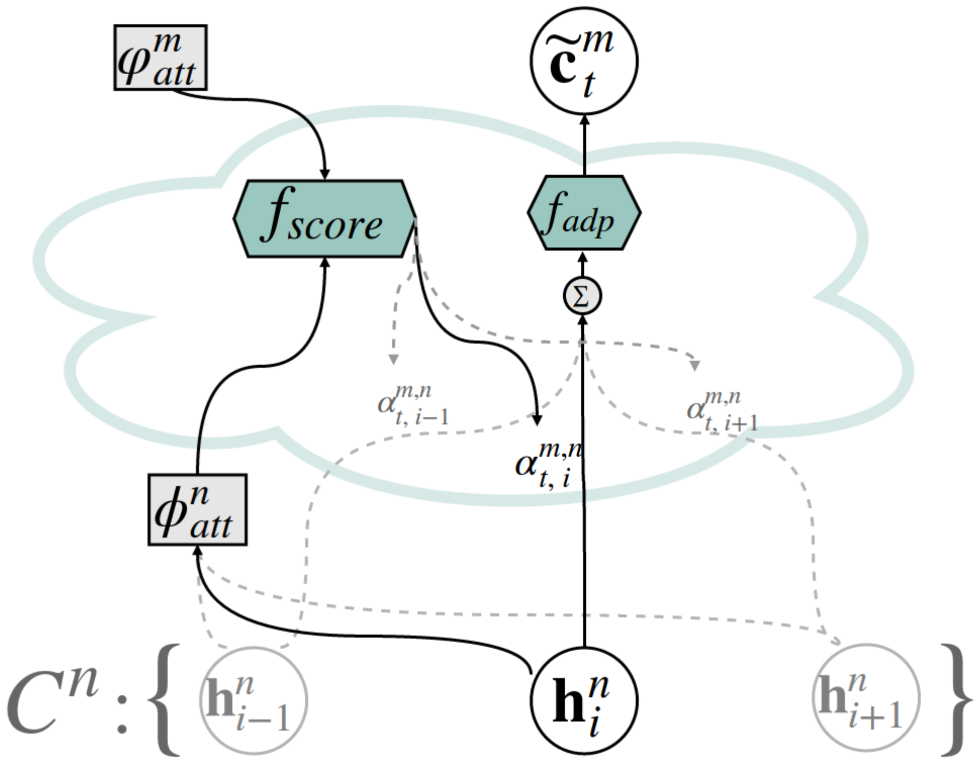


Figure 3.4: Shared Medium Details in multi-seq2seq.

3.3.3 Attention Mechanism

Unlike the encoders and decoders of which there is an instance for each language, there is only a single attention mechanism, shared across all the language pairs. This shared mechanism uses the *attention-specific* vectors $\tilde{\mathbf{h}}^n_t$ and $\tilde{\mathbf{z}}^m_{t-1}$ from the encoder and decoder, respectively.

The relevance score of each context vector $\mathbf{h}^n_t$ is computed based on the decoder's previous hidden state $\mathbf{z}^m_{t-1}$ and previous symbol $\tilde{y}^m_{t-1}$:

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

These scores are normalized according to Eq. (2.34) to become the attention weights $\alpha^{m,n}_{t,i}$.

With these attention weights, the time-dependent context vector is computed as the weighted sum of the *original* context vectors:

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

See Fig. 3.3 for the illustration of the entire model and for the illustration of shared medium, please see Fig. 3.4.

3.3.4 Overview of Shared Medium

Let us again, take a step back and explain how the shared medium functions and what does it mean to have a shared medium across many source and target domains.

Fundamentally, in our proposed model, the hypothesis of *interlingua*, the cloud or the common base to all languages or the channel, is represented by a parametric continuous function. This parametric function constructs the notion of *shared medium*, since only one single function, with a single set of trainable parameters is used to model the *interlingua*. Our shared medium in practice, is responsible to navigate the information between source and target representations, and encouraged to model regularities across multiple pairs. See Fig. 3.6 for a graphical illustration.

By modelling the interlingua with a single parametric function also benefits the ease of modularity. One can increase the number of both encoders and decoders without having need to add additional attention modules, ensuring the scalability of the entire architecture. It is straightforward with the proposed shared medium then, to build a model to perform many-to-one mapping which we call a multi-encoder (Fig. 3.2b), one-to-many mapping model, a multi-decoder (Fig. 3.2a) and many-to-many mapping model, a multi-way seq2seq (Fig. 3.2c).

3.4 Experiments

In this section, we describe the datasets used in our experiments, along with data processing routines and conducted experiments.

3.4.1 Data Preparation

3.4.1.1 Datasets

We evaluate the proposed multi-way, multilingual translation model on all the pairs available from WMT'15[2] Consider $\leftrightarrow$ as seq2seq from left to right and right to left, – English (En) $\leftrightarrow$ French (Fr), Czech (Cs), German (De), Russian (Ru) and Finnish (Fi)–, totalling ten directed pairs. For each pair, we concatenate all the available parallel corpora from WMT'15 and use it as a training set. We use newstest-2013 as the development set and newstest-2015 as the test set, in all the pairs other than Fi-En. In the case of Fi-En, we use newsdev-2015 and newstest-2015 as development and test sets, respectively.

WMT'15 dataset, as can be seen from Table 3.1, consists of both high-resource and low-resource language pairs in terms of available data. En-Fr language pair has almost 20 times more parallel data than En-Fi language pair. In addition to the varying amount of parallel data, considered languages also spread along a wide spectrum in terms of their linguistic difficulty, Czech , German, Russian and Finnish all being morphologically complex. WMT datasets are challenging datasets and considered to be the *real-world* translation problem in the community.

3.4.1.2 Data Preprocessing

Each training corpus is tokenized using the tokenizer script from the Moses decoder.[3] The tokenized training corpus is cleaned following the procedure in [65]. Instead of using space-separated tokens, or words, we use sub-word units extracted by byte pair encoding, as recently proposed in [112]. For each and every language, we include 30k sub-word symbols in a vocabulary. See Table 3.1 for the statistics of the final, preprocessed training corpora.

[2] http://www.statmt.org/wmt15/translation-task.html

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

Table 3.1: Statistics of the parallel corpora from WMT'15. Symbols are BPE-based sub-words.

	# Symbols		# Sentence
	# En	Other	Pairs
En-Fr	1.022b	2.213b	38.85m
En-Cs	186.57m	185.58m	12.12m
En-Ru	50.62m	55.76m	2.32m
En-De	111.77m	117.41m	4.15m
En-Fi	52.76m	43.67m	2.03m

3.4.2 Training and Evaluation

3.4.2.1 Training Setting

We train each model using stochastic gradient descent (SGD) with Adam [72] as an adaptive learning rate algorithm. We use the initial learning rate of $2 \cdot 10^{-4}$ and leave all the other hyperparameters as suggested in [72]. Each SGD update is computed using a minibatch of 80 sentence pairs, unless the model is parallelized over two GPUs, in which case we use a minibatch of 60 pairs. We only use sentences of length up to 50 symbols during training. We clip the norm of the gradient to be no more than 1 [100]. All training runs are early-stopped based on BLEU on the development set.[4] As we observed in preliminary experiments better scores on the development set if we further finetune the shared parameters and output layers of the decoders in the case of multilingual models, we do this for all the multilingual models. During finetuning, we clip the norm of the gradient to be no more than 5.

3.4.2.2 Model and Data Parallelism

The size of the multilingual model grows linearly w.r.t. the number of languages. We observed that a single model that handles six source and six target languages does not

[4] For multi-way multilingual setup, we early stop, by taking a majority voting score from the dev set BLEU scores of all 10 language pairs. Specifically at each iteration, we compute the number of language pairs that has better BLEU scores compared to the previous iteration. We decide to early-stop when the number of language pairs showing better scores does not change (subject to a *patience*, which was taken 20 in our experiments.)

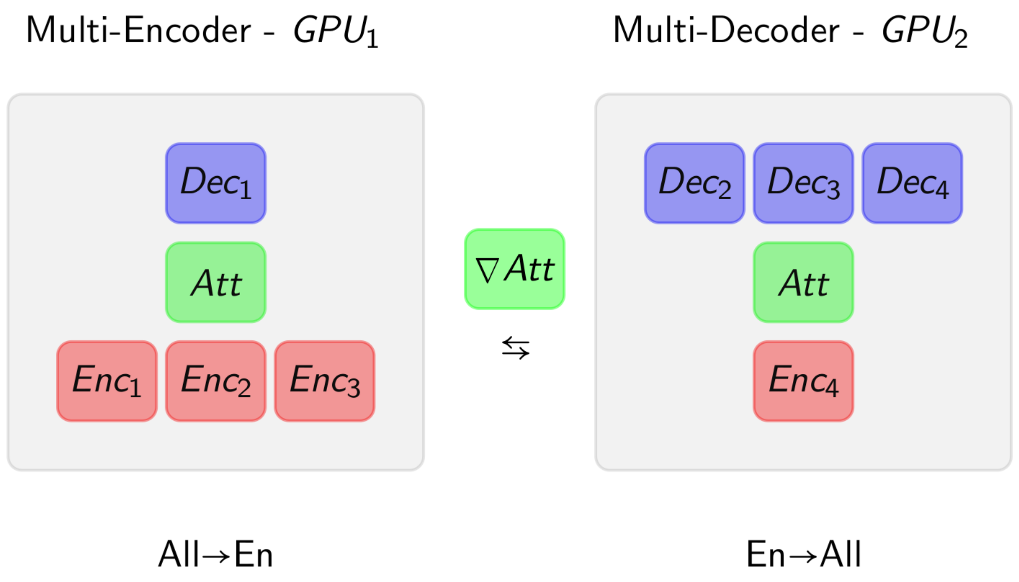


Figure 3.5: Multi-GPU Training of multi-seq2seq Architecture. ∇Att corresponds to the gradients of the loss function with respect to the attention module (shared components), that is being exchanged across GPUs.

fit in a single GPU[5] during training. We address this by distributing computational paths according to different translation pairs over multiple GPUs, following [35]. The shared parameters, mainly related to the attention mechanism, is duplicated on both GPUs, see Fig.3.5 for a sketch of model and data parallelism.

In more detail, we distribute language pairs across multiple GPUs such that those pairs in each GPU shares either an encoder or decoder. This allows us to avoid synchronizing a large subset of the parameters across multiple GPUs. Only the shared attention mechanism, which has substantially less parameters, is duplicated on all the GPUs. Before each update, we build a minibatch to contain an approximately equal number of examples per GPU in order to minimize any discrepancy in computation among multiple GPUs. Each GPU then computes the gradient with respect to the parameters on its own board and updates the local parameters. The gradients with respect to the attention mechanism are synchronized using direct memory access (DMA). In this way, we achieve near-linear speed-up. At each update on both GPUs, we ensure that the exchanged gradients are gathered from approximately equal sized batches and from language pairs in opposite direction, eg. computed gradients on GPU_1 correspond to task of En→Fr and computed gradients on GPU_2 correspond to

[5] NVidia Titan X with 12GB on-board memory

task of Fr$\rightarrow$En. We value to note that, by ensuring such exchange policy, the attention module, *shared medium*, is informed about the error signals for both tasks at the same time, and moves in the parameter space respectively. We observed instabilities and large fluctuations when gradients from tasks from different language pairs are mixed.

3.4.2.3 Schedule

As we have access to bilingual corpora only, each minibatch updates only a subset of the parameters. Excessive updates based on a single language pair may bias the model away from the other pairs. To avoid it, we cycle through all the language pairs, one pair of a minibatch at a time, in Fi$\leftrightarrows$En, De$\leftrightarrows$En, Fr$\leftrightarrows$En, Cs$\leftrightarrows$En, Ru$\leftrightarrows$En order.[6] Initial experiments on random scheduling across pairs and increasing the number of consecutive updates for a pair did not give better results in contrast caused the very problem of *catastrophic forgetting*[43]. We observed delayed learning curves with degraded performance when we increase the update interval (eg. updating one task multiple times before updating the other task), which can be explained by forgetting of one task because of over-exposing another task. This issue is also partially mitigated by the distributed training of the entire architecture, since the model updates it's parameters only after seeing examples from two tasks and exchanging the gradients for the shared components.

3.4.2.4 Evaluation Metric

We mainly use BLEU as an evaluation metric using the "multi-bleu.perl" script from Moses.[7] BLEU is computed on the tokenized, true-case text after merging the BPE-based sub- word symbols. We further look at the average log-probability assigned to reference translations by the trained model as an additional evaluation metric, as a way to measure the model's density estimation performance independent of any error caused by approximate decoding (beam-search).

[6] $\leftrightarrows$ indicates simultaneous updates on two GPUs.
[7] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

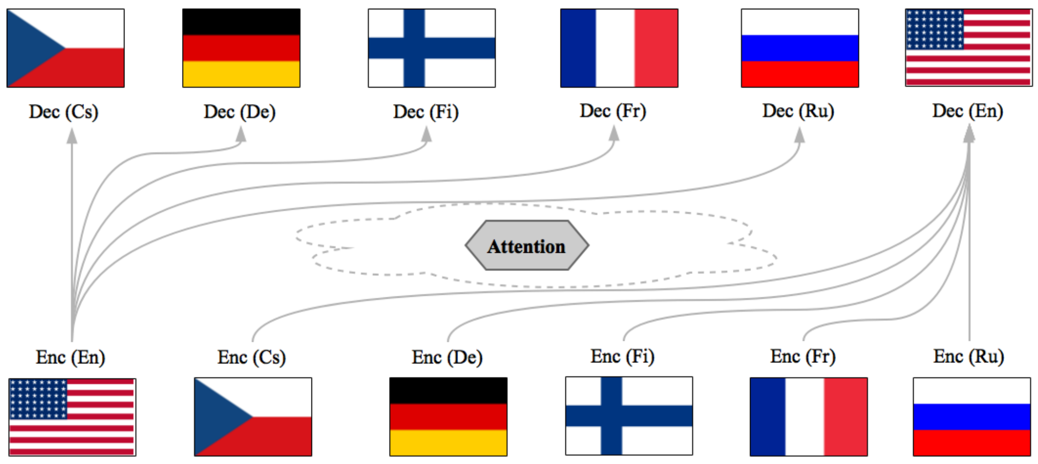


Figure 3.6: Shared Medium for WMT'15 all Languages.

3.4.3 Model Architecture

Each symbol, either source or target, is projected on a 620-dimensional space, the dimensionality of embedding matrix $\mathbf{E}$, d_{emb}. The encoder is a bidirectional recurrent neural network with 1,000 gated recurrent units (GRU) in each direction, and the decoder is a recurrent neural network with also 1,000 GRU's. The decoder's output function g_k from Eq. (2.38) is a feedforward network with 1,000 $\tanh$ hidden units. The dimensionalities of the context vector $\mathbf{h}_t^n$ in Eq. (3.2), the attention-specific context vector $\tilde{\mathbf{h}}_t^n$ in Eq. (3.3) and the attention-specific decoder hidden state $\tilde{\mathbf{h}}_{t-1}^m$ in Eq. (3.5) are all set to 1,200.

We use the same encoder architecture for every source language, and the same type of decoder for every target language. The only difference between the single-pair models and the proposed multilingual ones is the number of encoders N and decoders M. We leave those multilingual translation specific components, such as the ones in Eqs. (3.2)–(3.5), in the single-pair models in order to retain the number of shared parameters constant. This makes the baseline single-pair and multi-way, multilingual models comparable as they will have same number of parameters.[8]

3.4.4 Large-Scale Translation: Ten Language Pair/Directions

We first empirically verify the plausibility of sharing an attention mechanism across multiple language pairs by training one multi-way, multilingual model that has six encoders and six decoders. They correspond to six languages from WMT'15; En, Fr, De, Cs, Ru, Fi $\rightarrow$ En, Fr, De, Cs, Ru, Fi. We use the full corpora for all of them. As a comparison, we also train a single-pair model for each translation pair/direction.

[8] All the details on training along with the code are available at `github.com/nyu-dl/dl4mt-multi`.

Table 3.2: (a) BLEU scores and (b) average log-probabilities for all the ten language pairs from WMT'15. For each translation pair/direction, we bold-face the best score on the test set.

			Fr (39m)		Cs (12m)		De (4.2m)		Ru (2.3m)		Fi (2m)	
		Direction	→ En	En →	→ En	En →	→ En	En →	→ En	En →	→ En	En →
(a) BLEU	Dev	Single	27.22	26.91	21.24	15.9	24.13	20.49	21.04	18.06	13.15	9.59
		Multi	26.09	25.04	21.23	14.42	23.66	19.17	21.48	17.89	12.97	8.92
	Test	Single	27.94	**29.7**	20.32	**13.84**	24	**21.75**	22.44	**19.54**	12.24	**9.23**
		Multi	**28.06**	27.88	**20.57**	13.29	**24.20**	20.59	**23.44**	19.39	**12.61**	8.98
(b) LL	Dev	Single	-50.53	-53.38	-60.69	-69.56	-54.76	-61.21	-60.19	-65.81	-88.44	-91.75
		Multi	-50.6	-56.55	-54.46	-70.76	-54.14	-62.34	-54.09	-63.75	-74.84	-88.02
	Test	Single	-43.34	**-45.07**	-60.03	**-64.34**	-57.81	**-59.55**	-60.65	-60.29	-88.66	-94.23
		Multi	**-42.22**	-46.29	**-54.66**	-64.80	**-53.85**	-60.23	**-54.49**	**-58.63**	**-71.26**	**-88.09**

3.5 Quantitative and Qualitative Analyses

For quantitative analysis, we employ BLEU scores and learning curves. In Table 3.2, we observe that the proposed multilingual model either outperforms or is comparable to the single-pair models for the majority of the all ten pairs/directions considered. This happens in terms of both BLEU and average log-probability. This result is encouraging, considering the fact that there are twice more parameters in the whole set of single-pair models than in the multilingual model.

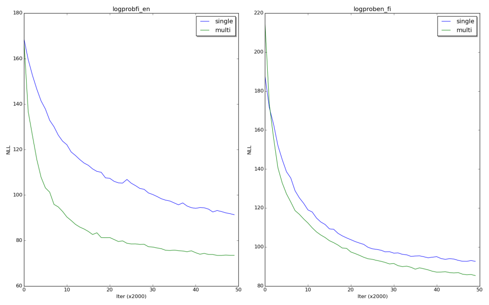
Note that, the performances are often below state-of-the-art neural MT systems, which use large vocabularies, unknown replacements techniques, target-side monolingual corpora and ensembling (see, e.g., [87, 64, 110].) We mainly focused on comparing the proposed model against single-pair models without these techniques in order to carefully control and analyze the effect of having multiple languages. The power of the suggested multi-way multilingual model should be further explored by incorporating these techniques for performance improvements.

It is worth to notice that the benefit of the suggested approach is more apparent when the model translates from a language to English (many-to-one setting). This may be due to the fact that all of the parallel corpora include English as either a source or target language, leading to a better parameter estimation of the English decoder. In the future, a strategy of using a pseudo-parallel corpus to increase the amount of training examples for the decoders of other languages [110] should be investigated to confirm this conjecture.

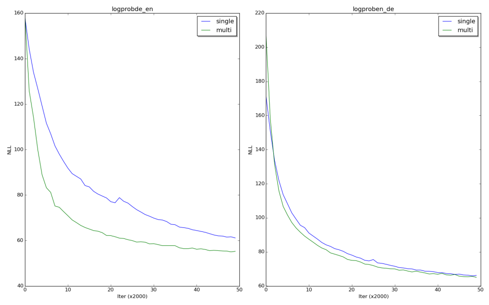
We further noticed that, the proposed multi-way multilingual model converges faster than the single pair models, that are trained individually. In Fig. 3.7, validation set negative-log probabilities are shown for both ten different single-pair models (seq2seq) and a single multi- way multilingual model (multi-seq2seq). It is clear from the figures that, multi-way model converges much faster compared to single pair models.

For qualitative analysis, we use soft alignments.

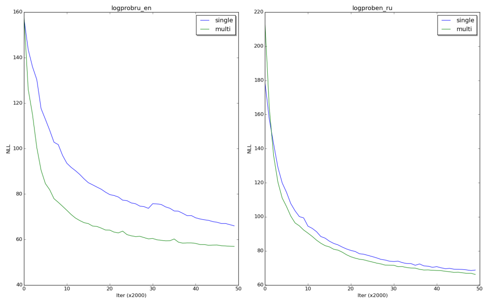
The most distinct characteristic of the proposed model is that a single attention mech-



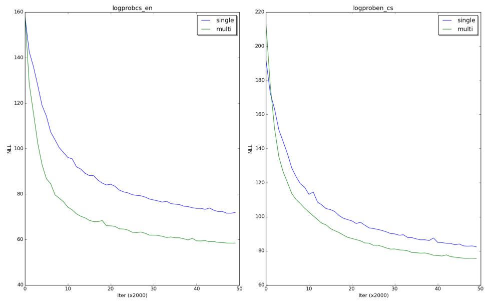
(a) Fi→En (left) En→Fi (right)



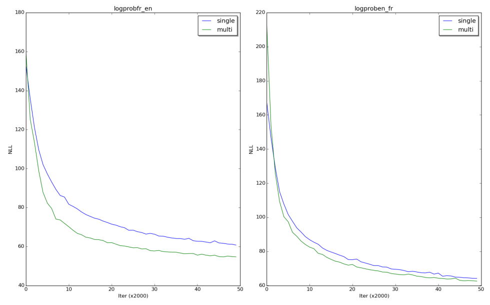
(b) De→En (left) En→De (right)



(c) Ru→En (left) En→Ru (right)



(d) Ru→En (left) En→Ru (right)



(e) Fr→En (left) En→Fr (right)

Figure 3.7: Learning Curve comparison of Single-Pair models (seq2seq - blue curves) and Multi-way model (multi-seq2seq - green curves).

anism is shared across multiple language pairs. More specifically in this experiment, a single attention mechanism is used for 10 translation directions with six different languages. Although the competitive performance by this multi-way, multilingual model suggests that the model is well utilizing the shared attention mechanism, this does not necessarily imply that the shared attention mechanism has learned good alignments across multiple languages.

Here we visualize the attention weights for all ten language pair/directions. We use the following sentence:

The freedom of speech is essential in a democratic society.

We translate this sentence into the five target languages–Fr, Cs, De, Ru and Fi– and also translate the corresponding translations in those languages to English.

In Fig. 3.8 and Fig. 3.5, we present the soft alignments returned by the shared attention mechanism of the multi-way, multilingual model. It is clear that the attention mechanism was able to align between two sentences regardless of which languages they were written in. Note that the attention mechanism used in this experiment is purely based on the content, meaning that it did not exploit the (near-)monotonicity of the true alignment between any language pair. This clearly demonstrates that it is indeed possible to share a single attention mechanism across multiple language pairs.

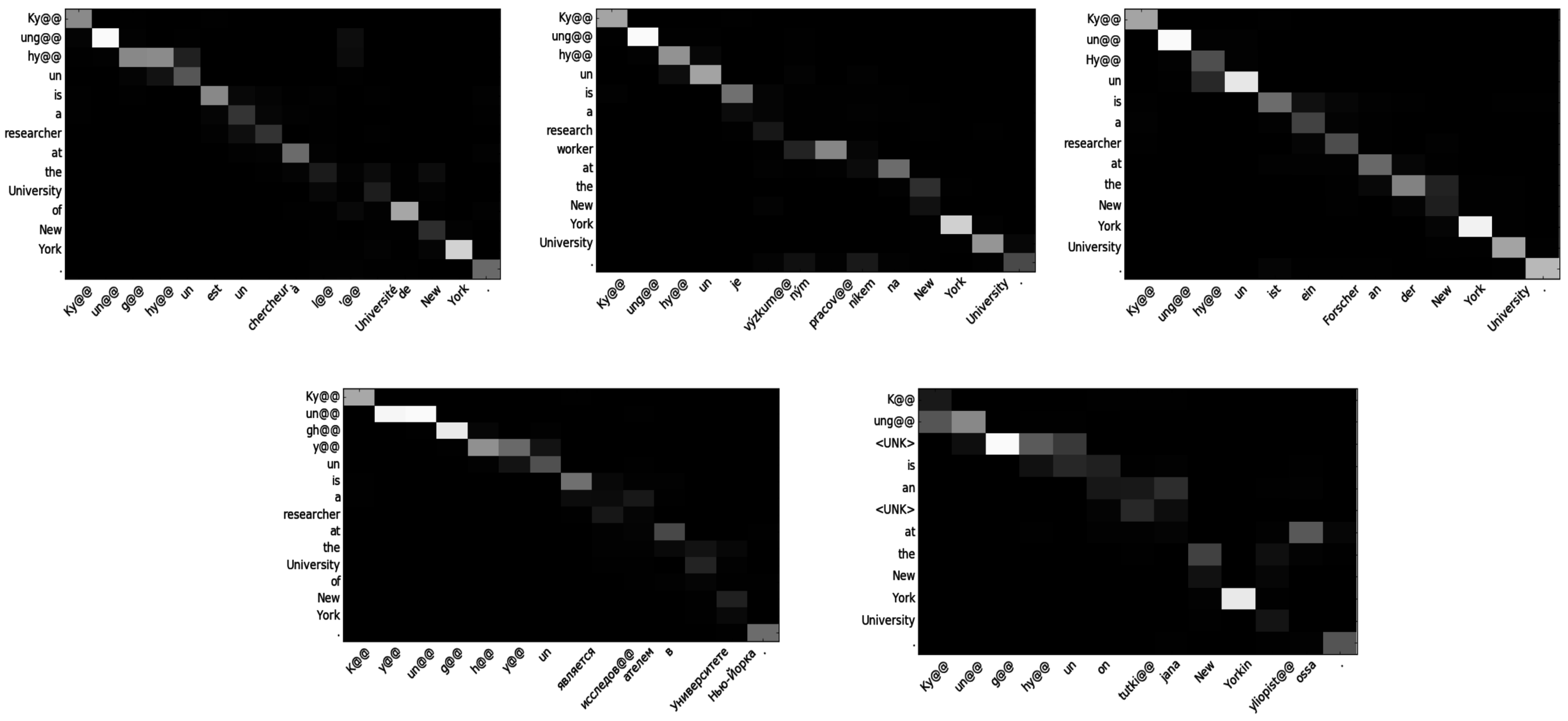


Figure 3.8: Visualization of the alignments found by the multi-way, multilingual model. Best viewed digitally. From top-down and left-to-right ordering, alignments between French → English, Czech → English, German → English, Russian → English and Finnish → English translations, estimated by the shared attention module.

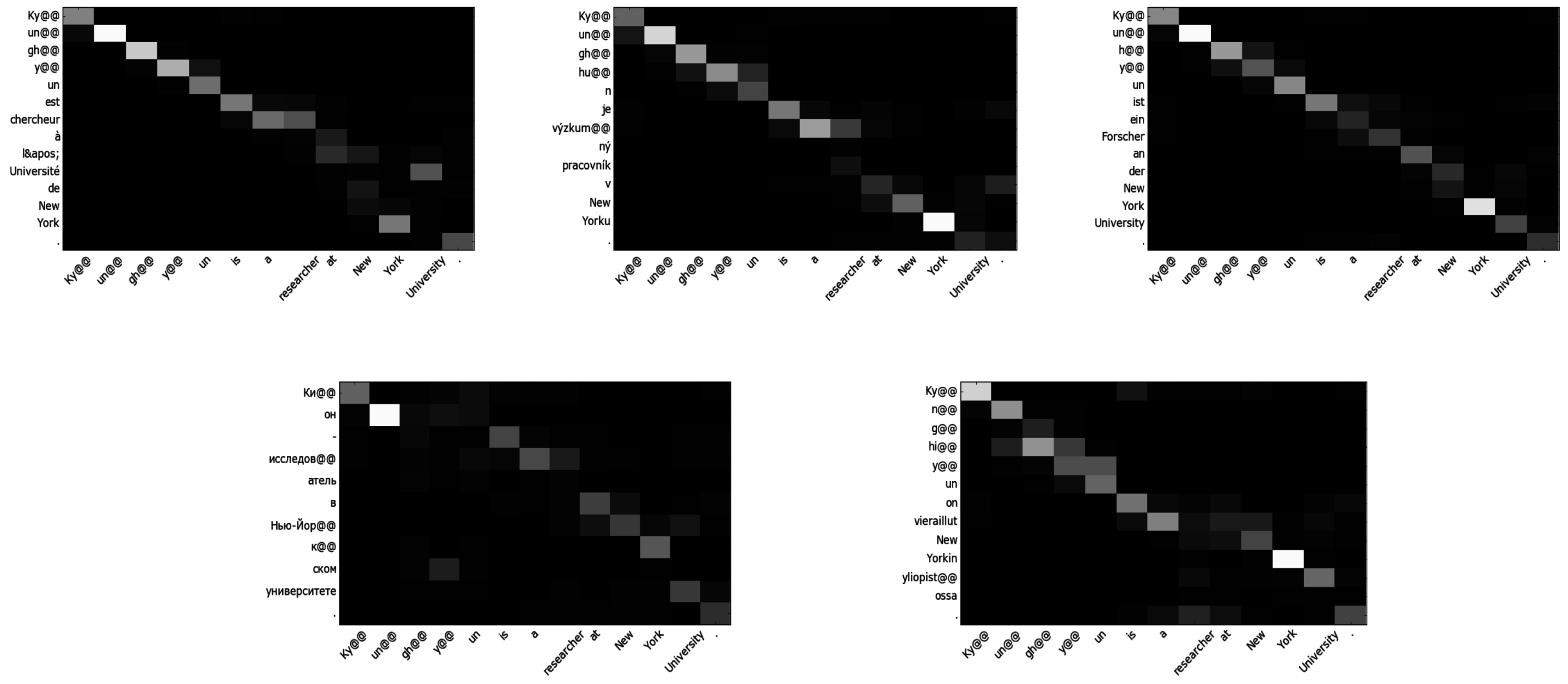


Figure 3.9: Visualization of the alignments found by the multi-way, multilingual model. Best viewed digitally. From top-down and left-to-right ordering, alignments between English → French, English → Czech, English → German, English → Russian and English → Finnish translations, estimated by the shared attention module.

3.6 Conclusion

In this section, we proposed multi-way, multilingual attention-based neural machine translation. The proposed approach allows us to build a single neural network that can handle multiple source and target languages simultaneously. The proposed model is a step forward from the recent works on multilingual neural translation, in the sense that we support attention mechanism, compared to [84] and multi-way translation, compared to [37]. Furthermore, we evaluate the proposed model on large-scale experiments, using the full set of parallel corpora from WMT'15.

We empirically evaluated the proposed model in large-scale experiments using all five languages from WMT'15 with the full set of parallel corpora. In the experimented settings, we observed the benefits of the proposed multilingual neural translation model over having a set of single-pair models.

In general, we observed the larger improvements when translating to English. We conjecture that this is due to a higher availability of English in most parallel corpora, leading to a better parameter estimation of the English decoder. More research on this phenomenon in the future will result in further improvements from using the proposed model. Also, all the other techniques proposed recently, such as large vocabulary tricks and character-level decoding, need to be tried together with the proposed multilingual model to improve the translation quality even further.

Interesting future directions are (1) to use and evaluate the proposed model to investigate the behaviors in very poor data regimes, low-resource translation, (2) to use the proposed model to leverage from multi-view data, multi-source translation (3) to use the proposed model to translate between a language pair not included in a set of training corpus. In the forthcoming Chapters, we investigate the above mentioned future directions one-by-one.

CHAPTER 4

MULTI-SEQUENCE MODELING FOR LOW DATA REGIMES: LOW(UNDER)-RESOURCE TRANSLATION

"It is a capital mistake to theorize before one has data. Insensibly, one begins to twist the facts to suit theories, instead of theories to suit facts."

SHERLOCK HOLMES

The goal of this chapter, is to analyze the behavior of multi-seq2seq architecture when there is not enough parallel data for a given source-target pair.

In the previous Chapter, we proposed the multi-sequence modelling architecture with a shared attention module and presented it's large scale capabilities in the context of neural machine translation. The next question we ask is the following: in which scenario would the proposed multi-way, multilingual neural translation have an advantage over the existing, single-pair model? Specifically, we consider the case of translation between pairs that the amount of training data is inadequate in terms of quantity, namely low-resource language pairs. The experiments in this section show that the proposed multi-way, multilingual model generalizes better than the single-pair translation model, when the amount of available parallel corpus is small. We validate that this is not only due to the increased amount of target-side, monolingual corpus.

In order to test the low-resource translation capabilities, we device two set of experiments. In the first set of experiment, *controlled setting* - Section 4.4.1, we gradually decrease the amount of data used to train a particular source-target pair in multiseq2seq model. This simulated low data help us to control and compare the benefits of using multi-seq2seq architecture over single pair seq2seq models.

In the second set of experiments, *real-world setting* - Section 4.4.2, we increased the difficulty of the low- resource translation problem by attacking real-world low resource translation problems and compare the results with strong statistical machine translation baselines. This is done by gradually augmenting the model with more source and target languages until the best translation quality is achieved on the target low-resource pair. In particular, we observe that the proposed multilingual model outperforms strong conventional statistical machine translation systems on low-resource Turkish-English and very low-resource Uzbek-English by incorporating the resources of other language pairs.

We finally present practical tips for successfully training such models when the sizes of datasets of language pairs vary significantly.

4.1 Low-Resource NMT and Positive Language Transfer

The motivation behind testing our proposed multi-seq2seq model on low resource translation tasks has two folds. Firstly, low-resource seq2seq problems, or more generally, the seq2seq problems where available training data is scarce is very common in real-world problems. Consider the 52% of the entire internet is in English[1], finding diverse datasets spanning multiple languages is a very challenging one.

Secondly, a natural premise of having a multilingual model is to benefit from positive language transfer [98], or more generally knowledge transfer, where applying knowledge from one language (task) to another language (task). Our proposed multi-seq2seq architecture naturally enables and makes it easier, for such knowledge to transfer from one task to another by the notion of shared medium.

For the above mentioned reasons and motivations, we tested the proposed multi-seq2seq architecture on low-resource language translation tasks. Next, we summarize the recent work using again seq2seq for low-resource translation.

[1] `en.wikipedia.org/wiki/Languages_used_on_the_Internet`

4.2 Related Work

Although there exists a few studies on applying seq2seq architectures on low-resource language translation tasks (NMT), to our knowledge there does not exist any work that uses multi-linguality in a connectionist framework. Therefore, in this section we summarize the low-resource translation remedies using seq2seq models.

Sennrich et al. [112] proposed two strategies for low resource NMT by making use of monolingual data (data in one language only - eg. English wikipedia). The first approach, called "dummy source sentences", is to train the decoder of NMT with a sentence from a monolingual corpus while setting all the context vectors $\mathbf{c}_t$ (see Eq. (2.35)) to all- zero vectors. The second approach uses "synthetic source sentences", where each sentence from a target-side monolingual corpus is translated to a synthetic source sentence by a reverse translation model. These pseudo-parallel sentence pairs are mixed into the existing parallel corpus and used to train an neural translation model. Both of these approaches, and especially the latter approach based on synthetic source sentences, were shown to improve the translation quality significantly on Tr-En, En-De and De-En. Their approaches are orthogonal to the ones proposed in this thesis, and can easily be adapted to work on multi-seq2seq architecture. We showcase the use of this approach in the sixty chapter for zero-resource translation task.

Luong et al. [84] proposed a multi-task neural machine translation model. In this multi-task model, it is possible to include multiple source languages as well as target languages. They experimented with a setting where monolingual translation paths (sequence autoencoders) were added to a neural translation model. The experiment revealed that the translation quality improves by jointly training the translation and autoencoding paths. Their work is however limited to a simple encoder-decoder model without the attention mechanism which has proven to be crucial in getting a good neural translation model. The benefits and caveats of this approach is discussed in the previous chapters.

More recently, Zoph et al. [141] demonstrated a transfer learning approach to low-resource translation. They first train a neural machine translation model with a large parallel corpus whose target-side matches that of the target low-resource language pair. Then, this model, or its part, is further finetuned on a small parallel corpus of the target task. They observed significant improvements with a variety of languages, including Hausa, Turkish, Uzbek and Urdu, when the model is pretrained with Fr-En was used as an initial point. Again, it is certainly possible to incorporate the proposed transfer learning approach to multi-seq2seq but since the transfer learning is not our main focus, we leave it as a future direction to be taken.

4.3 Shared Attention Mechanism

In this section we detail the differences and improvements made over the original multi-seq2seq architecture proposed in Chapter 3. We closely follow the model architectures from the earlier experiments (see Sec. 3.4.3) however with one modification to the attention mechanism and one modification to the decoder RNN.

Let us start with the improvements over decoder RNN. As a reminder, in the original multi-seq2seq decoder we proposed using GRU units, and here we improve it by proposing a variant cGRU which has additional internal gating mechanisms and a deeper recurrent transition function. Here we describe cGRU for regular RNNs but it's extension to multi-seq2seq is trivial.

Given a source sequence $(x_1, \ldots, x_{T_x})$ of length T_x and a target sequence $(y_1, \ldots, y_{T_y})$, let $\mathbf{h}_i$ be the annotation of the source symbol at position i, obtained by concatenating the forward and backward encoder RNN hidden states, $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$. A conditional GRU with attention mechanism, cGRU$_{\text{att}}$, uses it's previous hidden state $\mathbf{s}_{j-1}$, the whole set of source annotations $\text{C} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{T_x}\}$ and the previously decoded symbol y_{j-1} in order to update it's hidden state $\mathbf{s}_j$, which is further used to decode symbol y_j at position j,

$$\mathbf{s}_j = \text{cGRU}_{\text{att}}\left(\mathbf{s}_{j-1}, y_{j-1}, \text{C}\right). \tag{4.1}$$

Internals The conditional GRU layer with attention mechanism, cGRU$_{\text{att}}$, consists of three components, two recurrent cells and an attention mechanism ATT in between.

First recurrent cell REC$_1$, combines the previous decoded symbol y_{j-1} and previous hidden state $\mathbf{s}_{j-1}$ in order to generate an intermediate representation $\mathbf{s}'_j$ with the following formulations:

$$\mathbf{s}'_j = \text{REC}_1\left(y_{j-1}, \mathbf{s}_{j-1}\right) = (1 - \mathbf{z}'_j) \odot \underline{\mathbf{s}}'_j + \mathbf{z}'_j \odot \mathbf{s}_{j-1}, \tag{4.2}$$

$$\underline{\mathbf{s}}'_j = \tanh\left(\mathbf{W}'\mathbf{E}[y_{j-1}] + \mathbf{r}'_j \odot (\mathbf{U}'\mathbf{s}_{j-1})\right), \tag{4.3}$$

$$\mathbf{r}'_j = \sigma\left(\mathbf{W}'_r\mathbf{E}[y_{j-1}] + \mathbf{U}'_r\mathbf{s}_{j-1}\right), \tag{4.4}$$

$$\mathbf{z}'_j = \sigma\left(\mathbf{W}'_z\mathbf{E}[y_{j-1}] + \mathbf{U}'_z\mathbf{s}_{j-1}\right), \tag{4.5}$$

where $\mathbf{E}$ is the target word embedding matrix, $\underline{\mathbf{s}}'_j$ is the proposal intermediate representation, $\mathbf{r}'_j$ and $\mathbf{z}'_j$ being the reset and update gate activations. In this formulation, $\mathbf{W}'$, $\mathbf{U}'$, $\mathbf{W}'_r$, $\mathbf{U}'_r$, $\mathbf{W}'_z$, $\mathbf{U}'_z$ are trained model parameters[2] tanh and σ are hyperbolic tangent and logistic sigmoid activation functions respectively.

The attention mechanism ATT, inputs the entire context set C along with intermediate hidden state $\mathbf{s}'_j$ in order to compute the context vector $\mathbf{c}_j$ as follows:

$$\mathbf{c}_j = \text{ATT}\left(\text{C}, \mathbf{s}'_j\right) = \sum_{i}^{T_x} \alpha_{ij}\mathbf{h}_i, \tag{4.6}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{Tx}\exp(e_{kj})}, \tag{4.7}$$

$$e_{ij} = \mathbf{v}_a^{\intercal}\tanh\left(\mathbf{U}_a\mathbf{s}_j^{(1)} + \mathbf{W}_a\mathbf{h}_i\right), \tag{4.8}$$

where α_{ij} is the normalized alignment weight between source symbol at position i and target symbol at position j and $\mathbf{v}_a, \mathbf{U}_a, \mathbf{W}_a$ are the trained model parameters.

Finally, the second recurrent cell REC$_2$, generates $\mathbf{s}_j$, the hidden state of the cGRU$_{\text{att}}$, by looking at intermediate representation $\mathbf{s}'_j$ and context vector $\mathbf{c}_j$ with the following formulations:

[2] All the biases are omitted for simplicity.

$$\mathbf{s}_j = \text{REC}_2\left(\mathbf{s}'_j, \mathbf{c}_j\right) = (1 - \mathbf{z}_j) \odot \underline{\mathbf{s}}_j + \mathbf{z}_j \odot \mathbf{s}'_j, \tag{4.9}$$

$$\underline{\mathbf{s}}_j = \text{tanh}\left(\mathbf{W}\mathbf{c}_j + \mathbf{r}_j \odot (\mathbf{U}\mathbf{s}'_j)\right), \tag{4.10}$$

$$\mathbf{r}_j = \sigma\left(\mathbf{W}_r\mathbf{c}_j + \mathbf{U}_r\mathbf{s}'_j\right), \tag{4.11}$$

$$\mathbf{z}_j = \sigma\left(\mathbf{W}_z\mathbf{c}_j + \mathbf{U}_z\mathbf{s}'_j\right), \tag{4.12}$$

similarly, $\underline{\mathbf{s}}_j$ being the proposal hidden state, $\mathbf{r}_j$ and $\mathbf{z}_j$ being the reset and update gate activations with the trained model parameters $\mathbf{W}, \mathbf{U}, \mathbf{W}_r, \mathbf{U}_r, \mathbf{W}_z, \mathbf{U}_z$.

After defining the intrinsics of cGRU let us extend it to multi-seq2seq. First instead of a simple feedforward network for φ^m_{att} in Eq. (3.5), we use the following formula:

$$\tilde{\mathbf{z}}^m_{t-1} = \varphi^m_{\text{att}}\left(\tilde{y}^m_{t-1}, \mathbf{z}^m_{t-1}\right) = (1 - \mathbf{u}'_t) \odot \underline{\mathbf{s}}'_t + \mathbf{u}'_t \odot \mathbf{z}^m_{t-1}, \tag{4.13}$$

$$\underline{\mathbf{s}}'_t = \tanh\left(\mathbf{W}'\mathbf{E}^m_y[\tilde{y}^m_{t-1}] + \mathbf{r}'_j \odot (\mathbf{U}'\mathbf{z}^m_{t-1})\right), \tag{4.14}$$

$$\mathbf{r}'_t = \sigma\left(\mathbf{W}'_r\mathbf{E}^m_y[\tilde{y}^m_{t-1}] + \mathbf{U}'_r\mathbf{z}^m_{t-1}\right), \tag{4.15}$$

$$\mathbf{u}'_t = \sigma\left(\mathbf{W}'_u\mathbf{E}^m_y[\tilde{y}^m_{t-1}] + \mathbf{U}'_u\mathbf{z}^m_{t-1}\right), \tag{4.16}$$

where $\underline{\mathbf{s}}'_t$, $\mathbf{r}'_t$ and $\mathbf{u}'_t$ are the proposal intermediate representation, the reset gate and the update gate respectively. $\mathbf{W}'$, $\mathbf{U}'$, $\mathbf{W}'_r$, $\mathbf{U}'_r$, $\mathbf{W}'_u$ and $\mathbf{U}'_u$ are model parameters.[3] $\tanh$ and σ are respectively hyperbolic tangent and logistic sigmoid activation functions. Furthermore, instead of Eq. (3.6), we update the hidden state of the decoder by

$$\mathbf{z}_t = \Psi_{\text{dec}}\left(\tilde{\mathbf{z}}^m_{t-1}, f^m_{\text{adp}}(\mathbf{c}^m_t)\right). \tag{4.17}$$

One last change over the original multi-seq2seq architecture is the indicator vectors. Indicator vectors are one-hot vectors that encode the source or target language index (id) into a one-dimensional binary vector, having only one non- zero element, at the position of source or target language, within the whole set of languages considered. As an example, let us assume we have three different source languages, and we the provided data to multi-seq2seq system at time t is the second language. We can construct the indicator vector $\mathbf{s} = (s_1, s_2, s_3)$ such that $s_i = 0$ if $i \notin 2$ and $s_i = 1$ if $i = 2$. With the help of indicator vectors, we provide an additional information to the decoder, which language it is translating. Please see Fig. 4.1 for the graphical illustration of the proposed changes over Fig. 3.3.

[3] We omit the biases to make the equations less cluttered.

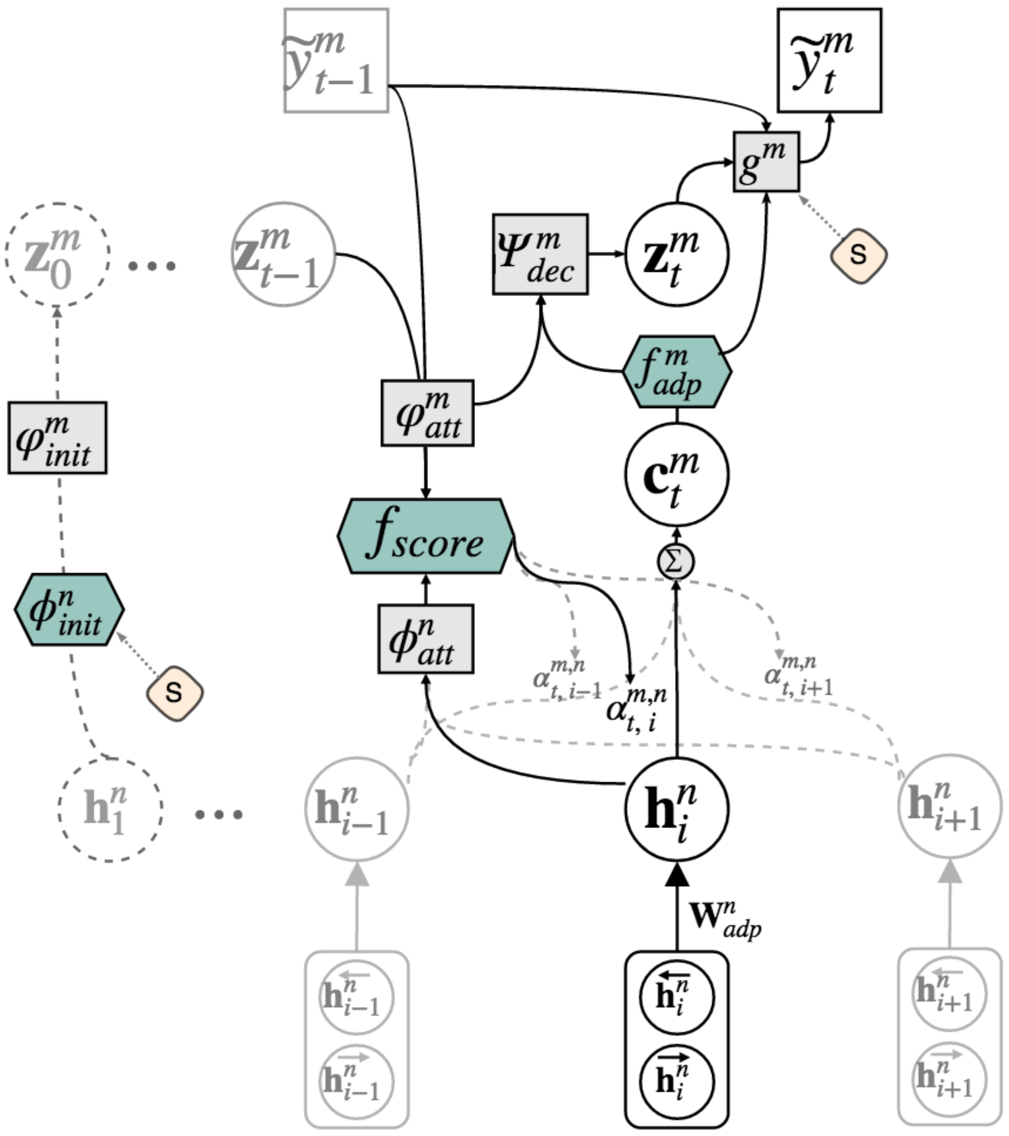


Figure 4.1: One step of the improved multi-way. multilingual Neural Machine Translation model, for the n-th encoder and the m-th decoder at time step t of the decoder. Shaded boxes are parametric functions and square boxes represent intermediate variables of the model. Initializer network is also illustrated as the left-most network with dashed boxes. All the shared components are drawn with diamond boxes highlighted in green. Orange boxes are indicator vectors. See Sec. 4.3 for details.

4.4 Experiments

In this section we provide two set of experiments for low resource translation. First we test the proposed multi-way, multilingual model in a controlled setup, second going further and testing the model on real world low resource translation tasks.

4.4.1 Controlled Settings: Fi-En, De-En, En-De

4.4.1.1 Setting

We investigate the effect of the proposed multi-way, multilingual model on low-resource language-pair translation under the same setting however with a smaller number of languages. Among the six languages from WMT'15, we choose English(En), German(De) and Finnish(Fi) as source languages, and En and De as target languages. We control the amount of the parallel corpus of each pair out of three to be 5%, 10%, 20% and 40% of the original corpus. In other words, we train four models with different sizes of parallel corpus for each language pair (En-De, De-En and Fi-En.)

As a baseline, we train a single-pair model for each multi-way, multilingual model. We further finetune the single-pair model to incorporate the target-side monolingual corpus consisting of all the target side text from the other language pairs (e.g., when a single-pair model was trained on Fi-En, the target-side monolingual corpus consists of the target sides from De-En.) This is done by the recently proposed deep fusion [52]. The latter is included to tell whether any improvement from the multilingual model is simply due to the increased amount of target-side monolingual corpus.

4.4.1.2 Result and Analysis

It is clear from Table 4.1 that the proposed model (Multi) outperforms the single-pair one (Single) in all the cases. This is true even when the single-pair model is strengthened with a target-side monolingual corpus (Single+DF). This suggests that the benefit of generalization from having multiple languages goes beyond that of

Table 4.1: BLEU scores where the target pair's parallel corpus is constrained to be 5%, 10%, 20% and 40% of the original size. We report the BLEU scores on the development and test sets (separated by /) by the single-pair model (Single), the single-pair model with monolingual corpus (Single+DF) and the proposed multi-way, multilingual model (Multi).

	Size	Single	Single+DF	Multi
Fi→En	100k	5.06/3.96	4.98/3.99	6.2/**5.17**
	200k	7.1/6.16	7.21/6.17	8.84/**7.53**
	400k	9.11/7.85	9.31/8.18	11.09/**9.98**
	800k	11.08/9.96	11.59/10.15	12.73/**11.28**
De→En	210k	14.27/13.2	14.65/13.88	16.96/**16.26**
	420k	18.32/17.32	18.51/17.62	19.81/**19.63**
	840k	21/19.93	21.69/20.75	22.17/**21.93**
	1.68m	23.38/23.01	23.33/22.86	23.86/**23.52**
En→De	210k	11.44/11.57	11.71/11.16	12.63/**12.68**
	420k	14.28/14.25	14.88/15.05	15.01/**15.67**
	840k	17.09/17.44	17.21/17.88	17.33/**18.14**
	1.68m	19.09/19.6	19.36/20.13	19.23/**20.59**

simply having more target-side monolingual corpus. The performance gap grows as the size of target parallel corpus decreases.

4.4.2 Real World Settings: Tr-En, Uz-En

The last experiment in Sec. 4.4.1 suggests that adding more language pairs works as a regularizer for low-resource translation and improves its translation quality. This means that we can use the proposed multi-way, multilingual model as an effective approach to low-resource language translation by tweaking the training procedure to focus solely on the performance on a single target task of low-resource translation rather than the average performance over all the language pairs (as was done in the previous experiments). We test this aspect of the proposed model by evaluating it on Turkish-English (Tr-En) and Uzbek-English (Uz-En) translation tasks with auxiliary language pairs including Spanish-English (Es-En) and French- English (En-Fr).

4.4.3 Data Preparation

4.4.3.1 Corpora

Tr-En We concatenate the following Tr-En parallel corpora to form a single Tr-En parallel corpus:

- LDC2014E115: 54k sentence pairs
- WIT TED Talks [18]: 228k sentence pairs
- SETimes2[4]: 207k sentence pairs
- OpenSubtitles[5]: 211k sentence pairs
- Tatoeba Corpus[6]: 156k sentence pairs

We tokenize the both sides of the corpora using the script from Moses. Development and test sets are randomly sampled from LDC2014E115 dataset before training.

[4] http://opus.lingfil.uu.se/SETIMES2.php
[5] http://opus.lingfil.uu.se/OpenSubtitles.php
[6] http://opus.lingfil.uu.se/Tatoeba.php

Uz-En We use LDC2014E112 which consists of 74k sentence pairs. Similarly, development and test sets are randomly sampled from training set before training.

Es-En and Fr-En We use a total of 34.71m parallel sentences for Es-En auxiliary data, which is a combination of LDC, Europarl, News-Commentary, Open-Subtitles 2013, UN, and IBM internal technical parallel corpora. For Fr-En auxiliary data, we used a combination of LDC, Europarl, News-Commentary, News-Crawl, UN, Gigaword, Hansard, Reuters and IBM internal technical parallel corpora which adds up to 65.77m sentence pairs.

4.4.3.2 Preprocessing

All the text is tokenized using the tokenizer script from Moses. We then replace all the special tokens, such as numbers, dates and urls, with special markers, which will be used during translation and replaced to original, corresponding tokens after decoding. Afterwards we build a dictionary for each language using BPE [112].

For each language, other than English and Turkish, we use only the parallel corpora to extract up to 40k BPE subwords and form a vocabulary. In the case of English, we use the English side of the parallel corpora for Tr-En and Uz-En in addition to the English Gigaword (LDC2011T07) to extract up to 40k BPE subwords. For Turkish, we use the Turkish side of the parallel corpora for Tr-En in addition to the Turkish Wikipedia,[7] again, to extract up to 40k BPE subwords.

The detailed statistics of the final corpora are presented in Table 4.2.

4.4.4 Model Architecture

Multi-Seq2Seq We closely follow the model hyperparameters in the previous Chapter and employed the enhancements explained in Section 4.3.

[7] http://dumps.wikimedia.org/trwiki/20131011

Table 4.2: Statistics of the parallel corpora used in Uz-En and Tr-En experiments. Symbols are BPE-based sub-words.

	# Symbols		# Sentence		
	# En	Other	Train	Dev	Test
En-Uz	1.361m	1.186m	73.66k	948	882
En-Tr	13.17m	12.43m	784.65k	862	940
En-Es	908.1m	924.9m	34.71m	3003	3000
En-Fr	1.837b	1.911b	65.77m	3003	3000

Conventional SMT In order to better evaluate the proposed model, we build conventional tree-to-string SMT models, based on [137] for both Tr-En and Uz-En and use them as our SMT baselines. We used the same data to train these models. They use the standard set of SMT features: forward and backward count-based probabilities, forward and backward lexical probabilities, word and rule penalty and a language model probability. The language model was trained to be a 5-gram model using a large English monolingual corpus including Gigaword and English-side of the parallel data from WMT'15 with a total of ~5B words. The parameters are further tuned with PRO [58] to minimize T-B metric. Notice that, these baseline models offer strong baseline numbers for the two language pairs and employ large amounts of monolingual data.

4.4.5 Training and Evaluation

4.4.5.1 Training Setting

We observed while inspecting the models trained during the earlier experiments that training converges at vastly different rates and they approximately correlate with the size of training corpora. A language pair with a smaller training corpus converged and began to overfit faster than another with a larger corpus. In order to alleviate this behavior, we use varying minibatch sizes for each language pair based on the respective training data size. A minibatch consisting of sentence pairs from a high-resource language pair will be larger than that from a low-resource language pair. This ensures that the approximately same number of updates is needed to make a full pass on a whole training corpus for each language pair. We call this strategy a *balanced batch* strategy. We explain the necessity and benefits of using *balanced batch* strategy shortly in Sec. 4.6.

Other than employing the balanced batch strategy, we closely follow the training procedure in Sec. 3.4.2 from the previous set of experiments.

4.4.5.2 Evaluation Metric

BLEU tends to favor longer translations because of the brevity penalty. In the context of this thesis employing shared attention, we are interested to evaluate its effectiveness in generating translations of optimal length. We thus use another well-known evaluation metric, T-B, which combines BLEU and TER, which is defined as:

$$\text{T-B} = \frac{\text{TER} - \text{BLEU}}{2},$$

where TER is a translation edit rate [117]. Note that, T-B is an error metric and hence smaller numbers correspond to better translations and *vice versa*. Consequently training in this section is early stopped based on the T-B score on the development set. However for the sake of comparison we also provide BLEU alongside.

4.5 Analysis

4.5.1 Turkish-English: Result and Analysis

We experimented with different configurations which are shown in Table 4.3 with the translation quality for each configuration. We make a number of observations.

First, there is a clear improvement even when simply one source language was added (Turkish $\rightarrow$ English + Spanish $\rightarrow$ English), compared to the single-pair model (Turkish $\rightarrow$ English.) However, adding another source language (Turkish $\rightarrow$ English + Spanish $\rightarrow$ English + French $\rightarrow$ English) did not improve as much. We conjecture that this is due to the similarity between Spanish and French. The addition of a new language contributes to the generalization performance if there is new knowledge.

Table 4.3: Results on Turkish → English Translation. We report TB↓ (BLEU↑) score in each cell. ⋆ Ensemble of four models. † Ensemble of two models. ‡ Ensemble of three Turkish(Tr) → English(En) + Spanish(Es) → English + French(Fr) → English and three Turkish → English + Spanish → English + English → Turkish + English → Spanish models. Best scores written in bold.

Added Pairs	Development		Test	
	Single	Ensemble	Single	Ensemble
Tr→En	31.99 (14.21)	28.00 (18.34)⋆	28.58 (17.28)	24.27 (20.83)⋆
+ Es→En	29.91 (16.00)	27.15 (18.36)†	27.49 (17.75)	23.94 (20.89)†
+ Es→En + Fr→En	29.05 (16.18)	26.54 (18.69)†	26.77 (18.13)	24.00 (20.90)†
+ Es→En + En→Tr + En→Es	29.74 (16.28)	28.03 (18.32)†	**26.30 (18.66)**	24.28 (20.23)†
MLNMT Ensemble	25.42 (20.00)‡		**21.78 (22.56)**‡	
Conventional SMT	26.55 (14.44)		23.42 (18.00)	

If the additional language (French) is closely related to one of the previously added languages (Spanish), there is less to be gained.

Second, the best single-model performance was achieved when we made the model multi-way (Turkish → English + Spanish → English + English → Turkish + English → Spanish), utilizing most of the parallel corpora available. Adding target languages was more beneficial than adding a source language, which supports the multi-way, multilingual translation proposed in this thesis.

Lastly, we see that the ensemble of multiple models with two different configurations[8] results in a better translation model than the strong, conventional statistical machine translation model (Conventional SMT). This is an encouraging sign for the proposed multi-way, multilingual neural machine translation system in the task of low-resource language translation, especially considering that the current neural models did not utilize any of the monolingual corpora.

Table 4.4: Results on Uzbek→English Translation. We report TB↓ (BLEU↑) score in each cell. ⋆ Ensemble of four models. † Ensemble of three models. ‡ Ensemble of three Uzbek → English + Turkish → English + Spanish → English and three Uzbek → English + Turkish → English + English → Uzbek + English → Turkish models. Best scores written in bold.

Added Pairs	Development		Test	
	Single	Ensemble	Single	Ensemble
Uz→En	41.58 (6.63)	38.98 (8.21)⋆	42.56 (6.45)	38.56 (8.81)⋆
+ Tr→En	37.13 (8.68)	35.83 (10.79)†	36.79 (9.34)	34.49 (11.69)†
+ Tr→En + Es→En	36.42 (9.55)	34.28 (11.96)†	**35.39 (10.34)**	33.20 (12.33)†
+ Tr→En + En→Uz + En→Tr	36.69 (8.93)	34.90 (10.57)†	36.28 (9.41)	33.65 (11.30)†
MLNMT Ensemble	33.40 (12.17)‡		**31.77 (12.99)**‡	
Conventional SMT	32.33 (11.50)		32.38 (9.37)	

4.5.2 Uzbek-English: Result and Analysis

In the case of Uzbek(Uz)-English(En), we observe a similar trend with Turkish(Tr)-English(En). One noticeable difference is that the addition of a single language pair, Tr→En (Uzbek → English + Turkish → English) shows a substantial improvement over the single-pair model (Uzbek → English). This is likely due to the similarity between Uzbek and Turkish (both of them are Turkic languages).

Unlike Turkish-English, we observed that having two additional source languages (Uzbek → English + Turkish → English + Spanish → English) was better than the multi-way model (Turkish → English + English → Uzbek + English → Turkish). We conjecture that this is due to the small training corpus of Tr-En, compared to Es-En. This observation indirectly confirms that it is important to have high-resource language pairs to fully exploit the capacity of the proposed multi-way, multilingual model.[9]

Similarly to Tr-En, the ensemble of the multilingual neural translation models outperformed the conventional statistical translation system. Especially, in terms of BLEU,

[8] An ensemble of three Uzbek → English + Turkish → English + Spanish → English and three Uzbek → English + Turkish → English + English → Uzbek + English → Turkish models

[9] Note that, in Section 4.5.1, two high resource, distant languages (to Turkish) are incrementally added. The second high resource language did not improve the performance, either as it did not introduce any new information or the decoder did not have enough capacity to leverage more data.

we see more than +3 improvement on both Tr-En and Uz-En.

4.6 Notes on Early-Stopping, Dataset Balancing and Scheduling

A major problem encountered in almost any multi-task learning setup is the imbalance between datasets, both in terms of quality and quantity. When we assume all tasks are equally difficult, this imbalance between available data for each task, causes a series of complications.

Let us describe the probable problems with a simplified scenario. Given three seq2seq tasks with equal difficulty, we name individual seq2seq problems as Pair-1, Pair-2 and Pair-3. Further, assume that the associated parallel data for each task is proportional to: Pair-1 has 10% of the total amount of data that Pair-3 has, and Pair-2 has 50% of the data that Pair-3 has.

In order to train a multi-seq2seq model given three tasks with their associated parallel data, we first need to decide an update schedule which determines the order and frequency of examples that are provided to model during training. Following the training routines from Sec. 3.4.2, the simplest scheduling is using equal sized mini-batches for all tasks, and providing the mini-batches in a round-robin fashion, performing one update per task at each turn.

The problem we encounter is related with over-fitting and early stopping [44]. During training, we need to stop training pick a the model to be used at test time. However, it is unclear when to stop training for a model that learns how to solve multiple tasks. Given the varying sizes of the datasets for our three pairs, and the update scheduling above, it is obvious that, at a given iteration of training, the model will pass through the dataset of Pair-1 ten times more compared to the dataset of Pair-3. And at the same iteration, the model will pass over the dataset of Pair-2 two times more, again compared to Pair-3. Practically, model will complete more epochs for the tasks that have less amount of data, Pair-1 and Pair-2, as a result, will start over-fitting way earlier on tasks that have less amount of data. And no matter when we early stop, the model will favor one task, and will not generalize for the other tasks, as it generalizes for the task we choose. This behavior is sketched in Fig. 4.2a, which we call *bad* *scenario*. More plausible scenario is, trained model becoming equally an expert on all three tasks, at a given time during training. In the second scenario, the loss curves will be expected to align in a way that, early stopping point of the model will generalize to test cases of all tasks equally likely. This plausible behavior is sketched in Fig. 4.2b and we call it the *good scenario*. Note that, the generalization performance of the end model that is picked differs purely on the early stopping point, and independent of the model architecture used.

In order to observe models following the *good scenario*, the learning process of the different tasks should be aligned, ensuring that the model that is good for one tasks will not lack generalization due to over-fitting on the other tasks. The solution is quite simple and intuitive. At a given time in the training process, we ensure the model to pass over the datasets of each tasks equal times. For instance, the model at iteration 550 of Fig. 4.2b, is expected to finish equal amount of epochs on the datasets of all three tasks, Pair-1, Pair-2 and Pair-3. And then we can decide our early stopping point to be iteration 550, at which the generalization performance of the model is balanced for all three tasks with respect to their dataset sizes.

When we follow a round-robin update schedule, we can simply adjust the batch sizes of tasks that have smaller datasets, to finish one epoch at the same time. In our example, *good scenario* can be achieved by using mini-batches that are ten times smaller for Pair-1 compared to Pair-3 and two times smaller for Pair-2 compared to again Pair-3. We call the above described heuristic *balanced-batch* strategy.

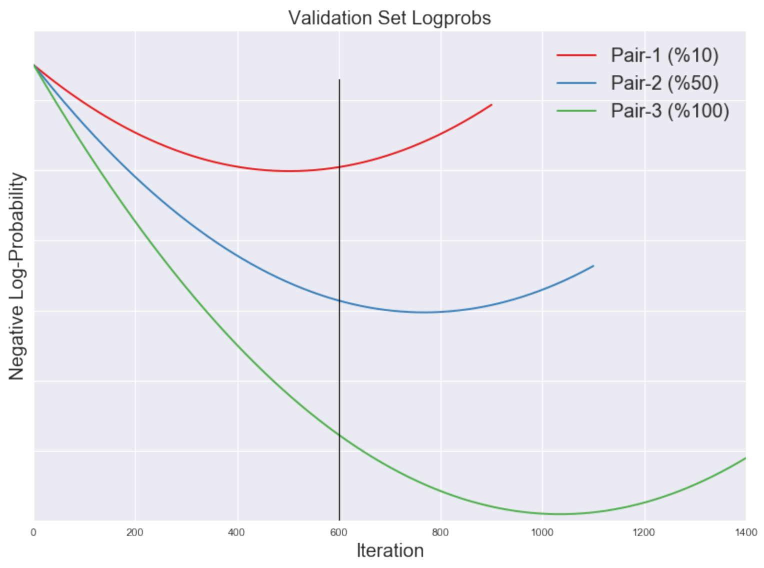
Now let us investigate the effect of using the proposed *balanced-batch* strategy in our real world setting, namely, Tr-En and Uz-En neural machine translation. Figure **??** illustrates the validation set negative log-likelihoods of a multi-way multilingual model that is trained for four pairs, Uz$\rightarrow$En (Pair-1), En$\rightarrow$Uz (Pair-2), Tr$\rightarrow$En (Pair-3) and En$\rightarrow$Tr (Pair-4). The early stopping point it chosen to be 40 (80k iterations) and the model at iteration 80k achieves TB scores of 38.85, 57.19, 39.52 and 51.86 for four pairs respectively. When we apply the *balanced-batch* strategy, effectively using mini-batches ten times smaller for Uz$\rightarrow$En and En$\rightarrow$Uz, the early stopping point shifts further in time dramatically, and let's us to chose the model at iteration 260k that achieves TB scores of 36.69, 55.32, 32.76 and 41.31 respectively.

Table 4.5: Balanced-Batch Strategy Improvements, T-B Scores (BLEU Scores)

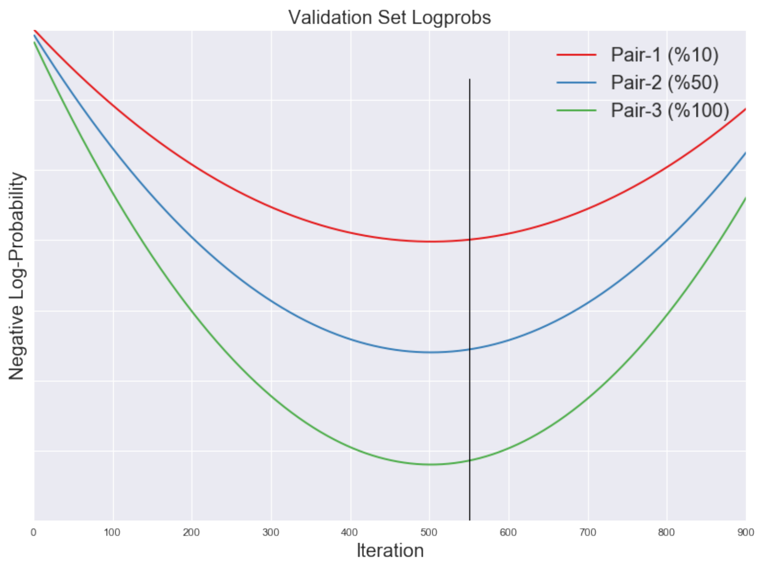
Pair	w/o Balanced-Batch	with Balanced-Batch	Improvement
Uz→En	38.85 (8.47)	36.69 (8.93)	2.16 (0.46)
En→Uz	57.19 (4.09)	55.32 (4.31)	1.87 (0.22)
Tr→En	39.52 (9.57)	32.76 (13.91)	6.76 (4.34)
En→Tr	51.86 (4.83)	41.31 (9.13)	10.55 (4.3)

As can be seen from Fig. 4.3, the early stopping point chosen without *balanced-batch* strategy heavily favors Uz→En and En→Uz pairs and performs poorly on Tr→En and En→Tr pairs. However, using the *balanced-batch* strategy allows us to early stop at an iteration of the model that performs better on all four pairs.

The improvements we observe for Tr→En and En→Tr pairs are as large as 10 TB points (4 BLEU points), see Table 4.5. Surprisingly, *balanced-batch* strategy also improves Uz→En and En→Uz pairs. We argue that,using the proposed strategy delivers an additional regularization benefit. Since we use smaller batches for low-resource language pairs, the variance of the computed gradients for these pairs increase (also mini-batch gradients become more noisy),serving as a regularizer for low-resource pairs [94, 20].

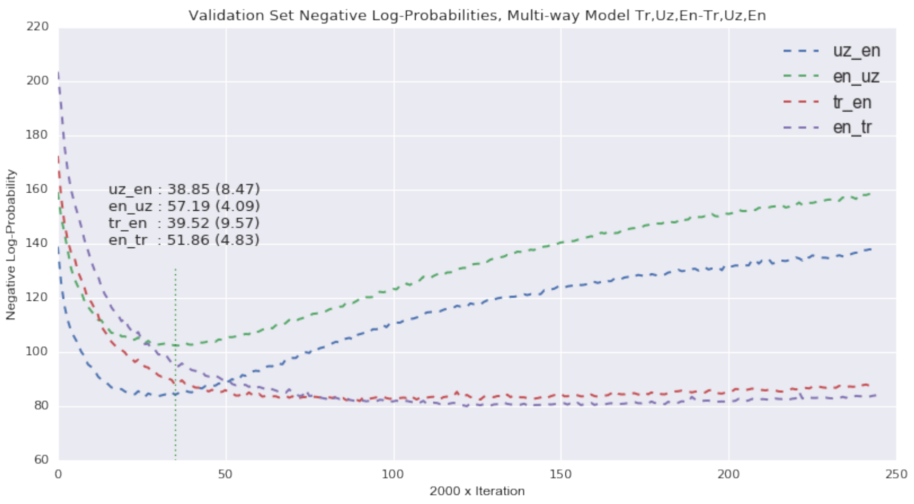


(a) Bad Scenario

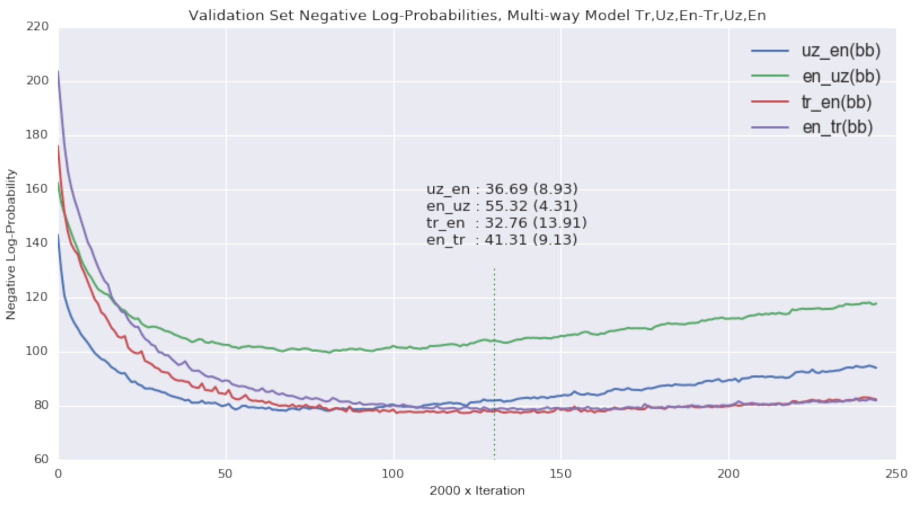


(b) Good Scenario

Figure 4.2: Balanced batches toy.



(a) Training without Balanced-batch Strategy



(b) Training with Balanced-batch Strategy

Figure 4.3: Negative Log-Likelihood plots on Validation Set using Balanced-batch Strategy (b) and not (a).

4.7 Conclusion

In this Chapter, we analyzed the behavior of proposed multi-way multilingual architecture for low-data regimes. We experimented positive language transfer capability of the *shared medium* hypothesis on two settings. In the first setting, we gradually decreased the amount of data that is being used to train one particular pair in a multi-way setup on three different language pairs, namely, Finnish-English, German-English and English-German. We compared the performance of the proposed model with models that are trained for a single pair only and models that use additional monolingual data for target-side language modelling. In the second setting, we experimented two real-world low resource language translation performance of the proposed architecture and compared it again with the single pair NMT models along with very strong conventional Phrase-Based Machine Translation systems.

In both of the settings, we observed the benefits of the proposed multilingual neural translation model over having a set of single-pair models. The improvement was especially clear in the cases when the amount of data is small.

Motivated by this observation, we tested the proposed model specifically to target low-resource language translation with the target tasks Uzbek-English and Turkish-English. In these experiments, we confirmed the effectiveness of the proposed multi-way, multilingual model by showing that it outperforms the strong conventional machine translation system based on a hybrid tree-to-string statistical model.

CHAPTER 5

MULTI-SEQUENCE MODELING FOR MULTI-VIEW DATA: MULTI-SOURCE TRANSLATION

"It is a narrow mind which cannot look at a subject from various points of view."

MARY ANNE EVANS

In the previous chapters, we first proposed the multi-sequence modelling architecture with a shared attention module and presented it's large scale capabilities in the context of neural machine translation in Chapter 3, then in Chapter 4, we presented it's transfer learning capabilities across language pairs, when the amount of available data is limited for a subset of translation pairs. In all of the previous experiments, we assumed that, an example in a sequence to sequence mapping problem is a 2-tuple $(\mathbf{x}, \mathbf{y})$ pair, and used bi-text both during training and test time. In other words, proposed multi-sequence modelling architecture is used to perform one-to-one mappings, from a source sequence $\mathbf{x}$ to a target sequence $\mathbf{y}$.

One very interesting and natural use case of the multi-sequence modelling architecture is the mapping problems when there are more than one views (or sources) of the same data point (example)[132]. This chapter investigates the possible extensions of the proposed multi-sequence modelling architecture, when multi-view and multi-source data is available.

As described in Section 2.3 in a multi-view dataset (multi-text or multi-way parallel data) each example consists of a tuple of n data points where $n >= 2$. We can develop architectures that can exploit the complementary and diverse information across multiple-views, practically benchmark multi-sequence modelling architectures on many-to-one mapping problems (multi-source translation).

In this Chapter, we first frame many-to-one mapping problem into multi-sequence mapping architecture. Then we describe different scenarios of many-to-one mapping problem according to the availability of multi-view data, either during training or test time. Next, we propose novel decoding strategies that exploit model ensembles in multi-sequence mapping architecture or manifold hypothesis of the attention module, the *shared medium*.

Finally, we present empirical evidence of how our proposed multi-sequence mapping architecture makes use of multi-view data, application to large-scale multi-source Neural Machine Translation between French, Spanish and English. In doing so, we begin by studying different translation strategies available in the multi-way, multilingual model in Sec.5.4. The strategies include a usual one-to-one translation as well as variants of many-to-one translation for multi-source translation [140]. We empirically show that the many-to-one strategies significantly outperform the one-to-one strategy.

5.1 Many-to-one Mapping with seq2seq and Related Work

As described and summarized by the previous chapters, neural machine translation [42, 70, 118, 24] has proven to be a platform for new opportunities in machine translation research. Rather than word-level translation with language-specific preprocessing, neural machine translation has found to work well with statistically segmented subword sequences as well as sequences of characters [28, 85, 112, 80]. Also, recent works show that neural machine translation provides a seamless way to incorporate multiple modalities other than natural language text in translation [84, 12]. Furthermore, neural machine translation has been found to translate between multiple languages, achieving better translation quality by exploiting positive language transfer [37, 39, 140].

Before going into the details and related work, let us first describe possible scenarios that can be encountered in a multi-seq2seq architecture according to the availability

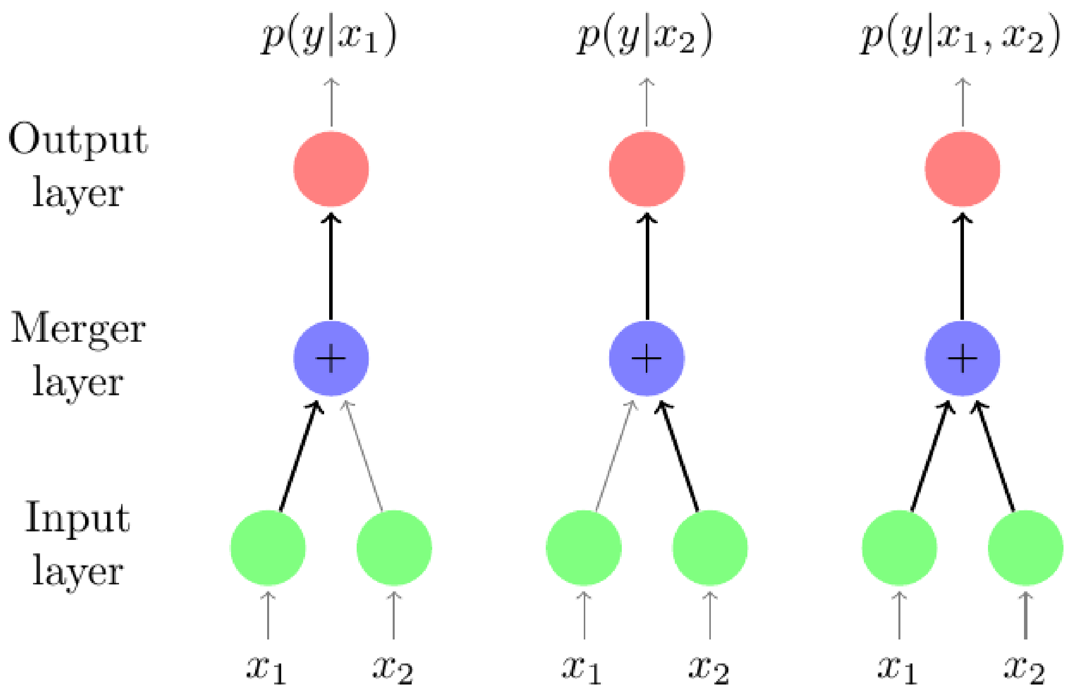


Figure 5.1: Multi-source Prediction Problems in a very simple Neural Network (NN). Bold arrows represent the source-target mapping. At the leftmost two scenarios, NN performs one-to-one mappings ($x_1 \rightarrow y$, $x_2 \rightarrow y$) and at the rightmost scenario, NN performs many-to-one mapping. The merger layer, indicated with a (+) node, should be adaptable to use either sources alone and/or both sources at the same time.

of the multi-view data. Consider a simple multi-seq2seq architecture, that has two input sources x_1, x_2 and one output target y, with both input and output sequences are always length one for simplicity. With the availability of source-target pair data (2-tuple or bi-text) in the form of (y, x_1) and (y, x_2), multi-seq2seq architecture can be trained to estimate $p(y|x_1)$ and $p(y|x_2)$. This means that, multi-seq2seq architecture will be trained to perform one-to-one mapping at training time and, at test time it will be tested again, to perform one-to-one mapping. Now consider the availability of multi-view data in the form of a 3-tuple (x_1, x_2, y). If this 3-tuple multi-view data is available during training along with 2-tuple data, multi-seq2seq architecture has to device a *merger layer* that can process single source information, coming from either source and also, multi-source information coming from multiple-sources at the same time. These scenarios are depicted in Fig. 5.1.

We now summarize the related work that makes use of multi-view data in a seq2seq framework. Zoph and Knight [140] was the first to extend NMT to handle multi-view data. Their proposed architecture consists of two separate encoders for two source languages, a single decoder for the target language, two separate attention modules for each source encoder and finally a merger layer that combines the information coming from two separate attention modules (see Fig. 3.2b). The crucial point of their proposed architecture is the merger layer which is implemented as a tree-LSTM [119] that combines multiple sources with a parametric function. As this function is parametrized with a neural network, it's parameters have to be learned during training, making the model necessitate multi-way data at training time. Since this is the only strategy that the model is trained, the model again has to be provided with multi-way data during test time. Further, bi-text is not used during training, which is almost always abundant compared to multi-way data. Although Zoph and Knight [140] developed a seq2seq architecture that necessitates multi-text both during training and test, the performance results on translation tasks are compelling.

In this thesis, we aim to develop multi-seq2seq architectures that can benefit from multi-way data when it is available, but also be able to be trainable by using bi-text only. A plausible combination is then, using both bi-text and multi-text during training and test.

Next, we will discuss two scenarios, and propose novel training and decoding methods, first when multi-view data is available both during training and test time, and second, the harder scenario, when multi-view data is available only during test time.

5.2 Availability of Multi-view Data Both During Training and Test

In this section, we assume the availability of multi-view data during training along with bi-text for each considered pair. To be concrete, having two separate views (sources) for input sequences, $\mathbf{x}_1$ and $\mathbf{x}_2$, with a single target sequence $\mathbf{y}$, multi-sequence modelling architecture aims to learn following mappings: first single source mappings, (1) $\mathbf{x}_1 \rightarrow \mathbf{y}$, (2) $\mathbf{x}_2 \rightarrow \mathbf{y}$ and then multi-source mapping (3) $\mathbf{x}_1 + \mathbf{x}_2 \rightarrow \mathbf{y}$. The crucial question is, how can we extend the proposed multi-sequence modelling architecture with a shared medium, to perform all above mappings at once. Simplest solution is, training the model using all multi- text and bi-text at the same time, using the very same *shared medium*. This can be achieved by alternating updates between (1), (2) and (3) and using a non-parametric function in the merger layer, such as mean, max, sum etc. The rationale of using a non-parametric function for the merger layer is, flexibility to the absence of any of the sources. If we were to train a parametric merger layer to combine information coming from two sources $\mathbf{x}_1$ and $\mathbf{x}_2$, then this function will not be able to used when only one of the sources is given [1]. Further, as we increase the number of views, each time we introduce an additional view, we need to train parametric merger layer with multi-view data.

Since learning a parametric merger layer is already being proposed by Zoph and Knight [140] and shown to improve translation quality, we skip this section to focus on a more interesting problem, where the multi-sequence modelling architecture is trained using bi-text only, but during test time, multi-view data is made available.

5.3 Availability of Multi-view Data During Test Only

Before going into the details of proposed strategies, let us remind our-selves the proposed multi-way multilingual architecture.

5.3.1 Model Description

The goal of multi-way, multilingual model is to build a neural translation model that can translate a source sentence given in one of N languages into one of M target languages. Thus to handle those N source and M target languages, the model consists of N encoders and M decoders. Unlike these language-specific encoders and decoders, only a single attention mechanism is shared across all $M \times N$ language pairs.

Encoder An encoder for the n-th source language reads a source sentence $X = (x_1, \ldots, x_{T_x})$ as a sequence of linguistic symbols and returns a set of context vectors

[1] We can give zero input for the sources that are not available, but this does not generalize to the case when multiple sources are not given.

$C^n = \left\{\mathbf{h}_1^n, \ldots, \mathbf{h}_{T_x}^n\right\}$. The encoder is usually implemented as a bidirectional recurrent network [109], and each context vector $\mathbf{h}_t^n$ is a concatenation of the forward and reverse recurrent networks' hidden states at time t. Without loss of generality, we assume that the dimensionality of the context vector for all source languages are all same.

Decoder and Attention Mechanism A decoder for the m-th target language is a conditional recurrent language model [89]. At each time step t', it updates its hidden state by

$$\mathbf{z}_{t'}^m = \varphi^m(\mathbf{z}_{t'-1}^m, \tilde{y}_{t'-1}^m, \mathbf{c}_{t'}^m),$$

based on the previous hidden state $\mathbf{z}_{t'-1}^m$, previous target symbol $\tilde{y}_{t'-1}^m$ and the time-dependent context vector $\mathbf{c}_{t'}^m$. φ^m is a gated recurrent unit (GRU, [24]).

The time-dependent context vector is computed by the shared attention mechanism as a weighted sum of the context vectors from the encoder C^n:

$$\mathbf{c}_{t'}^m = \mathbf{U} \sum_{t=1}^{T_x} \alpha_{t,t'}^{m,n} \mathbf{h}_t^n + \mathbf{b}, \tag{5.1}$$

where

$$\alpha_{t,t'}^{m,n} \propto \exp\left(f_{\text{score}}(\mathbf{W}^n \mathbf{h}_t^n, \mathbf{W}^m \mathbf{z}_{t'-1}^m, \tilde{y}_{t'-1}^m)\right). \tag{5.2}$$

The scoring function f_{score} returns a scalar and is implemented as a feedforward neural network with a single hidden layer. For more variants of the attention mechanism for machine translation, see [86].

The initial hidden state of the decoder is initialized as

$$\mathbf{z}_0^m = \phi_{\text{init}}^m(\mathbf{W}^n \mathbf{h}_t^n). \tag{5.3}$$

With the new hidden state $\mathbf{z}_{t'}^m$, the probability distribution over the next symbol is computed by

$$p(y_t = w | \tilde{y}_{<t}, X^n) \propto \exp(g_w^m(\mathbf{z}_t^m, \mathbf{c}_t^m, \mathbf{E}_y^m\left[\tilde{y}_{t-1}\right]), \tag{5.4}$$

where g_w^m is a decoder specific parametric function that returns the unnormalized probability for the next target symbol being w.

5.3.2 Learning

Training this multi-way, multilingual model does not require multi-way parallel corpora but only a set of bilingual corpora. For each bilingual pair, the conditional log-probability of a ground-truth translation given a source sentence is maximize by adjusting the relevant parameters following the gradient of the log-probability.

5.3.3 Proposed Decoding (Translation) Strategies: One-to-One Translation

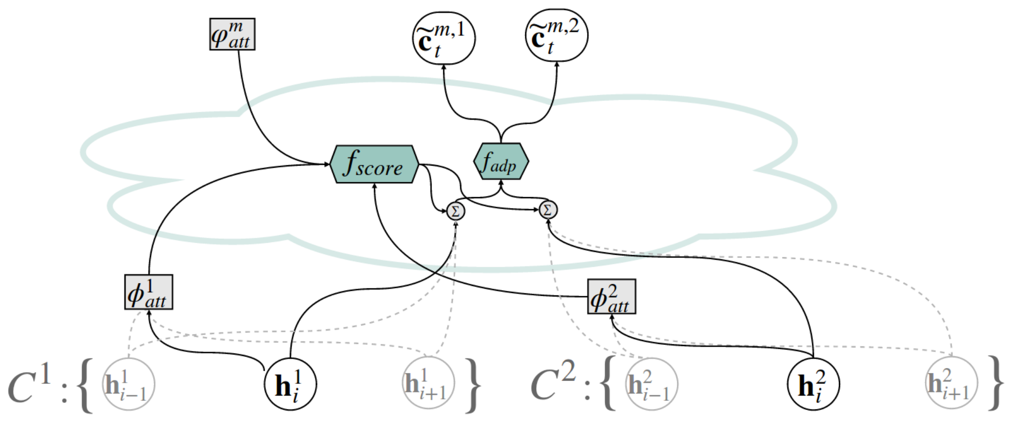
Up until now, in the context of proposed multi-sequence modelling architecture, only one translation strategy was evaluated, that is, *one-to-one translation*. This one-to-one strategy works on a source sentence given in one language by taking the encoder of that source language, the decoder of a target language and the shared attention mechanism. These three components are glued together as if they form a single-pair neural translation model and translates the source sentence into a target language.

We however notice that this is not the only translation strategy available with the multi-way, multilingual model. As we end up with multiple encoders, multiple decoders and a shared attention mechanism, this model naturally enables us to exploit a source sentence given in multiple languages, leading to a *many- to-one translation* strategy which was proposed recently by [140] in the context of neural machine translation.

Unlike [140], the multi-way, multilingual model is not trained with multi-way parallel corpora. This however does not necessarily imply that the model cannot be used in this way. In the remainder of this section, we propose two alternatives for doing multi-source translation with the multi-way, multilingual model, which eventually pave the way towards zero-resource translation of Chapter 6.

5.3.4 Proposed Decoding (Translation) Strategies: Many-to-One Translation

In this section, we consider a case where a source sentence is given in two languages, X_1 and X_2. However, any of the approaches described below applies to more than two source languages trivially.


φ^m_{att}
$\tilde{\mathbf{c}}_t^{m,1}$
$\tilde{\mathbf{c}}_t^{m,2}$
f_{score}
f_{adp}
Σ
Σ
ϕ^1_{att}
ϕ^2_{att}
$C^1 : \{ \mathbf{h}^1_{i-1} \quad \mathbf{h}^1_i \quad \mathbf{h}^1_{i+1} \}$
$C^2 : \{ \mathbf{h}^2_{i-1} \quad \mathbf{h}^2_i \quad \mathbf{h}^2_{i+1} \}$
Figure 5.2: Multi-Source Translation and Interpretation of Shared Medium.

Given X_1 and X_2 and separate encoders for each of the source languages, we first obtain context sets C^1 and C^2 for both sequences. Then, the *shared medium*, the attention module, generates two context vectors $\tilde{\mathbf{c}}_t^{m,1}$, $\tilde{\mathbf{c}}_t^{m,2}$ for each source sequence, given a query from the decoder m via φ^m_{att}. The next question is, how can we combine these two sources of information? The overall picture is also illustrated in Fig. **??**.

In our proposed multi-way, multilingual model, multi-source translation can be thought of as averaging two separate translation paths. For instance, in the case of Es+Fr to En, we want to combine Es→En and Fr→En so as to get a better English translation. We notice that there are two points in the multi-way, multilingual model where this averaging may happen.

Early Average The first candidate is to averaging two translation paths when computing the time-dependent context vector (see Eq. (5.1).) At each time t in the decoder, we compute a time-dependent context vector for each source language, $\tilde{\mathbf{c}}_t^1$ and $\tilde{\mathbf{c}}_t^2$ respectively for the two source languages. In this early averaging strategy, we simply take the average of these two context vectors:

$$\mathbf{c}_t = \frac{\tilde{\mathbf{c}}_t^1 + \tilde{\mathbf{c}}_t^2}{2}. \tag{5.5}$$

Similarly, we initialize the decoder's hidden state to be the average of the initializers of the two encoders:

$$\mathbf{z}_0 = \frac{1}{2}\left(\phi_{\text{init}}(\phi^1_{\text{init}}(\mathbf{h}^1_{T_{x_1}})) + \phi_{\text{init}}(\phi^2_{\text{init}}(\mathbf{h}^2_{T_{x_1}}))\right), \tag{5.6}$$

where ϕ_{init} is the decoder's initializer (see Eq. (5.3).)

Late Average Alternatively, we can average those two translation paths (e.g., Es→En and Fr→En) at the output level. At each time t, each translation path computes the distribution over the target vocabulary, i.e., $p(y_t = w|y_{<t}, X_1)$ and $p(y_t = w|y_{<t}, X_2)$. We then average them to get the multi-source output distribution:

$$\begin{aligned} p(y_t = w|y_{<t}, X_1, X_2) = & \\ & \frac{1}{2}(p(y_t = w|y_{<t}, X_1) + p(y_t = w|y_{<t})). \end{aligned} \tag{5.7}$$

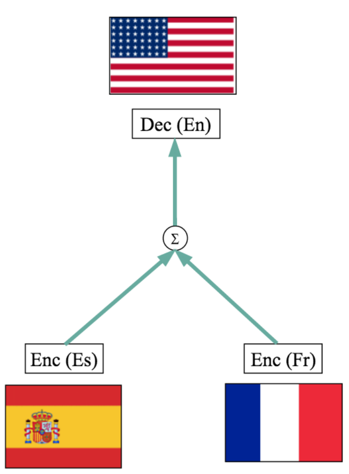
An advantage of this late averaging strategy over the early averaging one is that this can work even when those two translation paths were not from a single multilingual model. They can be two separately trained single-pair models. In fact, if X_1 and X_2 are same and the two translation paths are simply two different models trained on the same language pair–direction, this is equivalent to constructing an ensemble, which was found to greatly improve translation quality [118, 65]

Early+Late Average The two strategies above can be further combined by late-averaging the output distributions from the early averaged model and the late averaged one. We empirically evaluate this early+late average strategy as well.

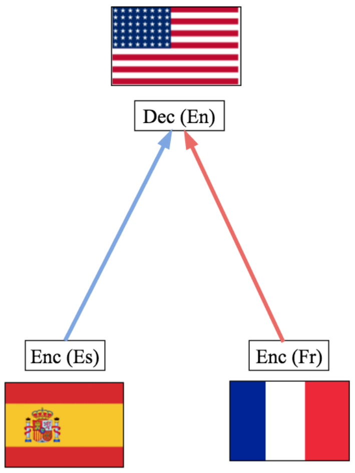
All three translation strategies are depicted in Fig. 5.3.

5.4 Experiments: Translation Strategies and Multi-Source Translation

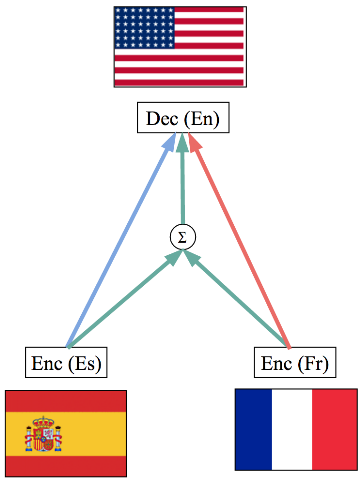
In this section, we evaluate the translation strategies described in the previous section on multi-source translation, as these translation strategies form a basic foundation on which we extend the multi-way, multilingual model for zero-resource machine translation of the next Chapter.



(a) Early-Averaging



(b) Late-Averaging



(c) Early+Late

Figure 5.3: Multi-Source Translation Strategies.

5.4.1 Settings

When evaluating the multi-source translation strategies, we use English, Spanish and French, and focus on a scenario where only En-Es and En-Fr parallel corpora are available.

5.4.1.1 Corpora

En-Es We combine the following corpora to form 34.71m parallel Es-En sentence pairs: UN (8.8m), Europarl-v7 (1.8m), news-commentary-v7 (150k), LDC2011T07-T12 (2.9m) and internal technical-domain data (21.7m).

Table 5.1: Data statistics. †: newstest-2012. ‡: newstest-2013

# Sents	Train	Dev†	Test‡
En-Es	34.71m	3003	3000
En-Fr	65.77m	3003	3000
En-Es-Fr	11.32m	3003	3000

En-Fr We combine the following corpora to form 65.77m parallel En-Fr sentence pairs: UN (9.7m), Europarl-v7 (1.9m), news-commentary-v7 (1.2m), LDC2011T07-T10 (1.6m), ReutersUN (4.5m), internal technical-domain data (23.5m) and Gigaword R2 (20.66m).

Evaluation Sets We use newstest-2012 and newstest-2013 from WMT as development and test sets, respectively.

Monolingual Corpora We do not use any additional monolingual corpus.

Preprocessing All the sentences are tokenized using the tokenizer script from Moses [77]. We then replace special tokens, such as numbers, dates and URL's with predefined markers, which will be replaced back with the original tokens *after* decoding. After using byte pair encoding (BPE, [112]) to get subword symbols, we end up with 37k, 43k and 45k unique tokens for English, Spanish and French, respectively. For training, we only use sentence pairs in which both sentences are only up to 50 symbols long.

See Table 5.1 for the detailed statistics.

5.4.2 Models and Training

We start from the code made publicly available as a part of [39]. We made two changes to the original code as described in the previous Chapter. First, we replaced

[1] `https://github.com/nyu-dl/dl4mt-multi`

Table 5.2: One-to-one translation qualities using the multi-way, multilingual model and four separate single-pair models.

			Multi		Single	
	Src	Trgt	Dev	Test	Dev	Test
(a)	Es	En	30.73	28.32	29.74	27.48
(b)	Fr	En	26.93	27.93	26.00	27.21
(c)	En	Es	30.63	28.41	31.31	28.90
(d)	En	Fr	22.68	23.41	22.80	24.05

the decoder with the conditional gated recurrent network with the attention mechanism as outlines in [40]. Second, we feed a binary indicator vector of which encoder(s) the source sentence was processed by to the output layer of each decoder (g_w^m in Eq. (5.4)). Each dimension of the indicator vector corresponds to one source language, and in the case of multi-source translation, there may be more than one dimensions set to 1.

We train the following models: four single-pair models (Es$\leftrightarrow$En and Fr$\leftrightarrow$En) and one multi-way, multilingual model (Es,Fr,En$\leftrightarrow$Es,Fr,En), see Fig. 5.4. As proposed by [39], we share one attention mechanism for the latter case.

Training We closely follow the setup from [39]. Each symbol is represented as a 620-dimensional vector. Any recurrent layer, be it in the encoder or decoder, consists of 1000 gated recurrent units (GRU, [24]), and the attention mechanism has a hidden layer of 1200 $\tanh$ units (f_{score} in Eq. (5.2)). We use Adam [72] to train a model, and the gradient at each update is computed using a minibatch of at most 80 sentence pairs. The gradient is clipped to have the norm of at most 1 [100]. We early-stop any training using the T-B score on a development set.[2]

5.4.3 One-to-One Translation

We first confirm that the multi-way, multilingual translation model indeed works as well as single-pair models on the translation paths that were considered during train-

[2] [2]T-B score is defined as $\frac{\text{TER}-\text{BLEU}}{2}$ which we found to be more stable than either TER or BLEU alone for the purpose of early-stopping [138].

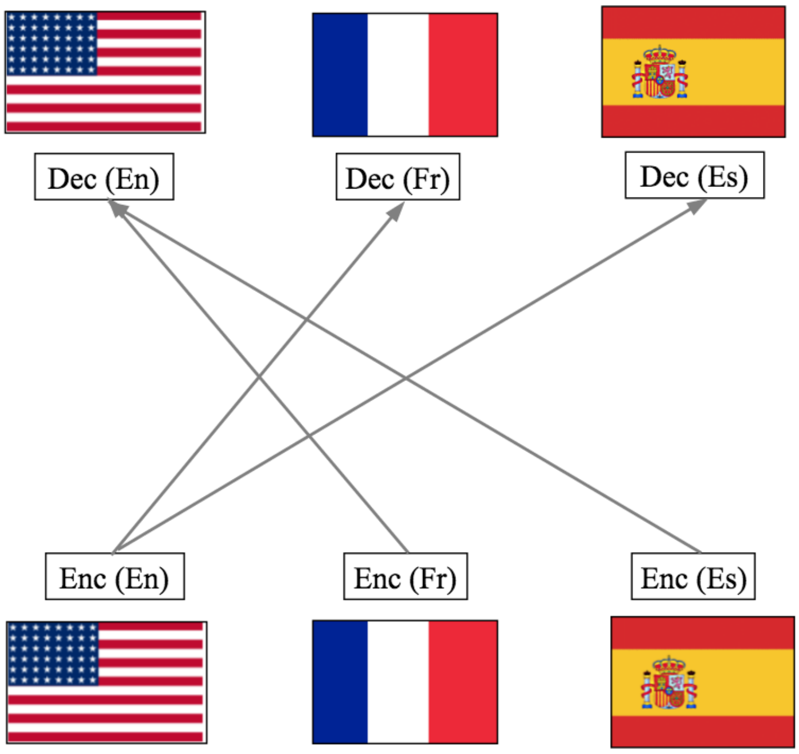


Figure 5.4: Multi-source Prediction Problems in a very simple Neural Network.

ing, which was the major claim in [39]. In Table 5.2, we present the results on four language pair-directions (Es↔En and Fr↔En).

It is clear that the multi-way, multilingual model indeed performs comparably on all the four cases with less parameters (due to the shared attention mechanism.) As observed earlier in [39], we also see that the multilingual model performs better when a target language is English.

Table 5.3: Many-to-one quality (Spanish + French → English) using three translation strategies. Compared to Table 5.2 (a–b) we observe a significant improvement (up to 3+ BLEU), although the model was never trained in these many-to-one settings. The second column shows the quality by the ensemble of two separate single-pair models.

		Multi		Single	
		Dev	Test	Dev	Test
(a)	Early	31.89	31.35	–	–
(b)	Late	32.04	31.57	32.00	31.46
(c)	E+L	32.61	31.88	–	–

5.4.4 Many-to-One Translation

We consider translating from a pair of source sentences in Spanish (Es) and French (Fr) to English (En). It is important to note that the multilingual model was *not* trained with any multi-way parallel corpus. Despite this, we observe that the early averaging strategy improves the translation quality (measured in BLEU) by 3 points in the case of the test set (compare Table 5.2 (a–b) and Table 5.3 (a).) We conjecture that this happens as training the multilingual model has implicitly encouraged the model to find a *common context vector* space across multiple source languages.

The late averaging strategy however outperforms the early averaging in both cases of multilingual model and a pair of single-pair models (see Table 5.3 (b)) albeit marginally. The best quality was observed when the early and late averaging strategies were combined at the output level, achieving up to +3.5 BLEU (compare Table 5.2 (a) and Table 5.3 (c).)

We emphasize again that there was *no* multi-way parallel corpus consisting of Spanish, French and English during training. [3] The result presented in this section shows that the multi-way, multilingual model can exploit multiple sources effectively without requiring any multi-way parallel corpus, and we will rely on this property together with the proposed many-to-one translation strategies in the later sections where we propose and investigate zero-resource translation.

5.4.5 What Does It Mean to Take the Average of Representations?

In the previous section, we proposed a test time strategy to leverage multi-view data, by simply taking the mean of time-dependent context vectors from multiple sources. The experimental results supported that, even if the multi-way multilingual model is trained using only bi-text, supposedly having no notion of multiple-views, simply by taking the average of representations during test time yields significant improvements.

Although we empirically show that such decoding strategy achieves better results, it

[3] We do not assume the availability of annotation on multi-way parallel sentence pairs. It is likely that there will be some sentence (or a set of very close variants of a single sentence) translated into multiple languages (eg. Europarl). One may decide to introduce a mechanism for exploiting these [140], or as we present here, it may not be necessary at all to do so.

is surprising that a simple arithmetic operation between intermediate contextual representations can achieve significant boost without even training the model. In this section we try to explain the above improvements by resorting to the *manifold hypothesis* [46]. Manifold hypothesis in connectionist models state that, hidden layers of a deep neural network flatten (stretch) the data generating distribution (the manifold) [7, 93, 17]. These manifolds are low dimensional regions in input space near which the distribution concentrates. Each point in the manifold, is then, surrounded by other highly (semantically) similar examples that can be reached by applying simple transformations. In practical terms, if the network is deep enough to disentangle the factors of variations in the data, into a flat enough manifold, then traversing in the manifold will translate into transitioning between semantically similar points.

Now let us consider the architecture proposed in this thesis. We first apply a deep encoder to generate a set of annotations by making use of encoder RNNs. Then these set of annotations are summarized by a deep attention module. Since the attention module is exposed to many source-target pairs, first, it is implicitly encouraged to find a common vector space that represents the shared commonalities across all source-target pairs. Second, since the mapping from input to the attention module output $\mathbf{c}_t$ is a quite deep and non-linear mapping, we would expect the output of the attention module lies in a flat (locally-linear) manifold. This hypothesis is illustrated in Fig. 5.6.

Then, consider we use the same *shared medium* to map two semantically similar input sequences, into this representational manifold (see Fig. 5.7). If the output of the *shared medium* is a locally-flat manifold, and two points are semantically similar, then their mean should also lie in the manifold, and hopefully the new point in the manifold is more informative than either of the points alone.

We tested this hypothesis by trying to brake the *manifold hypothesis* with simple arithmetic operations. We trained a multi-lingual model that translates from Es-Fr to En, by using only bi-text, but during training, we evaluated the model using multi-text with different arithmetic operations in the merger layer. In Fig. 5.5, red curves correspond to different merger operations, blue curve is the negative log-probability for Fr-En and green curve is the negative log probability for Es-En pair. When we apply *max* operation as the merger operation (curves with (x) label), resulting log- probabilities are close to Es-En curve and highly fluctuates. We can interpret this as the max operation yields sparse features, result of the operation may not be on the manifold. A similar result is observed for the *sum* operation (curves with (s) label). Since the sum of two points can overshoot the manifold tangent, results can still be off the manifold, explaining the similar performance with *max* operation. The *mean* operation on the other hand (curves with (m) label) achieves the best performance and fluctuates less compared to other operations, giving us more clue that the representations induced by the *shared medium*, is likely to lie on a semantic manifold.

We manipulated the manifold hypothesis to exploit the flatness of representations, and applied simple arithmetic operations, such as mean, between semantically similar points to obtain more informative representations during test time. These experiments pave the way to zero-resource translation where we again use the manifold hypothesis and exploit the locality of representations in the next Chapter. Also, our findings support that "with good disentangling, there is no need for further learning, only good inference." by Bengio[7].

5.5 Conclusion

In this chapter, we first showed that the multi-way, multilingual neural translation model by [39] is able to exploit common, underlying structures across many languages in order to better translate when a source sentence is given in multiple languages. This confirms the usefulness of positive language transfer, which has been believed to be an important factor in human language learning [97, 103], in machine translation. Furthermore, our result significantly expands the applicability of multi-source translation [140], as it does not assume the availability of multi-way parallel corpora for training and relies only on *bilingual* parallel corpora.

Second, although the proposed many-to-one translation is indeed generally applicable to any number of source languages, we have only tested a source sentence in two languages. We expect even higher improvement with more languages, but it must be tested thoroughly in the future.

Being able to perform many-to-one mapping further allows us to extend the multi-

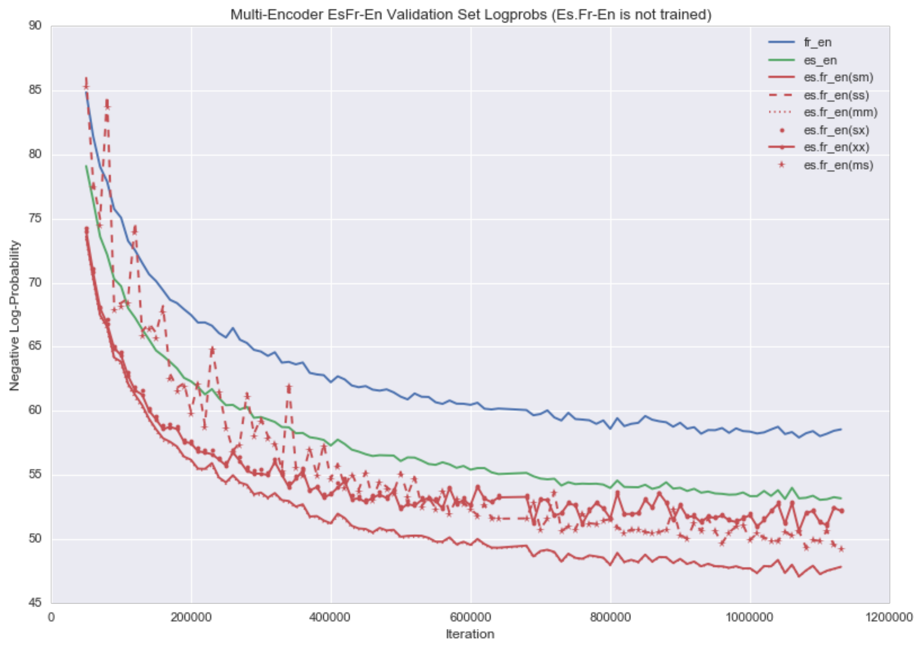


Figure 5.5: Merger Operations for Multi-Source Translation. Blue curve corresponds to validation set negative log-likelihood of French $\rightarrow$ English, red green curve corresponds validation set negative log-likelihood of Spanish $\rightarrow$ English, and all red curves correspond to validation set negative log-likelihood of multi-source French + Spanish $\rightarrow$ English with different merger operations.

sequence modelling architecture to make use of different modalities, opening new research directions for multi-modal mapping problems .

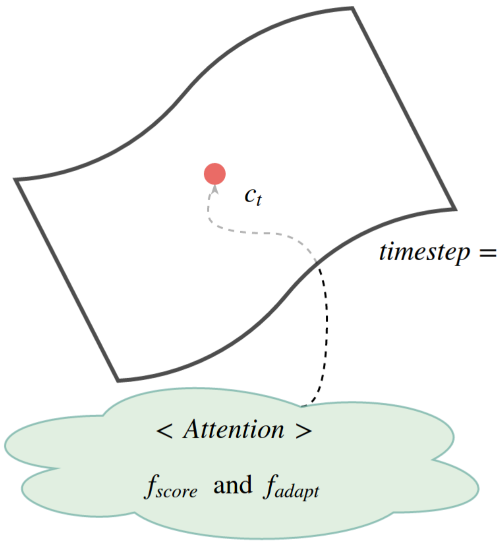


Figure 5.6: Manifold-hypothesis and Interpretation of Shared Medium.

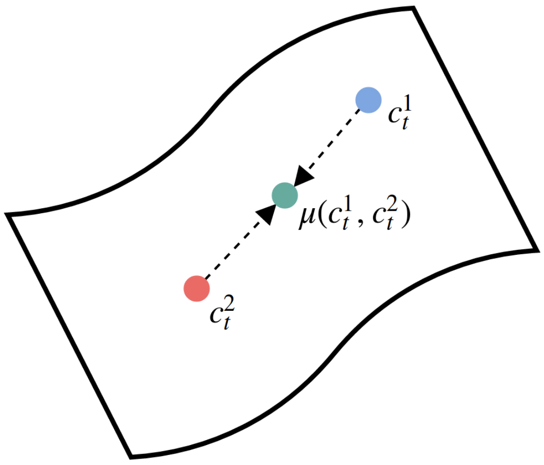


Figure 5.7: Multi-Source Translation and Interpretation of Shared Medium.

CHAPTER 6

MULTI-SEQUENCE MODELING FOR ZERO-DATA REGIMES: ZERO-RESOURCE TRANSLATION

"These conclusions have been based upon a finite segment of a numerically determined solution. They cannot be regarded as mathematically proven, even though the evidence for them is strong. One apparent contradiction requires further examination. It is difficult to reconcile the merging of two surfaces, one containing each spiral, with the inability of two trajectories to merge. It is not difficult, however, to explain the apparent merging of the surfaces."

EDWARD LORENZ - DETERMINISTIC NON-PERIODIC FLOW, 1962

In this section, we propose a novel finetuning algorithm for the multi-way, multilingual neural machine translation architecture proposed previously, that enables zero-resource machine translation. When used together with novel many-to-one translation strategies introduced in Chapter 5, we empirically show that this finetuning algorithm allows the multi-way, multilingual model to translate a zero- resource language pair (1) as well as a single-pair neural translation model trained with up to 1M direct parallel sentences of the same language pair and (2) better than pivot-based translation strategy, while keeping only one additional copy of attention-related parameters.

We move on to zero-resource translation by first evaluating a vanilla multi-way, multilingual model on a zero-resource language pair, which revealed that the vanilla model cannot do zero-resource translation in Sec. 6.2.1. Based on the many-to-one strategies we proposed earlier, we design a novel finetuning strategy that does not require any direct parallel corpus between a target, zero-resource language pair in Sec. 6.1.2, which uses the idea of generating a pseudo-parallel corpus [110]. This strategy makes an additional copy of the attention mechanism and finetunes only this small set of parameters.

Large-scale experiments with Spanish, French and English show that the proposed finetuning strategy allows the multi-way, multilingual neural translation model to perform zero-resource translation as well as a single-pair neural translation model trained with up to 1M true parallel sentences. This result re-confirms the potential of the multi-way, multilingual model for low/zero-resource language translation, which we argued in Chapter 3 and 4 [39].

6.1 Zero-Resource Translation Strategies

The network architecture of multi-way, multilingual model suggests the potential for translating between two languages *without* any direct parallel corpus available. In the setting considered in this chapter (see Sec. 5.4.1,) these translation paths correspond to Es$\leftrightarrow$Fr (dashed orange arrow in Fig. 6.1), as only parallel corpora used for training were Es$\leftrightarrow$En and Fr$\leftrightarrow$En (solid black arrows in Fig. 6.1).

The most naive approach for translating along a zero-resource path is to simply treat it as any other path that was included as a part of training. This corresponds to the one-to-one strategy from Sec. 5.3.3. In our experiments, it however turned out that this naive approach does not work at all, as can be seen in Table 6.1 (a).

In this section, we investigate this potential of zero-resource translation with the multi-way, multilingual model in depth. More specifically, we propose a number of approaches that enable zero-resource translation without requiring any additional bilingual or multi-way corpora.

6.1.1 Pivot-based Translation

The first set of approaches exploits the fact that the target zero-resource translation path can be decomposed into a sequence of high-resource translation paths [130, 124, 54]. For instance, in our case, Es$\rightarrow$Fr can be decomposed into a sequence of Es$\rightarrow$En

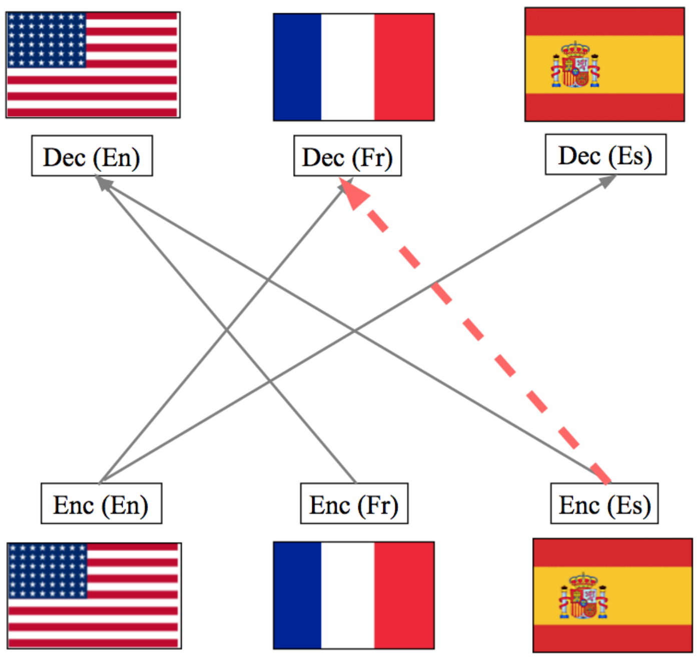


Figure 6.1: Zero-shot Pair in a multilingual neural machine translation model. Pairs with parallel data are English $\rightarrow$ French, English $\rightarrow$ Spanish, French $\rightarrow$ English, Spanish $\rightarrow$ English, indicated by gray arrows. Zero shot pair, Spanish $\rightarrow$ French, has no direct parallel data, and indicated by dashed red arrow.

and En$\rightarrow$Fr. In other words, we translate a source sentence (Es) into a pivot language (En) and then translate the English translation into a target language (Fr), all within the same multi-way, multilingual model trained by using bilingual corpora.

One-to-One Translation The most basic approach here is to perform each translation path in the decomposed sequence independently from each other. This one-to-one approach introduces only a minimal computational complexity (the multiplicative factor of two.) We can further improve this one-to-one pivot-based translation by maintaining a set of k-best translations from the first stage (Es$\rightarrow$En), but this increase the overall computational complexity by the factor of k, making it impractical in practice. We therefore focus only on the former approach of keeping the best pivot translation in this section.

Many-to-One Translation With the multi-way, multilingual model proposed in this thesis, we can extend the naive one-to-one pivot-based strategy by replacing the second stage (En→Fr) to be many-to-one translation from Sec. 5.4.4 using both the original source language and the pivot language as a pair of source languages. We first translate the source sentence (Es) into English, and use both the original source sentence and the English translation (Es+En) to translate into the final target language (Fr).

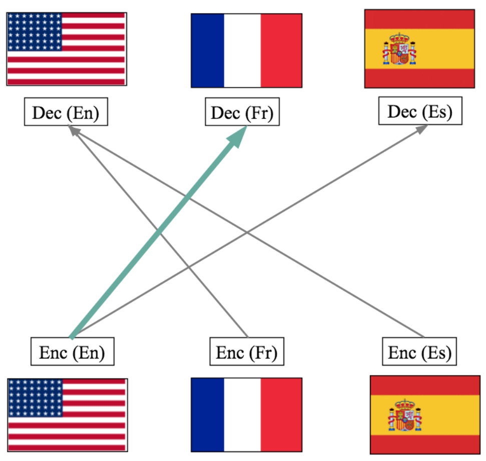
Both approaches described and proposed above do not require any additional action on an already-trained multilingual model. They are simply different translation strategies specifically aimed at zero-resource translation.

6.1.2 Finetuning with Pseudo Parallel Corpus

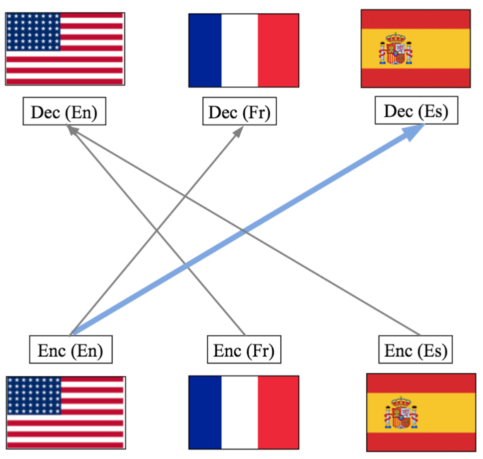
The failure of the naive zero-resource translation earlier (see Table 6.1 (a)) suggests that the context vectors returned by the encoder are not compatible with the decoder, when the combination was not included during training. The good translation qualities of the translation paths included in training however imply that the representations learned by the encoders and decoders are good. Based on these two observations, we conjecture that all that is needed for a zero-resource translation path is a simple adjustment that makes the context vectors from the encoder to be compatible with the target decoder. Thus, we propose to adjust this zero-resource translation path however without any additional parallel corpus.

First, we generate a small set of *pseudo bilingual pairs* of sentences for the zero-resource language pair (Es→Fr) in interest, Fig. 6.2a - 6.2b. We randomly select N sentences pairs from a parallel corpus between the target language (Fr) and a pivot language (En) and translate the pivot side (En) into the source language (Es). Then, the pivot side is discarded, and we construct a *pseudo* parallel corpus consisting of sentence pairs of the source and target languages (Es-Fr).

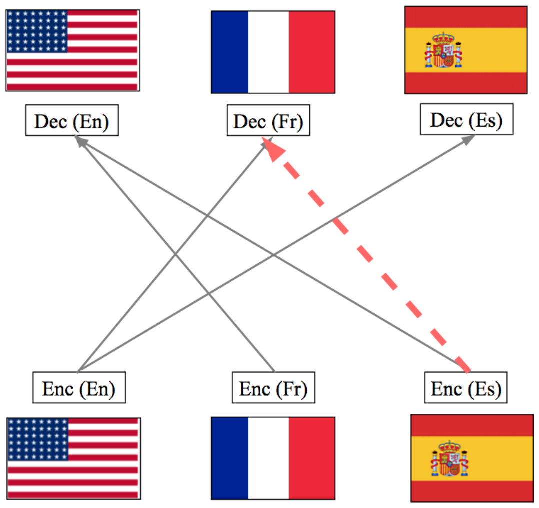
We make a copy of the existing attention mechanism, to which we refer as *target-specific attention mechanism.* We then finetune only this target-specific attention mechanism while keeping all the other parameters of the encoder and decoder intact, using the generated pseudo parallel corpus. We do not update any other parameters in the encoder and decoder, because they are already well-trained (evidenced by high



(a) Given Pair



(b) Generate Pseudo Parallel Data



(c) Finetune with Generated Pseudo-parallel Data

Figure 6.2: Finetuning with Pseudo Parallel Corpus

Table 6.1: Zero-resource translation from Spanish (Es) to French (Fr) *without* finetuning, using multi-way, multilingual model. When pivot is $\surd$, English is used as a pivot language.

	Pivot	Many-to-1	Dev	Test
(a)			< 1	< 1
(b)	$\surd$		20.64	20.4
(c)	$\surd$	Early	9.24	10.42
(d)	$\surd$	Late	18.22	19.14
(e)	$\surd$	E+L	13.29	14.56

translation qualities in Table 5.2) and we want to avoid disrupting the well-captured structures underlying each language.

Once the model has been finetuned with the pseudo parallel corpus, Fig. 6.2c, we can use any of the translation strategies described earlier in Sec. **??** for the finetuned zero-resource translation path. We expect a similar gain by using many-to-one translation, which we empirically confirm in the next section.

6.2 Experiments: Zero-Resource Translation

6.2.1 Without Finetuning

Settings We use the same multi-way, multilingual model trained earlier in Sec. 5.4.2 to evaluate the zero-resource translation strategies. We emphasize here that this model was trained only using Es-En and Fr-En *bilingual* parallel corpora without any Es-Fr parallel corpus.

We evaluate the proposed approaches to zero-resource translation with the same multi-way, multilingual model from Sec. 5.4.1. We specifically select the path from Spanish to French (Es$\to$Fr) as a target zero-resource translation path.

Result and Analysis As mentioned earlier, we observed that the multi-way, multilingual model *cannot* directly translate between two languages when the translation path between those two languages was not included in training (Table 6.1 (a).) On the other hand, the model was able to translate decently with the pivot-based one-to-one translation strategy, as can be seen in Table 6.1 (b). Unsurprisingly, all the many-to-one strategies resulted in worse translation quality, which is due to the inclusion of the useless translation path (direct path between the zero-resource pair, Es-Fr). Another interesting trend we observe is the Early+Late averaging (Table 6.1 (e)) seems to perform worse than Late averaging (Table 6.1 (d)) alone, opposite of the results in Table 5.3 (b-c). We conjecture that, by simply averaging two model outputs (as in E+L), when one of them is drastically worse than the other, has the effect of pulling down the performance of final results. But early averaging can still recover from this deficiency, upto some extent, since the decoder output probability function g_w^m (Eq. (5.4).) is a smooth function not only using the averaged context vectors (Eq. (5.5).).

These results clearly indicate that the multi-way, multilingual model trained with only bilingual parallel corpora is not capable of direct zero-resource translation *as it is*.

Table 6.2: Zero-resource translation from Spanish (Es) to French (Fr) *with* finetuning. When pivot is $\surd$, English is used as a pivot language. Row (b) is from Table 6.1 (b).

				Pseudo Parallel Corpus				True Parallel Corpus			
	Pivot	Many-to-1		1k	10k	100k	1m	1k	10k	100k	1m
(a)	Single-Pair Models		Dev	–	–	–	–	–	–	11.25	21.32
			Test	–	–	–	–	–	–	10.43	20.35
(b)	$\surd$	No Finetuning		Dev: 20.64, Test: 20.4				–			
(c)			Dev	0.28	10.16	15.61	17.59	0.1	8.45	16.2	20.59
			Test	0.47	10.14	15.41	17.61	0.12	8.18	15.8	19.97
(d)	$\surd$	Early	Dev	19.42	21.08	21.7	21.81	8.89	16.89	20.77	22.08
			Test	19.43	20.72	21.23	21.46	9.77	16.61	20.40	21.7
(e)	$\surd$	Early+	Dev	20.89	20.93	21.35	21.33	14.86	18.28	20.31	21.33
		Late	Test	20.5	20.71	21.06	21.19	15.42	17.95	20.16	20.9

6.2.2 Finetuning with a Pseudo Parallel Corpus

Settings The proposed finetuning strategy raises a number of questions. First, it is unclear how many pseudo sentence pairs are needed to achieve a decent translation quality. Because the purpose of this finetuning stage is simply to adjust the shared attention mechanism so that it can properly bridge from the source-side encoder to the target-side decoder, we expect it to work with only a small amount of pseudo pairs. We validate this by creating pseudo corpora of different sizes–1k, 10k, 100k and 1m.

Second, we want to know how detrimental it is to use the generated pseudo sentence pairs compared to using true sentence pairs between the target language pair. In order to answer this question, we compiled a true multi-way parallel corpus by combining the subsets of UN (7.8m), Europarl-v7 (1.8m), OpenSubtitles-2013 (1m), news-commentary-v7 (174k), LDC2011T07 (335k) and news-crawl (310k), and use it to finetune the model.[1] This allows us to evaluate the effect of the pseudo and true parallel corpora on finetuning for zero-resource translation.

Lastly, we train single-pair models translating directly from Spanish to French by using the true parallel corpora. These models work as a baseline against which we compare the multi-way, multilingual models.

Training Unlike the usual training procedure described in Sec. 5.4.2, we compute the gradient for each update using 60 sentence pairs only, when finetuning the model with the multi-way parallel corpus (either pseudo or true.)

Result and Analysis Table 6.2 summarizes all the result. The most important observation is that the proposed finetuning strategy with *pseudo*-parallel sentence pairs outperforms the pivot-based approach (using the early averaging strategy from Sec. 5.4.4) even when we used only 10k such pairs (compare (b) and (d).) As we increase the size of the pseudo-parallel corpus, we observe a clear improvement. Furthermore, these models perform comparably to or better than the single-pair model

[1] See the last row of Table 5.1.

trained with 1M *true* parallel sentence pairs, *although they never saw a single true bilingual sentence pair* of Spanish and French (compare (a) and (d).)

Another interesting finding is that it is only beneficial to use true parallel pairs for finetuning the multi-way, mulitilingual models when there are enough of them (1m or more). When there are only a small number of true parallel sentence pairs, we even found using pseudo pairs to be more beneficial than true ones. This effective as more apparent, when the direct one-to-one translation of the zero-resource pair was considered (see (c) in Table 6.2.) This applies that the misalignment between the encoder and decoder can be largely fixed by using pseudo-parallel pairs only, and we conjecture that it is easier to learn from pseudo-parallel pairs as they better reflect the inductive bias of the trained model and as the pseudo- parallel corpus is expected to be more noisy, this may be an implicit regularization effect. When there is a large amount of true parallel sentence pairs available, however, our results indicate that it is better to exploit them.

Unlike we observed with the multi-source translation in Sec. 5.3.4, we were not able to see any improvement by further averaging the early-averaged and late-average decoding schemes (compare (d) and (e).) This may be explained by the fact that the context vectors computed when creating a pseudo source (e.g., En from Es when Es$\rightarrow$Fr) already contains all the information about the pseudo source. It is simply enough to take those context vectors into account via the early averaging scheme.

These results clearly indicate and verify the potential of the multi-way, multilingual neural translation model in performing zero-resource machine translation. More specifically, it has been shown that the translation quality can be improved even without any direct parallel corpus available, and if there is a small amount of direct parallel pairs available, the quality may improve even further.

6.3 Conclusion: Implications and Limitations

Implications There are two main results in this chapter. First, we showed that the multi-way, multilingual neural translation model proposed in this thesis, [39], is able to exploit common, underlying structures across many languages in order to better translate when a source sentence is given in multiple languages. This confirms the usefulness of positive language transfer, which has been believed to be an important factor in human language learning [97, 103], in machine translation. Furthermore, our result significantly expands the applicability of multi-source translation [140], as it does not assume the availability of multi-way parallel corpora for training and relies only on *bilingual* parallel corpora.

Second, the experiments on zero-resource translation revealed that it is not necessary to have a direct parallel corpus, or deep linguistic knowledge, between two languages in order to build a machine translation system. Importantly we observed that the proposed approach of zero-resource translation is better both in terms of translation quality and data efficiency than a more traditional pivot-based translation [130, 124]. Considering that this is the first attempt at such zero-resource, or extremely low-resource, translation using neural machine translation, we expect a large progress in near future.

Limitations Despite the promising empirical results presented in this paper, there are a number of shortcomings that needs to addressed in follow-up research. First, our experiments have been done only with three European languages–Spanish, French and English. More investigation with a diverse set of languages needs to be done in order to make a more solid conclusion, such as was done in [39, 28]. Furthermore, the effect of varying sizes of available parallel corpora on the performance of zero-resource translation must be studied more in the future.

Second, although the proposed many-to-one translation is indeed generally applicable to any number of source languages, we have only tested a source sentence in two languages. We expect even higher improvement with more languages, but it must be tested thoroughly in the future.

Lastly, the proposed finetuning strategy requires the model to have an additional set of parameters relevant to the attention mechanism for a target, zero-resource pair. This implies that the number of parameters may grow linearly with respect to the number of target language pairs. We expect future research to address this issue by, for instance, mixing in the parallel corpora of high-resource language pairs during finetuning as well.

CHAPTER 7

CONCLUSION

In this thesis, we first proposed a multi-way, multilingual neural machine translation method which is the first multi-seq2seq architecture, explicitly modelling *interlingua*. We initially showed that, the proposed approach enables a single neural translation model to translate between multiple languages, with a number of parameters that grows only linearly with the number of languages. This is made possible by having a single attention mechanism, that is shared across all language pairs, *the shared medium*, by explicitly parameterizing *interlingua*. We demonstrated that a single neural network can be trained on ten language pairs from WMT'15 simultaneously and observe a substantial performance improvements over models trained on a single language pair.

Following chapter, we analyzed the behavior of multi-seq2seq architecture when there is not enough parallel data for a given source-target pair. We showed by experiments that the proposed multi-way, multilingual model generalizes better than the single-pair translation models, when the amount of available parallel corpus is small. We validated that this is not only due to the increased amount of target-side, monolingual corpus but the capability of positive language transfer.

Next, we explored one very interesting and natural use case of the multi-sequence modelling architecture, the mapping problems when there are more than one views (or sources) of the same data point (example). We investigated the possible extensions of the proposed multi-sequence modelling architecture, when multi-view and multi-source data is available. We first framed many-to-one mapping problem into multi-sequence mapping architecture. Then we described different scenarios of many-to-one mapping problem according to the availability of multi-view data, either during training or test time. Next, we proposed novel decoding strategies that exploit model ensembles in multi-sequence mapping architecture or manifold hypothesis of the attention module, the *shared medium*.

Finally, we proposed a novel finetuning algorithm for the multi-way, multilingual neural machine translation architecture, that enables zero-resource machine translation. When used together with novel many-to-one translation strategies, also proposed in this thesis, we empirically showed that the finetuning algorithm allows the multi-way, multilingual model to translate a zero- resource language pair (1) as well as a single-pair neural translation model trained with up to 1M direct parallel sentences of the same language pair and (2) better than pivot-based translation strategy, while keeping only one additional copy of attention-related parameters.

Connectionist multi-sequence modelling architecture and the techniques proposed in this thesis opened up various research directions to be pursued, below we summarize recent extensions of our proposed multi-sequence model by the research community.

- Multi-modal Extensions of Multi-Sequence Modeling. A natural extension of multi-sequence modelling approach is to replace or introduce new modalities as additional encoders or decoders in the proposed multi-seq2seq architecture. [13, 14] extended the multi-seq2seq approach proposed in this thesis for image-grounded translation tasks, where multiple sequences are presented to the model that has an image encoder and a text encoder, translating the input sequences to a translated output text sequence.
- Zero-Shot Translation. Following Chapter 6 of this thesis, [67, 53] extended the *shared-medium* to it's extreme, where the shared components between a set of encoders and decoders not only span the attention mechanism but entire encoders, decoders and attention mechanism. This is made possible by collapsing the encoders and decoders to a single encoder and decoder, sharing the same encoder across all the source languages, a single shared attention module, and again sharing the same decoder across all the target languages. This extreme *shared-medium* appears as a natural extension of this thesis, and made possible to perform zero-shot translation without any finetuning phase, as proposed in

Chapter 6 of this thesis.

- Multi-task learning. As the target side decoders can represent different tasks, multi-seq2seq architecture is extended to multi-task learning [81] by making use of a shared memory. Further, multi-seq2seq allows to use mono-lingual data to enable unsupervised learning in seq2seq architectures [102, 136].
- Larger-Context Translation and System Combination. Last, by extending the multi-source translation strategies proposed in this thesis, it is straight-forward to replace different language encoders to a single encoder that encodes only one language, but the sequences coming from the left and right context sentences of the source sequence. This allows a multi-seq2seq model to make use of larger (or extended) context and opens up a new direction for neural translation models to translate in context [66].

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas. Lipnet: Sentence-level lipreading. *CoRR*, abs/1611.01599, 2016.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, 2014.

[4] L. R. Bahl, F. Jelinek, and R. L. Mercer. Readings in speech recognition. pages 308–319, 1990.

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, Aug. 2013.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003.

[7] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 552–560, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[8] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, Mar. 1994.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] L. Bottou, Y. Bengio, and Y. Le Cun. Global training of document processing systems using graph transformer networks. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 489–494. IEEE, 1997.

[11] J. S. Bridle. *Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*, pages 227–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[12] O. Caglayan, W. Aransa, Y. Wang, M. Masana, M. García-Martínez, F. Bougares, L. Barrault, and J. van de Weijer. Does multimodality help human and machine for translation and image captioning? *arXiv preprint arXiv:1605.09186*, 2016.

[13] O. Caglayan, L. Barrault, and F. Bougares. Multimodal attention for neural machine translation. *arXiv preprint arXiv:1609.03976*, 2016.

[14] I. Calixto, Q. Liu, and N. Campbell. Doubly-attentive decoder for multi-modal neural machine translation. *CoRR*, abs/1702.01287, 2017.

[15] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[16] M. A. Castaño, F. Casacuberta, and E. Vidal. Machine translation using neural networks and finite-state models, 1997.

[17] L. Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 2005.

[18] M. Cettolo, C. Girardi, and M. Federico. Wit3: Web inventory of transcribed and translated talks. *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, 2012.

[19] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[20] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. T. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *CoRR*, abs/1611.01838, 2016.

[21] K. Cho. Natural language understanding with distributed representation. Technical report, New York University, 2015. Lecture Note for DS-GA 3001.

[22] K. Cho, A. Courville, and Y. Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015.

[23] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Oct. 2014.

[24] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[25] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: first results. *arXiv preprint arXiv:1412.1602*, 2014.

[26] J. Chorowski and N. Jaitly. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016.

[27] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585, 2015.

[28] J. Chung, K. Cho, and Y. Bengio. A character-level decoder without explicit segmentation for neural machine translation. In *ACL*, 2016.

[29] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[30] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman. Lip reading sentences in the wild. *arXiv preprint arXiv:1611.05358*, 2016.

[31] J. S. Chung, A. W. Senior, O. Vinyals, and A. Zisserman. Lip reading sentences in the wild. *CoRR*, abs/1611.05358, 2016.

[32] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[33] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[34] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3061–3069, 2015.

[35] W. Ding, R. Wang, F. Mao, and G. W. Taylor. Theano-based large-scale visual recognition with multiple gpus. *arXiv:1412.2302*, 2014.

[36] D. Dong, H. Wu, W. He, D. Yu, and H. Wang. Multi-task learning for multiple language translation. In *ACL*, 2015.

[37] D. Dong, H. Wu, W. He, D. Yu, and H. Wang. Multi-task learning for multiple language translation. ACL, 2015.

[38] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179 – 211, 1990.

[39] O. Firat, K. Cho, and Y. Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*, 2016.

[40] O. Firat, K. Cho, and Y. Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *HLT-NAACL*, 2016. to appear.

[41] M. L. Forcada and R. P. Ñeco. Recursive hetero-associative memories for translation. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks: Biological and Artificial Computation: From Neuroscience to Technology*, IWANN '97, pages 453–462, London, UK, UK, 1997. Springer-Verlag.

[42] M. L. Forcada and R. P. Ñeco. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology*, pages 453–462. Springer, 1997.

[43] R. M. French. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[44] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[45] Y. Goldberg. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726, 2015.

[46] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[47] J. T. Goodman. A bit of progress in language modeling. *Comput. Speech Lang.*, 15(4):403–434, Oct. 2001.

[48] A. Graves. *Supervised sequence labelling with recurrent neural networks*. Studies in Computational intelligence. Springer, Heidelberg, New York, 2012.

[49] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[50] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[51] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recog-

nition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

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

[53] T. Ha, J. Niehues, and A. H. Waibel. Toward multilingual neural machine translation with universal encoder and decoder. *CoRR*, abs/1611.04798, 2016.

[54] N. Habash and J. Hu. Improving arabic-chinese statistical machine translation using english as pivot language. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 173–181. Association for Computational Linguistics, 2009.

[55] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, new ed edition, 1949.

[56] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[57] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[58] M. Hopkins and J. May. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[59] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[60] J. Hutchins. From first conception to first demonstration: the nascent years of machine translation, 1947–1954. a chronology. *Machine Translation*, 12(3):195–252, 1997.

[61] W. J. Hutchins. Machine translation over fifty years. *HISTOIRE, EPISTEMOLOGIE, LANGAGE, TOME XXII, FASC. 1 (2001)*, 23:7–31, 2001.

[62] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[63] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007, 2014.

[64] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In *ACL 2015*, 2014.

[65] S. Jean, O. Firat, K. Cho, R. Memisevic, and Y. Bengio. Montreal neural machine translation systems for wmt'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[66] S. Jean, S. Lauly, O. Firat, and K. Cho. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*, 2017.

[67] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016.

[68] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling, 2016.

[69] M. Junczys-Dowmunt, T. Dwojak, and H. Hoang. Is neural machine translation ready for deployment? A case study on 30 translation directions. *CoRR*, abs/1610.01108, 2016.

[70] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709, 2013.

[71] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.

[72] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*, 2015.

[73] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[74] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284, 2015.

[75] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995.

[76] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

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

[78] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

[79] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[80] W. Ling, I. Trancoso, C. Dyer, and A. W. Black. Character-based neural machine translation. *arXiv:1511.04586*, 2015.

[81] P. Liu, X. Qiu, and X. Huang. Deep multi-task learning with shared memory. *CoRR*, abs/1609.07222, 2016.

[82] W. N. W. N. Locke and j. e. Booth, Andrew Donald. *Machine translation of languages : fourteen essays*. Westport, Conn. : Greenwood Press, 1975. Reprint of the ed. published jointly by Technology Press of the Massachusetts Institute of Technology, Cambridge and Wiley, New York.

[83] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114, 2015.

[84] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[85] M.-T. Luong and C. D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv:1604.00788*, 2016.

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

[87] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*, 2015.

[88] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[89] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[90] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[91] M. Minsky. Steps toward artificial intelligence. In *Computers and Thought*, pages 406–450. McGraw-Hill, 1961.

[92] M. C. Mozer. Backpropagation. chapter A Focused Backpropagation Algorithm for Temporal Pattern Recognition, pages 137–169. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[93] H. Narayanan and S. K. Mitter. Sample Complexity of Testing the Manifold Hypothesis. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *NIPS*, pages 1786–1794. Curran Associates, Inc., 2010.

[94] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[95] G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

[96] N. Nguyen and Y. Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 681–688, New York, NY, USA, 2007. ACM.

[97] T. Odlin. *Language Transfer*. Cambridge University Press, 1989. Cambridge Books Online.

[98] T. Odlin. *Language transfer: Cross-linguistic influence in language learning*. Cambridge University Press, 1989.

[99] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[100] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[101] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[102] P. Ramachandran, P. J. Liu, and Q. V. Le. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683, 2016.

[103] H. Ringbom. *Cross-linguistic similarity in foreign language learning*, volume 21. Multilingual Matters, 2007.

[104] A. J. Robinson and F. Fallside. The Utility Driven Dynamic Error Propagation Network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[105] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. 1996.

[106] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[107] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015.

[108] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

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

[110] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[111] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015.

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

[113] C. E. Shannon and W. Weaver. The mathematical theory of communication. *Urbana: University of Illinois Press*, 1949.

[114] H. Siegelmann and E. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132 – 150, 1995.

[115] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4:77–80, 1991.

[116] H. T. Siegelmann and E. D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331 – 360, 1994.

[117] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231, 2006.

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

[119] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.

[120] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[121] J. M. T. Thompson and H. B. Stewart. *Nonlinear dynamics and chaos*. John Wiley & Sons, 2002.

[122] T. Tieleman and G. Hinton. RMSprop Gradient Optimization.

[123] P. D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[124] M. Utiyama and H. Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *HLT-NAACL*, pages 484–491, 2007.

[125] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc., 2015.

[126] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014.

[127] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[128] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988.

[129] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, June 1989.

[130] H. Wu and H. Wang. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181, 2007.

[131] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian,

N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[132] C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *CoRR*, abs/1304.5634, 2013.

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

[134] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *International Conference on Computer Vision*, 2015.

[135] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[136] J. Zhang and C. Zong. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*, pages 1535–1545, 2016.

[137] B. Zhao and Y. Al-Onaizan. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 572–581, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

[138] B. Zhao and S. Chen. A simplex armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *HLT-NAACL*, pages 21–24, 2009.

[139] B. Zoph and K. Knight. Multi-source neural translation. *CoRR*, abs/1601.00710, 2016.

[140] B. Zoph and K. Knight. Multi-source neural translation. In *NAACL*, 2016.

[141] B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. *CoRR*, abs/1604.02201, 2016.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Firat, Orhan
Born In: Turkey
Date and Place of Birth: 1984, Ankara
Phone1: +1 914 314 7451
Phone2: +1 650 499 1890

EDUCATION

Degree	Institution	Year of Graduation
M.S.	METU	2011 (transfer to Post BSc. PhD.)
B.S.	Military Academy	2007
High School	Maltepe Military High School	2003

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2017 - ongoing	Google Research	Research Scientist
2016 - III	Facebook AI Research	Visiting Researcher
2016 - II	New York University	Visiting PhD Student
2016 - I	IBM TJ Watson Research Center	Research Scholar
2014-2015	Montreal Institute for Learning Algorithms (MILA)	Visiting PhD Student
2011 - 2014	Middle East Technical University	Research Assistant
2009 - 2011	Cybersoft	Software Developer

PUBLICATIONS

International Conference Publications

1. S. Jean*, S. Lauly*, **Firat, O.***, K. Cho, Neural Machine Translation for Cross-Lingual Pronoun Prediction. *in preparation*, 2017. *equal contribution

2. S. Jean, S. Lauly, **Firat, O.**, K. Cho, "Does Neural Machine Translation Benefit from Larger Context?". arXiv:1704.05135, 2017.

3. R. Sennrich, **Firat, O.**, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. LÃubli, A. Valerio Miceli Barone, J. Mokry and M. Nadejde, "Nematus: a Toolkit for Neural Machine Translation". In Proceedings of the Demonstrations at EACL'17.

4. H. Schwenk, K. Tran, **Firat, O.**, M. Douze, "Learning Joint Multilingual Sentence Representations with Neural Machine Translation". arXiv:1704.04154, 2017.

5. **Firat, O.**, B. Sankaran, K. Cho, Fatos T. Yarman Vural, Y. Bengio, Multi-Lingual Neural Machine Translation. *Computer Speech and Language, Deep Learning for Machine Translation Special Edition*, 2016.

6. C. Gulcehre*, O. Firat*, K. Cho, Y. Bengio, "On Integrating a Language Model Into Neural Machine Translation", Computer Speech and Language, 2016.*equal contribution

7. **Firat, O.**, B. Sankaran, Y. Al-Onaizan, Fatos T. Yarman Vural, K. Cho, Zero-Resource Translation with Multi-Lingual Neural Machine Translation. *to appear in EMNLP*, 2016.

8. **Firat, O.**, K. Cho, Y. Bengio, Multi-Lingual Neural Machine Translation with Shared Attention Mechanism. *NAACL-HLT*, 2016
(**Best Paper Runner-up**).

9. S. Jean*, **Firat, O.***, K. Cho, R. Memisevic, Y. Bengio, Montreal Machine Translation System for WMT'15. *EMNLP 10th Workshop on Statistical Machine Translation*, 2015. *equal contribution

10. C. Gulcehre*, **Firat, O.***, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, Y. Bengio, On Using Monolingual Corpora in Neural Machine Translation. *arXiv:1503.03535*, 2015. *equal contribution

11. **Fırat, O.**, E. Aksan, I. Öztekin, F.T. Yarman Vural. Learning Deep Temporal Representations for fMRI Brain Decoding. *International Conference on Machine Learning - Medical Imaging Workshop*, 2015.

12. **Fırat, O.**, I. Öztekin, F.T. Yarman Vural. Deep Learning for Brain State Decoding. *13th IEEE International Conference on Image Processing (ICIP)*, 2014.

13. **Fırat, O.**, Can, G., F.T. Yarman Vural. Representation Learning for Contextual Object and Region Detection in Remote Sensing. *22nd International Conference on Pattern Recognition (ICPR),* 2014.

14. **Fırat, O.**, I. Önal, E. Aksan, B. Velioglu, I. Öztekin, F.T. Yarman Vural. Large Scale Functional Connectivity for Brain Decoding. *11th IASTED International Conference on Biomedical Engineering (BioMed)*, 2014.

15. B. Velioglu, E. Aksan, I. Önal, **Fırat, O.**, F.T. Yarman Vural. Functional Networks of Anatomic Brain Regions. *13th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC)*, 2014.

16. I. Önal, E. Aksan, B. Velioglu, **Fırat, O.**, M. Özay, I. Öztekin, F.T. Yarman Vural. Modelling the Brain Connectivity for Pattern Analysis. *22nd International Conference on Pattern Recognition (ICPR),* 2014.

17. **Fırat, O.**, M. Özay, I. Önal, I. Öztekin, F.T. Yarman Vural. Enhancing Local Linear Models Using Functional Connectivity for Brain State Decoding. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2014.

18. **Fırat, O.**, M. Özay, I. Öztekin, F.T. Yarman Vural. Discriminative Functional Connectivity Measures for Brain Decoding. *CoRR. abs/1402.5684*, 2014.

19. I. Önal, E. Aksan, B. Velioglu, **Fırat, O.**, M. Özay, I. Öztekin, F.T. Yarman Vural. Estimating Brain Connectivity for Pattern Analysis. ***accepted** from IEEE 22th Conference on Signal Processing and Communications Applications (SIU)*, 2014.

20. **Fırat, O.**, M. Özay, I. Önal, I. Öztekin, F.T. Yarman Vural. Functional Mesh Learning for Pattern Analysis of Cognitive Processes. *IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC)*. 2013 (**Best Paper Award**).

21. I. Önal, M. Özay, **Fırat, O.**, I. Öztekin, F.T. Yarman Vural. An Information Theoretic Approach to Classify Cognitive States Using fMRI. *13th IEEE International Conference on BioInformatics and BioEngineering (BIBE)*,2013.

22. Ekmekci, O. **Fırat, O.**, M. Özay, I. Öztekin, F.T. Yarman Vural. O. Mesh Learning for Object Classification using fMRI Measurements,*12th IEEE International Conference on Image Processing (ICIP)*, 2013.

23. **Fırat, O.**, M. Özay, I. Önal, I. Öztekin, F.T. Yarman Vural. Representation of Cognitive Processes Using the Minimum Spanning Tree of Local Meshes. *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*,2013.

24. I. Önal, M. Özay, **Fırat, O.**, I. Öztekin, F.T. Yarman Vural. Analyzing the Information Distribution in the fMRI measurements by estimating the degree of locality. *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*,2013.

25. **Fırat, O.**, F.T. Yarman Vural. Representation Learning with Convolutional Sparse Autoencoders for Remote Sensing. *IEEE 21th Conference on Signal Processing and Communications Applications (SIU)*,2013.

26. **Fırat, O.**, M. Özay, I. Önal, I. Öztekin, F.T. Yarman Vural. Cognitive Processes Representation Using Minimum Spanning Tree of Local Meshes. *IEEE 21th Conference on Signal Processing and Communications Applications (SIU)*,2013.

27. Can, G., **Fırat, O.**, F.T. Yarman Vural. Contextual Object Recognition with Conditional Random Fields. *IEEE 21th Conference on Signal Processing and Communications Applications (SIU)*,2013.

28. Aktas, R., **Fırat, O.**, F.T. Yarman Vural. Dry Dock Detection in Satellite Images with Representation Learning. *IEEE 21th Conference on Signal Processing and Communications Applications (SIU)*,2013.

29. I. Önal, M. Özay, **Fırat, O.**, I. Öztekin, F.T. Yarman Vural. Analyzing the Information Distribution in the fMRI measurements by estimating the degree of locality. *IEEE 21th Conference on Signal Processing and Communications Applications (SIU)*,2013.

30. Can, G., **O. Fırat**, F.T. Yarman Vural. Conditional Random Fields for Land Use/Land Cover Classification and Complex Region Detection. *14th International Workshop on Structural and Syntactic Pattern Recognition (SSPR)*. 2012.

31. **Fırat, O.**, O.T. Tursun, F.T. Yarman Vural. Application of Context Invariants in Airport Region of Interest Detection for Multi-Spectral Satellite Imagery. *IEEE 20th Conference on Signal Processing and Communications Applications (SIU)*,2012.

32. **Fırat, O.**, M. Özay, I. Önal, I. Öztekin, F.T. Yarman Vural. A Mesh Learning Approach for Brain Data Modelling. *IEEE 20th Conference on Signal Processing and Communications Applications (SIU)*,2012.

33. **Fırat, O.**, A. Temizel. Parallel Spectral Graph Partitioning on CUDA. *GPU Technology Conference, San Jose, California (GTC)*,2012.